SKRIPSI

PENGEMBANGAN APLIKASI SISTEM INFORMASI PEMESANAN LAPANGAN FUTSAL

Disusun Oleh :

GEMA MARDHIKA WICAKSONO

NIM 05.12.732

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

AGUSTUS 2010

LEMBAR PERSETUJUAN

PENGEMBANGAN APLIKASI SISTEM INFORMASI PEMESANAN LAPANGAN FUTSAL

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Komputer dan Informatika Strata Satu (S-1)

Disusun oleh :

GEMA MARDHIKA WICAKSONO

NIM : 05.12.732

Diperiksa dan Disetujui

Mengetahui, Ketua Jurusan Teknik Elektro S-1	Dosen Pembimbing
Ir. Yusuf Ismail Nakhoda, MT	**Joseph Dedy Irawan,ST,MT**
NIP.Y 1018800189	**NIP. 19740416 200501 1 002**

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2010

A B S T R A K S I

PENGEMBANGAN APLIKASI SISTEM INFORMASI

PEMESANAN LAPANGAN FUTSAL

(GEMA MARDHIKA WICAKSONO, Nim 05.12.732, T.KOMP. & INFORMATIKA S-1)
(Dosen Pembimbing : Joseph Dedy Irawan,ST,MT.)

Kata Kunci : Telepon Seluler, GPRS , Mobile

Teknologi semakin pesat membawa era mobilitas disegala faktor, salah satunya teknologi pada telepon seluler adalah layanan GPRS. Telepon seluler sekarang ini bukan hanya digunakan untuk melakukan telp saja tetapi juga sudah masuk kepelayanan di bidang bisnis salah satunya penyelenggaranya adalah operator – operator telepon seluler . Contohnya aplikasi jual beli atau pemesanan cepat oleh XL dan I-Iklan oleh Indosat. Aplikasi ini memiliki kekurangan yaitu tidak adanya tanggung jawab operator atas data yang dilayankan.

Agar data yang dilayankan benar – benar keaslinya maka harus ada penyelenggara yang bertanggung jawab atas data tersebut seperti membangun aplikasi diserver, untuk melakukan transaksi atas data tersebut user harus terkoneksi ke server sehingga ada timbal balik atas data tersebut.

Aplikasi pemesanan lapangan futsal ini akan menghubungkan user dengan pelayan dengan teknologi telepon seluler. Dengan begitu terbangun rasa saling percaya atas transaksi yang dilakukan dan memudahkan penggunanya.

KATA PENGANTAR

Dengan mengucap syukur kehadirat Allah SWT yang dengan segala Kasih dan Anugerah – Nya, telah memberikan kekuatan, kesabaran, bimbingan dan perlindungan sehingga penulis dapat menyelesaikan laporan skripsi dengan judul : **"PENGEMBANGAN APLIKASI SISTEM INFORMASI PEMESANAN LAPANGAN FUTSAL"**

Pembuatan skripsi ini disusun guna memenuhi syarat akhir kelulusan pendidikan jenjang Strata–1 di Institut Teknologi Nasional Malang. Dalam penyusunan skripsi ini penulis banyak mendapat bantuan baik moril maupun materil, saran dan dorongan semangat dari berbagai pihak, untuk itu penulis mengucapkan terima kasih kepada :

1. Bapak Prof. Dr. Ir. Abraham Lomi, MSEE., selaku Rektor ITN Malang.
2. Bapak Ir. Sidik Noertjahyono, MT., selaku Dekan Fakultas Teknologi Industri.
3. Bapak Ir. Yusuf Ismail Nakhoda, MT., selaku Ketua Jurusan Teknik Elektro S – 1 ITN Malang
4. Bapak Joseph Dedy Irawan, ST, MT., selaku Dosen Pembimbing .
5. Semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini yang tidak bisa penulis sebutkan satu persatu.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang membangun sangat diharapkan.

Akhir kata, penulis mohon maaf kepada semua pihak bilamana selama penyusunan skripsi ini penyusun membuat kesalahan secara tidak sengaja dan semoga skripsi ini dapat bermanfaat bagi kita semua. Amiin.

Malang, Agustus 2010

Penulis

DAFTAR ISI

LEMBAR PERSETUJUAN ..... **i**

ABSTRAK ..... **ii**

KATA PENGANTAR ..... **iii**

DAFTAR ISI ..... **v**

DAFTAR GAMBAR ..... **viii**

DAFTAR TABEL ..... **xi**

BAB I PENDAHULUAN **1**

1.1. Latar Belakang 1

1.2. Rumusan Masalah 3

1.3. Tujuan 3

1.4. Batasan Masalah 3

1.5. Metodologi 4

1.6. Sistematika Penulisan 5

BAB II DASAR TEORI **7**

2.1. Pemesanan 7

2.2. Java 9

2.2.1. J2ME 10

2.2.1.1. Configuration 11

2.2.1.2. Profile 11

2.2.1.3. CLDC 12

2.2.1.4. CDC 12
2.2.1.5. MIDP 12
2.2.1.6. KVM 13
2.2.1.7. CVM 14
2.2.1.8. MIDlet 14
2.2.1.9. Alur Hidup MIDlet 14
2.2.1.10. GCF 16
2.3. WAP 17
2.4. GPRS 17
2.5. Web Server Apache 18
2.6. PHP 19
2.7. MySQL 20

BAB III PERANCANGAN DAN PEMBUATAN SISTEM 22
3.1. Spesifikasi Sistem 22
3.1.1. Spesifikasi Software 22
3.1.2. Spesifikasi Hardware 26
3.2. Desain Keseluruhan Sistem 26
3.2.1. Desain Komunikasi HTTP 28
3.2.2. Desain Transaksi 33
3.2.3. Desain Konten 35
3.2.4. Desain Server 36
3.2.5. Desain Aplikasi Di Server 37

3.2.6. Desain Database 38

BAB IV IMPLEMENTASI DAN PENGUJIAN 41

4.1. Implementasi Sistem 41

4.1.1. Login 41

4.2. Pengujian Hasil 43

4.2.1. Pengujian Lapangan 44

4.2.2. Pengujian Pemesanan 45

4.2.2.1. Pengujian Login 45

4.2.2.2. Pengujian Pemilihan Lapangan 47

4.2.2.3. Pengujian Pemesanan 48

4.2.3. Pengujian Pembatalan 50

4.2.3.1. Pengujian Login Pembatalan 51

4.2.3.2. Pengujian Penampilan Lapangan Dipesan 51

4.2.3.3. Pengujian Pembatalan 52

4.2.3.4. Pengujian Report Pembatalan 53

4.2.4. Pengujian Menu Info 53

4.2.5. Pengujian Login Aplikasi Web 54

4.2.6. Pengujian Menu Lapangan 55

4.2.7. Pengujian Menu Data 57

4.2.8. Pengujian Laporan 59

BAB V PENUTUP **60**

5.1. Kesimpulan 60

5.2. Saran 60

DAFTAR PUSTAKA **62**

LAMPIRAN – LAMPIRAN

DAFTAR GAMBAR

BAB II DASAR TEORI

Gambar 2.1. Ruang Lingkup Hubungan J2ME,J2SE, dan J2EE 10

Gambar 2.2. Arsitektur J2ME 11

Gambar 2.3. Arsitektur J2ME MIDP 14

Gambar 2.4. Alur Hidup MIDlet 15

Gambar 2.5. Connection GCF 16

BAB III PERANCANGAN DAN PEMBUATAN SISTEM

Gambar 3.1. Cara Kerja PHP 24

Gambar 3.2. Emulator Sony Ericsson W200i 25

Gambar 3.3. Sony Ericsson W200i 26

Gambar 3.4. Desain keseluruhan Aplikasi Pemesanan Lapangan Futsal 27

Gambar 3.5. Desain Komunikasi Hardware 29

Gambar 3.6. Flowchart HTTP 31

Gambar 3.7. Desain Login 33

Gambar 3.8. Desain Flowchart Program 35

Gambar 3.9. Diagram Konten 36

Gambar 3.10. Flowchart Server 37

Gambar 3.11. Diagram Konten 38

Gambar 3.10. Database 38

BAB IV IMPLEMENTASI DAN PENGUJIAN HASIL

Gambar 4.1. Form Login 42

Gambar 4.2. Pemograman login PHP pada Server 42

Gambar 4.3. Pemograman Login Java 43

Gambar 4.4. Form Menu Utama 44

Gambar 4.5. Form Lihat Lapangan 45

Gambar 4.6. Form login Pada Aplikasi 46

Gambar 4.7. Konfirmasi Kesalahan Menulis Nama dan Password 46

Gambar 4.8. Persutujuan Mengirim Data dan menerima Data Dari Server..... 46

Gambar 4.9. Nama Dan Password Telah DiValidasi Oleh Server..... 47

Gambar 4.10. Pemilihan Produk Yang Dipesan 48

Gambar 4.11. Lihat Lapangan padaTransaksi Pemesanan..... 48

Gambar 4.12. Form Pemesanan 49

Gambar 4.13. Form Report 49

Gambar 4.14. Form jika Salah Memasukkan Kode 50

Gambar 4.15. Form Jika Lapangan telah di pesan 50

Gambar 4.16. Login Pada Pembatalan 51

Gambar 4.17. Form Lapangan Yang Sudah DiPesan..... 52

Gambar 4.18. Form Pembatalan 52
Gambar 4.19. Konfirmasi Server Pembatalan Sukses 53
Gambar 4.20. Menu Info 54
Gambar 4.21. Menu Login 54
Gambar 4.22. Konfirmasi Login 55
Gambar 4.23. Menu Lapangan 55
Gambar 4.24. Detail Lapangan 56
Gambar 4.25. Lapangan DiPesan 56
Gambar 4.26. Data Edit 57
Gambar 4.27. Form Edit 57
Gambar 4.28. Konfirmasi Data DiHapus 58
Gambar 4.29. Form Tambah 58
Gambar 4.30. Laporan 59

DAFTAR TABEL

BAB IV IMPLEMENTASI DAN PENGUJIAN HASIL

Tabel 3.1. Tabel Login 39

Tabel 3.2. Tabel Pesan 39

Tabel 3.3. Tabel Lapangan1 39

Tabel 3.4. Tabel Lapangan2 39

Tabel 3.5. Tabel Lapangan3 40

Tabel 3.6. Tabel Tanggal 40

Tabel 3.7. Tabel Pelanggan 40

BAB I

PENDAHULUAN

1.1. LATAR BELAKANG

Perkembangan teknologi yang semakin pesat, membuat kehidupan manusia menjadi lebih mudah dan praktis. Perkembangan teknologi yang selalu mengahadirkan kemudahaan bagi penggunanya salah satunya adalah teknologi telepon seluler. Telepon seluler atau *Handphone* merupakan perangkat telekomunikasi elektronik yang digunakan untuk komunikasi suara atau data secara *mobile* tanpa menggunakan kabel. Fungsi telepon seluler bukan hanya digunakan berkomunikasi saja, dengan fitur – fitur tambahan seperti *Internet*, kamera digital, *mobile* tv, perangkat *office* dan presentasi, perangkat multimedia, media koneksi *bluetooth* , *infrared*, *wireless* LAN, USB dan GPS, bahkan dapat memiliki kapasitas penyimpanan 32 GB. Telepon seluler bisa di bilang perangkat yang pintar dan canggih.

Telepon seluler sudah menjadi kebutuhan umat manusia saat ini. Pemakainyapun sudah semua kalangan karena harga yang sudah terjangkau dan dapat digunakan untuk berbagai keperluan termasuk keperluan bisnis. Transaksi – transaksi bisnis yang memerlukan mobilitas tinggi dan *real time* dapat menggunakan perangkat telepon seluler.

Dalam dunia bisnis kepuasan pelanggan adalah hal yang paling penting, untuk itu berbagai fasilitas diciptakan guna menciptakan pelayanan secara maksimal. Pelayan secara *mobile* telah banyak dikeluarkan *provider* GSM

seperti i-Iklan oleh Indosat, Pemesanan cepat oleh XL, *Mobile message board* oleh Telkomsel dan dari perbankan ada SMS Banking yang sempat menjadi sorotan *public*. Aplikasi penunjang dalam dunia bisnis pun di harapkan mampu memberikan kemudahaan secara meluas, murah dan dapat berinteraksi langsung melalui media *mobile*.

Media bisnis *mobile* diatas memiliki beberapa kelebihan dan kekurangan. Kelebihanya adalah tinggal pakai dalam penggunaanya dan sudah adanya *reward* pulsa dari operator jika melakukan registrasi dan memenuhi persyaratan. Kekuranganya adalah tidak adanya kepastian tanggung jawab atas resiko jika informasi yang diberikan palsu atau bohong karena operator hanya menyediakan jasa dan tidak mengetahui apakah data yang diberikan pemakai benar atau tidak. Hal ini yang mendasari pembuatan aplikasi pemesanan lapangan Futsal berbasis seluler menggunakan J2ME (Java 2 Micro Edition) yang nantinya pemesan harus menginstall di *handphone* dan akan selalu terkoneksi ke *server* dalam setiap transaksi pemesanan.

Dalam pembuatan aplikasinya menggunakan J2ME karena java sudah banyak terintegrasi diperangkat telepon seluler saat ini. Karena dukungan dari *vendor handphone* yang banyak menyebabkan perkembangan teknologi inipun semakin pesat. Dengan teknologi J2ME dapat di desain sebuah aplikasi – aplikasi berbasis *mobile* dimana aplikasi tersebut di desain untuk mempermudah penggunanya dalam setiap kegiatanya melalui media perangkat telepon.

1.2. RUMUSAN MASALAH

Berdasarkan latar belakang diatas maka timbul suatu permasalahan bagaimana mendesain aplikasi Pemesanan menggunakan J2Me dan Web server menggunakan PHP

1.3. TUJUAN

Adapun tujuan yang ingin dicapai dalam pelaksanaan skripsi ini adalah : Menghasilkan desain aplikasi pemesanan lapangan futsal yang mempermudah pelanggan dalam melakukan transaksi pemesanan secara *mobile*.

1.4. BATASAN MASALAH

Agar permasalahan mengarah sesuai dengan tujuan maka pembahasan dibatasi pada hal-hal sebagai berikut :

1. Dalam desain sistem terdapat sistem aplikasi pelanggan pada perangkat telepon seluler dan aplikasi di *server* sebagai penyedia layanan.
2. Tidak membahas macam-macam transaksi secara kompleks.
3. Tidak Membahas Teknologi GPRS secara luas, tetapi hanya pada *handphone*.
4. Tidak membahas tentang ilmu pemesanan dan cara kerjanya.
5. Pembahasan ditekankan pada aplikasi di telepon seluler.
6. Tidak membahas Teknologi keamanan jaringan secara luas

1.5. METODOLOGI

Data merupakan sumber atau bahan mentah yang sangat berharga bagi proses menghasilkan informasi. Oleh sebab itu dalam pengambilan data perlu dilakukan penanganan secara cermat dan hati-hati, sehingga data yang diperoleh dapat bermanfaat dan berkualitas.

Dalam pengumpulan data penyusun mengunakan metode sebagai berikut :

1. Studi Lapangan

   Dengan metode ini data-data diperoleh langsung dari sumber yang bersangkutan, dimana peneliti berhadapan langsung dengan obyek yang diteliti, yang dilakukan dengan cara :

   a. Survey

      Teknik pengumpulan data dengan cara terjun secara langsung dan mencatat secara sistematis terhadap obyek masalah.

   b. Wawancara / Interview

      Teknik pengumpulan data dengan jalan mengadakan komunikasi atau tanya jawab secara langsung dengan pimpinan atau pegawai tentang sistem yang diterapkan.

2. Studi Pustaka / Literatur

   Pengumpulan data ini dilakukan dengan cara mencari bahan-bahan kepustakaan sebagai landasan teori yang ada hubunganya dengan permasalahan yang dijadikan obyek penelitian

3. Perancangan

Merancang dan mengembangkan desain aplikasi pemesanan lapangan futsal yang sesuai dengan transaksi pemesanan umum yang ada di indonesia.

4. Pembuatan Program

Pembuatan aplikasi pemesanan lapangan Futsal dengan Bahasa Pemrograman JAVA.

5. Implementasi dan Pengujian

Merancang dan mengembangkan desain aplikasi pemesanan berdasarkan data-data yang diperoleh.

1.6. SISTEMATIKA PENULISAN

Untuk mempermudah dan memahami pembahasan skripsi ini, maka peneliti menyajikan secara sistematis sebagai berikut :

BAB I : PENDAHULUAN

Berisi Latar Belakang, Perumusan Masalah, Tujuan dan Manfaat Penelitian, Pembatasan Permasalahan, Metodelogi Penelitian dan Sistematika Penulisan.

BAB II : DASAR TEORI

Berisi tentang landasan teori mengenai permasalahan yang berhubungan dengan pembahasan yang dilakukan, yang di

dalamnya memuat teori-teori tentang Pemesanan *Mobile* dan J2ME.

BAB III : PERANCANGAN DAN PEMBUATAN SISTEM

Bab ini membahas Desain dan analisis dari Aplikasi Pemesanan lapangan futsal secara *mobile* pada *hardware* dan *software*.

BAB IV : IMPLEMENTASI DAN PENGUJIAN SISTEM

Berisi tentang Implementasi dari Hasil Desain Aplikasi pada emulator dan pengujian pada aplikasi.

BAB V : PENUTUP

Bab ini tentang kesimpulan yang dapat diambil berdasarkan hasil uraian pada bab–bab sebelumnya dan saran mengenai hasil yang telah diperoleh.

BAB II

TINJAUAN PUSTAKA

2.1. Pemesanan

Secara umum pengertian pemesanan adalah sebuah proses dimana kebutuhan pemesan dan kebutuhan penyedia jasa pemesanan terpenuhi, melalui antar pertukaran informasi dan kepentingan.

Pertukaran informasi dan kepentingan tersebut menimbulkan sebuah transaksi – transaksi pemesanan yang menjadi konsep awal pemesanan lapangan futsal. Transaksi – transaksi pemesanan atau pemindahan hak milik secara komersial atau barang dan jasa itu pada prinsipnya melibatkan dua pihak yaitu pemesan sebagai pihak pertama dan penyedia jasa pemesanan sebagai pihak kedua. Pihak penyedia jasa pemesanan dan pemesan memiliki tujuan masing – masing, dimana pihak penyedia jasa pemesanan ingin menawarkan fasilitas atau jasa sesuai target yang telah direncanakan sedangkan pihak pemesan memerlukan sesuatu barang atau jasa sesuai kebutuhanya. Pemesanan pada umumnya mempunyai tiga tujuan umum, yaitu sebagai berikut :

- Mencapai *volume* pemesanan
- Mendapatkan laba tertentu
- Menunjang pertumbuhan pasar

Dengan Teknologi yang semakin berkembang membuat proses pemesanan sekarang tidak hanya dilakukan dengan bertatap muka namun dengan bantuan Telepon, *Web* dan Telepon Seluler dapat melakukan transaksi

pesan dan memesan yang bertujuan meningkatkan kemudahan dan efisiensi dari sisi penyedia jasa pemesanan maupun pemesan . penyedia jasa dapat meyakinkan kepada pemesan agar dapat berhasil mencapai sasaran pemesanan yang diharapkan dengan menggunakan perangkat *mobile*. Pemesan dapat berinteraksi langsung dengan Penyedia jasa pemesanan secara *mobile* untuk melakukan transaksi – transaksi Pemesanan. Tanpa bertatap muka penyedia jasa pemesanan harus mampu memahami beberapa kriteria yang berkaitan dengan pemesan yang ingin melakukan transaksi pemesanan secara *mobile*, dengan begitu pemesan tidak memiliki keraguan dan kebingungan dalam melakukan transaksi pemesanan secara *mobile*.

Kriteria yang berkaitan dengan pemesanan secara *mobile*, yaitu :

- Jenis dan karakteristik fasilitas yang ditawarkan
- Harga
- Syarat pemesanan, seperti pembayaran, pelayanan sesudah pemesanan.

Kriteria diatas biasanya menjadi pusat perhatian pemesan sebelum melakukan pemesanan secara *mobile*. Dengan perencanaan secara matang dan terarah maka transaksi dapat berjalan dengan baik yang dapat meningkatkan *volume* pemesanan.

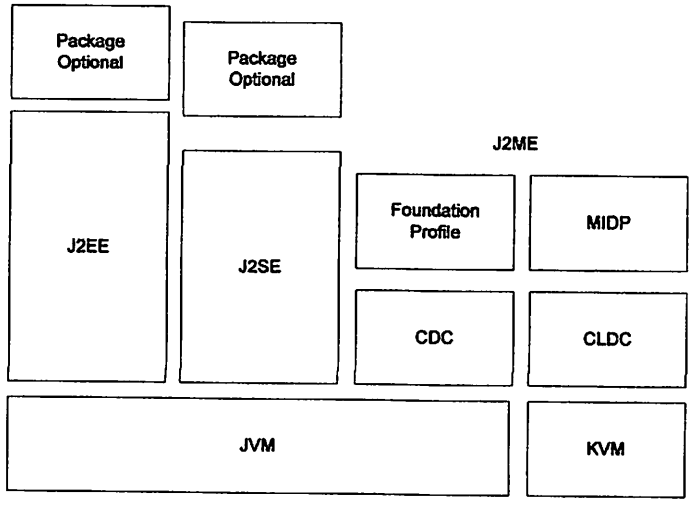
2.2. Java

Java merupakan bahasa pemograman komputer berorientasi objek yang disusun oleh James Gosling dan teman-teman pada perusahaan perangkat lunak Sun Microsystem pada tahun 1991. Java menurut definisi Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Java memiliki sebuah *interprenter* yang disebut Java Virtual Machine (JVM). JVM inilah yang akan membaca program dan menginterprenter kebahasa mesin setelah program dicompile menjadi kode objek yang disebut *bytecode*(*.class*) oleh *compiler*. Dengan konsep *bytecode* inilah java dikenal dengan istilah" *write once, run anywhere* ".

Java 1 merupakan versi awal java yang pada saat perilisanya disebut JDK (Java Development Kit). Didalam JDK semua kebutuhan untuk pengembangan program dan eksekusi program masih tergabung menjadi satu, setelah Java 1.2 dirilis Sun Microsystem memisahkan bagian pengembangan program dengan nama JSDK (Java Software Development Kit) dan eksekusi program dengan nama JRE (Java Runtime Environment). Java 1.2 namanya yang disederhanakan menjadi Java 2. Sun Microsystem mendefinisikan tiga edisi dari java 2 diantaranya adalah J2SE (Java 2 Standart Edition), J2EE (Java 2 Enterprise Edition) dan J2ME (Java 2 Micro Edition).

J2SE adalah lingkungan dasar pemograman java, J2SE digunakan untuk mengembangkan aplikasi – aplikasi *dekstop* dan *applet* (Aplikasi java yang berjalan pada *web browser*). J2EE adalah pengembangan dari J2SE yang

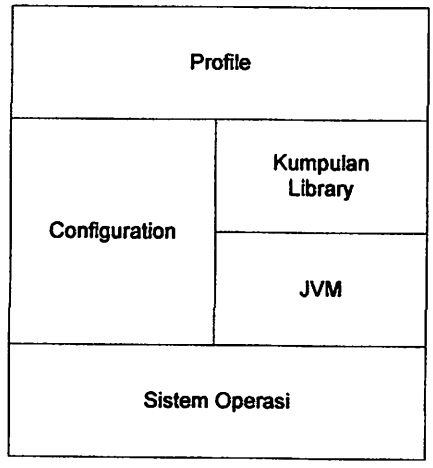
digunakan untuk mengembangkan aplikasi berskala besar seperti aplikasi – aplikasi *server* yang menggunakan EJBs (Enterprise Netbeans), aplikasi *web* dengan menggunakan *Servlet* , JSP (Java Server Page) dan teknologi XML (Extendible Markup Language). J2ME adalah lingkungan pemograman java yang digunakan pada perangkat *mobile*.



Gambar 2.1. Ruang Lingkup Hubungan J2EE, J2SE dan J2ME

2.2.1. J2ME

J2ME (Java 2 Micro Edition) adalah lingkungan pemograman java yang digunakan pada perangkat lunak java pada perangkat *mobile*. J2ME ini merupakan sebuah kombinasi yang terbentuk antara sekumpulan *interface* yang disebut dengan java API (Application Programming Interface) dengan JVM (Java Virtual Machine) yang terbatas. J2ME merupakan bagian dari J2SE karena itu beberapa *library* dapat digunakan pada J2ME dengan kemampuan yang berbeda dikarenakan orientasi dari alat yang digunakan. J2ME juga memiliki beberapa *library* khusus yang tidak dimiliki J2SE.



Gambar 2.2. Arsitektur J2ME

2.2.1.1. Configuration

Configuration merupakan java *library* minimum dan kapabilitas yang dipunya oleh para pengembang J2ME. *Configuration* juga merupakan bagian dari JVM. Di dalam J2ME telah terdefinisi dua buah *configuration* yaitu CLDC (Connected Limited Device Configuration) dan CDC (Connected Device Configuration). CLDC untuk perangkat yang kecil dan CDC untuk perangkat yang lebih besar.

2.2.1.2. Profile

Profile merupakan pengembangan dari konfigurasi. Didalam *profile* ini menyediakan kelas-kelas yang tidak terdapat pada *Configuration. Profile* yang disediakan oleh sun Microsystem yaitu dinamakan dengan MIDP (Mobile Information Device Profile) dan *Foundation Profile.* Hubungan antara *Configuration* dan *Profile* yang ada pada J2ME.

2.2.1.3. CLDC

CLDC (Connected Limited Device Configuration) adalah perangkat dasar dari J2ME dan spesifikasi dasar berupa *library* dari API yang diimplementasikan pada J2ME seperti yang digunakan pada perangkat seluler, pager dan PDA. Perangkat tersebut dibatasi dengan keterbatasan memori dan sumber daya dan kemampuan memproses. Spesifikasi CLDC pada J2ME adalah spesifikasi minimal dari *package*, kelas dan sebagian fungsi *Java Virtual Machine* yang dikurangi agar dapat diimplemetasikan dengan keterbatasan sumber daya pada alat-alat tersebut, JVM yang digunakan disebut KVM (Kilobyte Virtual Machine).

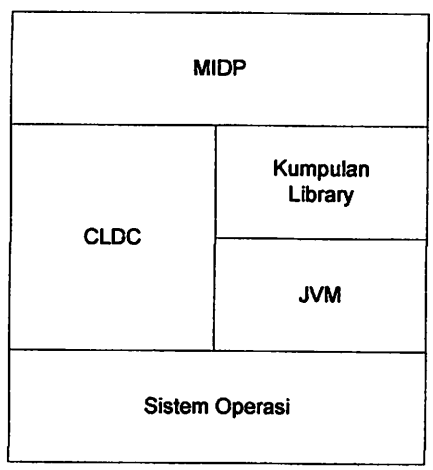
2.2.1.4. CDC

CDC (Connected Device Configuration) adalah spesifikasi dari *configuration* J2ME. CDC terdiri dari *Virtual Machine* dan kumpulan *library* dasar untuk digunakan pada *profile* industri. Implemetasi dari CDC pada J2ME adalah *source code* yang menyediakan sambungan dengan macam – macam platform.

2.2.1.5. MIDP

MIDP (Mobile Information Device Profile) adalah spesifikasi untuk sebuah *profile* J2ME. MIDP memiliki lapisan diatas CLDC, API tambahan untuk daur hidup aplikasi, antarmuka, jaringan, dan penyimpanan *persisten.* Pada saat ini terdapat MIDP 1.0 dan MIDP 2.0 fitur tambahan MIDP 2.0

terdapat dukungan memainkan *tone*, *tone sequence*, dan *file wav* walaupun tanpa adanya *mobile media api* (MMAPI). MIDP *User Interface* API memiliki level tinggi dan rendah. API level rendah merupakan dari *abstrak Canvas* sedangkan level tinggi merupakan dari *abstrak Screen.* API kelas rendah lebih memberikan kemudahan kepada pengembang untuk memodifikasai sesuai kehendaknya, sedangkan API level tinggi biasanya hanya memberikan pengaksesan terbatas.



Gambar 2.3. Arsitektur J2ME MIDP

2.2.1.6. KVM

KVM (Kilobyte Virtual Machine) adalah paket JVM yang didesain untuk perangkat *mobile*. KVM mendukung sebagian dari fitur – fitur JVM, seperti KVM tidak mendukung opersi *floating – point* dan finalisasi objek.

2.2.1.7. CVM

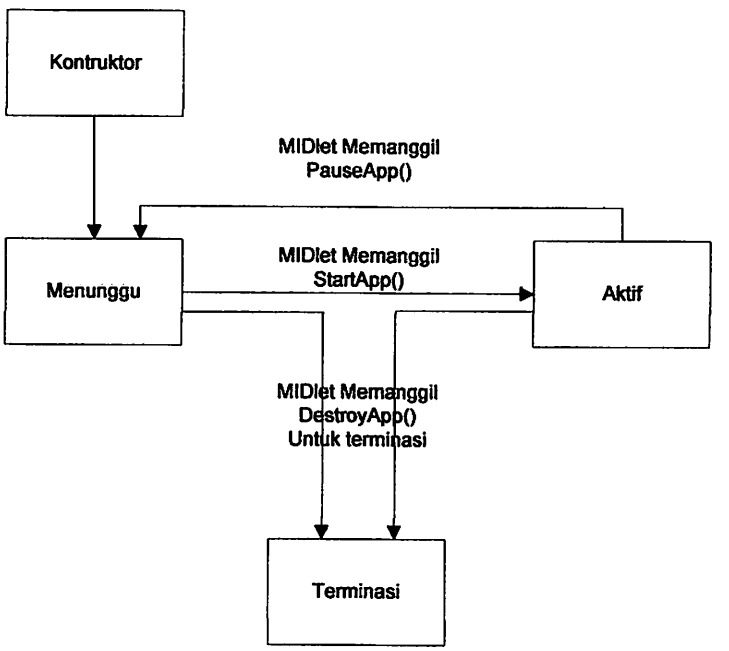
CVM(C Virtual Machine) adalah paket JVM optimal yang digunakan pada CDC. CVM mempunyai seluruh fitur dari *virtual machine* yang didesain untuk perangkat yang memerlukan fitur – fitur Java 2 Virtual Machine.

2.2.1.8. MIDlet

MIDlet adalah aplikasi yang ditulis untuk MIDP. Aplikasi MIDlet adalah bagian dari kelas *javax.microedition,midlet.midlet* yang didefiniskan pada MIDP. MIDlet berupa sebuah kelas abstrak yang merupakan subkelas dari bentuk dasar aplikasi sehingga antar muka antara aplikasi J2ME dan aplikasi manajemen pada perangkat dapat terbentuk.

2.2.1.9. Alur Hidup MIDlet

MIDlet memiliki metode yang harus ada dalam setiap aplikasinya yaitu *constructor(), protected void startApp() throws MIDletstateChangeException, protected void pauseApp(), protected void destroyApp(boolean unconditional) throws MIDletStateChangeExeption.*

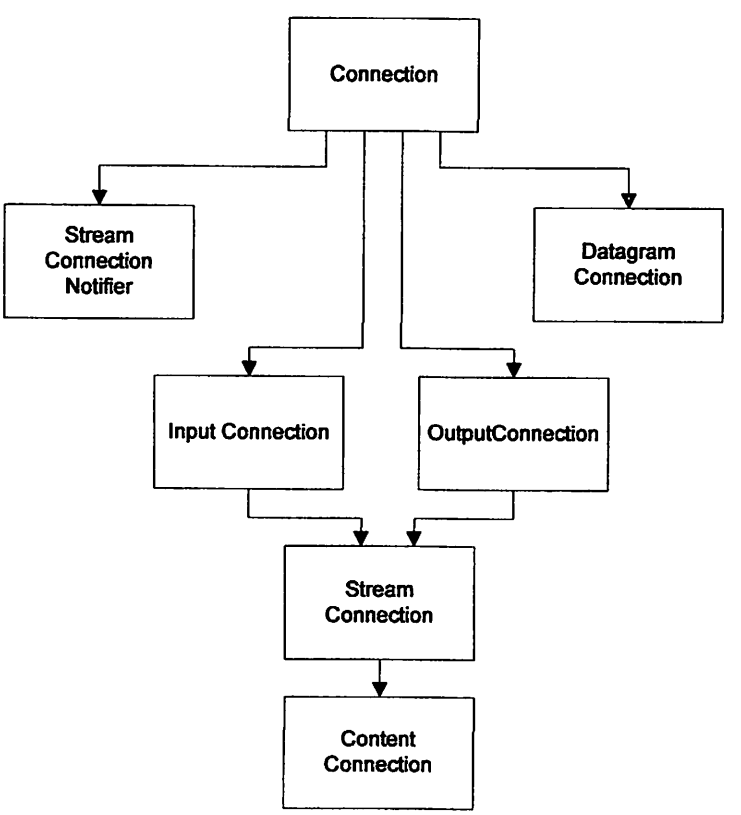


Gambar 2.4. Alur Hidup MIDlet

Ketika MIDlet dijalankan maka akan diinisialisasi dengan kondisi *pause* dan dijalankan *pauseApp()*, kondisi berikutnya adalah fungsi MIDlet dijalankan, yaitu pada *StartApp()*. Metode tersebut diimplementasikan sebagai *protected*, hal ini dimaksudkan agar MIDLet lain tidak memanggil metode tersebut. Pada saat pemakai keluar dari MIDlet, maka metode *DestroyApp()* akan dijalankan sebelum MIDlet benar –benar tidak berjalan lagi.Metode *notifyDestroy()* akan dipanggil sebelum MIDlet benar –benar tidak berjalan lagi, *Destroypp()* akan memanggil *notifyDestroyed()*, dan *notifyDestroyed()* akan memberitahu platform untuk menterminasi MIDlet dan membersihkan semua sumber daya yang mengacu pada MIDlet.

2.2.1.10. GCF(Generic Connection Framework)

Generic Connection Framework adalah *framework* mendukung koneksi *packet* (socket) dan *stream*(datagram). Sesuai dengan namanya, *framework* ini menyediakan API dasar bagi koneksi di CLDC. Framework ini menyediakan pondasi umum dari berbagai koneksi seperti HTTP, socket, dan *datagram.* Walaupun *Bluetooth* dan serial I/O termasuk kedalam API ini, GCF menyediakan satu set API yang lebih *generic* dan mendasar yangmenjadi abstraksi dari berbagai tipe koneksi. Harus dicatat, bahwa tidak semua tipe koneksi dibutuhkan bagi implementasi sebuah MIDP *device.*



Gambar 2.5. Connection CGF

2.3. WAP

WAP (Wireless Application Protocol) adalah sebuah *protokol* atau sebuah teknik *messaging service* yang memungkinkan sebuah telepon genggam digital atau *terminal mobile* yang mempunyai fasilitas WAP, melihat/membaca isi sebuah situs di *internet* dalam sebuah format teks khusus. Ada beberapa versi WAP antara lain WAP 1.2.1 dan 2.0. WAP 1.2.1 hanya dapat menampilkan laman sederhana saja dibandingkan dengan WAP 2.0 yang mendukung bahasa xhtml dan gambar. WAP di buat pertama kali sebagai protokol komunikasi bergerak yang tidak bergantung pada sistem tertentu. WAP dirancang sebagai bagian dari sistem di masa depan sama halnya dengan *Bluetooth* dan GPRS. WAP merupakan *protokol* komunikasi bergerak yang terdiri dari beberapa *layer* dan dapat dijalankan pada sistem jaringan yang berbeda. Teknologi ini merupakan hasil kerjasama antar industri untuk membuat sebuah standar yang terbuka dan berbasis pada standar *Internet*, serta beberapa *protokol* yang sudah dioptimasi untuk lingkungan nirkabel. Teknologi ini bekerja dalam modus teks dengan kecepatan sekitar 9,6 kbps.

2.4. GPRS

GPRS (General Packet Radio Service) adalah suatu teknologi yang memungkinkan pengiriman dan penerimaan data lebih cepat jika dibandingkan dengan penggunaan teknologi *Circuit Switch* Data atau CSD. Sering disebut pula dengan teknologi 2,5G .Sistem GPRS dapat digunakan untuk transfer data (dalam bentuk paket data) yang berkaitan dengan *e-mail*, data gambar (MMS),

dan penelusuran (browsing) internet. Layanan GPRS dipasang pada jenis ponsel tipe GSM dan IS - 136, walaupun jaringan GPRS saat ini terpisah dari GSM.

GPRS merupakan sistem transmisi berbasis paket untuk GSM yang menggunakan prinsip '*tunnelling*'. Ia menawarkan laju data yang lebih tinggi. Laju datanya secara kasar sampai 160 kbps dibandingkan dengan 9,6 kbps yang dapat disediakan oleh rangkaian tersakelar GSM. Kanal-kanal radio ganda dapat dialokasikan bagi seorang pengguna dan kanal yang sama dapat pula digunakan secara berbagi '*sharing*' di antara beberapa pengguna sehingga menjadi sangat efisien. Dari segi biaya, pentarifan diharapkan hanya mengacu pada *volume* penggunaan. Penggunanya ditarik biaya dalam kaitannya dengan banyaknya *byte* yang dikirim atau diterima, tanpa memperdulikan panggilan, dengan demikian dimungkinkan GPRS akan menjadi lebih cenderung dipilih oleh pelanggan untuk mengaksesnya daripada layanan-layanan IP.

2.5. Web Server Apache

Apache HTTP Server adalah *web server* yang merupakan *project Open Source* dalam upaya untuk mengembangkan dan mempertahankan sebuah *open-source* HTTP *server* modern yang efisien, aman dan murah. Apache memiliki fitur – fitur cangih seperti pesan kesalahan yang dapat dikonfigurasi, autentika berbasis data dan lain – lain. Apache juga didukung oleh sejumlah antarmuka pengguna berbasis grafik (GUI) yang memungkinkan penangganan *server* menjadi mudah. Apache merupakan perangkat lunak sumber terbuka

dikembangkan oleh komunitas terbuka yang terdiri dari pengembang – pengembang dibawah naungan Apache Software Foundation.

2.6. PHP

PHP adalah bahasa pemograman *web* dinamis, yang membuat *user* dapat berinteraksi *web*. PHP adalah *open source* sehingga banyak orang yang menyukainya dikarenakan menggunakan , menggembangkan dan mendistribusikan secara gratis dan tidak berlisensi. PHP juga bahasa pemograman yang memiliki sintaks yang mudah dipelajari bahkan untuk kalangan non – *programer*, dapat berjalan pada bermacam – macam operating system seperti Linux, Windows, MacOS dan memiliki support komunitas yang besar. Beberapa kelebiahan PHP.

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaanya.
2. *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulai apache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena memiliki referensi yang banyak.

2.7. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang *multithread, multi-user*, dan berbasis *open source*. Tidak sama dengan Apache yang perangkat lunak dikembangkan oleh komunitas dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing – masing, MySQL dimiliki dan disponsori oleh perusahaan komersial di Swedia MySQL AB, yang memegang hak cipta hampir atas semua kode sumbernya.

MySQL memiliki beberapa keistimewaan, antara lain :

1. Portabilitas. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
2. *Open Source*. MySQL didistribusikan secara *open source*, dibawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.
3. *Multiuser*. MySQL dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. *Performance Tuning*. MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. Jenis Kolom. MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed / unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain.
6. Perintah dan Fungsi. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).

7. Keamanan. MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. Skalabilitas dan Pembatasan. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. Konektivitas. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix soket (UNIX), atau Named Pipes (NT).
10. Lokalisasi. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.
11. Antar Muka. MySQL memiliki interface (antar muka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (Application Programming Interface).
12. Klien dan Peralatan. MySQL dilengkapi dengan berbagai peralatan (tool)yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk online.
13. Struktur tabel. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle.

BAB III

ANALISIS DAN DESAIN SISTEM

3.1. SPESIFIKASI SISTEM

Sistem aplikasi pemesanan lapangan futsal berbasis telepon seluler terdiri atas dua buah program pada *server* sebagai pelayanan dan sebuah program lainya pada *client mobile* sebagai *user interface*. Sistem aplikasi pemesanan lapangan futsal berbasis telepon seluler akan diinstall di telepon seluler *user* dan akan terkoneksi dengan *web server* dalam melakukan transaksi – transaksi pemesanan lapangan futsal. Didalam *web server* terdapat *database*, untuk berkomunikasi dengan *database* aplikasi pemesanan lapangan futsal ini akan melakukan koneksi menggunakan *gprs*. *Web server* akan menerima *request respone* dan membandingkan data yang di*request* dengan data di *database* dalam setiap transaksi- transaksi pemesanan. Di *server* terdapat aplikasi berbasis *web* yang akan memberitahukan adanya *user*/pemesan yang melakukan sebuah transaksi, dan pelayan dapat melakukan pemrosesan terhadap *request user*. Adapun spesifikasi *software* dan *hardware* dalam pada aplikasi pemesanan lapangan futsal ini.

3.1.1 SPESIFIKASI SOFTWARE

Dalam mendesain aplikasi pemesanan lapangan futsal ada beberapa *software* yang dibutuhkan untuk mendukung kinerja desain aplikasi pemesanan *client* maupun aplikasi *server*. *Software - software* tesebut akan selalu digunakan

dalam setiap transaksi pemesanan, adapun *software* yang berada di *server* maupun berada pada *user*/pemesan. *Software* pada *server* berguna untuk membaca dan menulis data di *database* serta menerima *request* dam me*reply* data tersebut ke *user*/pemesan. *Software* pada *user*/pemesan berguna untuk menerima dan mengirim data dari *server*. Adapun spesifikasi *software* pada *server* dan *user*/pemesan pada aplikasi pemesanan lapangan futsal ini.

- **Software Server**

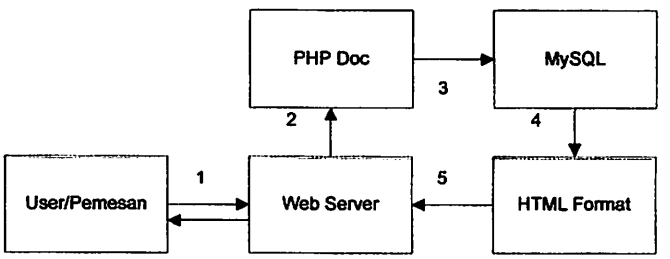
❖ Web server apache

Didalam *web server* apache ini tersimpan data – data *user* dan transaksi – transaksi pemesanan. *Web server* akan menampung *request* dari user dan me*reply* ke telepon seluler user. Dan untuk membangun aplikasi yang berjalan maka dibutuhkan *web server* pada *hosting* idwebhost.com dengan spesifikasi.

1. Web Server : Apache
2. PHP Version : 5.2.6.
3. MySQL Version : 5.0.45
4. Support : Virtual Directory

❖ PHP

Aplikasi yang berjalan di *server* yang akan memberitahukan pemesan akan adanya transaksi pemesanan yang dilakukan oleh *user*/pemesan. Aplikasi ini akan berkomunikasi dengan *database* dan dijalankan menggunakan *web browser*. Adapun aplikasi di *server* menggunakan bahasa pemograman PHP.



Gambar3.1. Cara Kerja PHP

Keterangan :

1. *User* melakukan permintaan ke *Web Server.*
2. Web Server melakukan pengecekan dengan skrip PHP.
3. Skrip PHP melakukan proses dan olahan dengan Database MySQL.
4. Hasil PHP dan olahan Database di baca oleh *Web Server.*
5. *User* menerima data dari Web Server.

- **Software User**

Software pada *user* yang dibutuhkan guna untuk membangun aplikasi pemesanan adalah:

1. JDK

   JDK (Java Delevlopment Kit) adalah lingkungan Java yang harus diinstal sebelum penginstallan aplikasi pendukung lainya. Didalamnya berisi kumpulan kelas – kelas *loader*, paket – paket, kompiler dan *debugger* .

2. JRE

   JRE(Java Runtime Environment) adalah Lingkungan Java yang ada didalamnya adalah JVM dan Library Java. JRE dibutuhkan untuk menjalankan file *bytecode* dalam Java.

3. J2ME Wireless Toolkit

   Adalah peralatan lunak yang menyediakan lingkungan *emulator*, dokumentasi dan contoh – contoh aplikasi java pada perangkat kecil.J2ME WTK berbasiskan CLDC dan MIDP.

❖ **Testing**

Setelah dilakukan pembangunan aplikasi pemesanan di *user*/pemesan dibutuhkan *emulator* untuk digunakan testing terhadap aplikasi tersebut yang berguna untuk menghindari kesalahan dan kerusakan pada *hardware*. Dan jika terjadi *error* dapat di tanggani terlebih dahulu sebelum diuji coba di *hardware* yang sesungguhnya. Emulator yang digunakan adalah *emulator* Sony Ericsson W200i

Gambar3.2. Emulator Sony Ericsson W200i

3.1.2 SPESIFIKASI HARDWARE

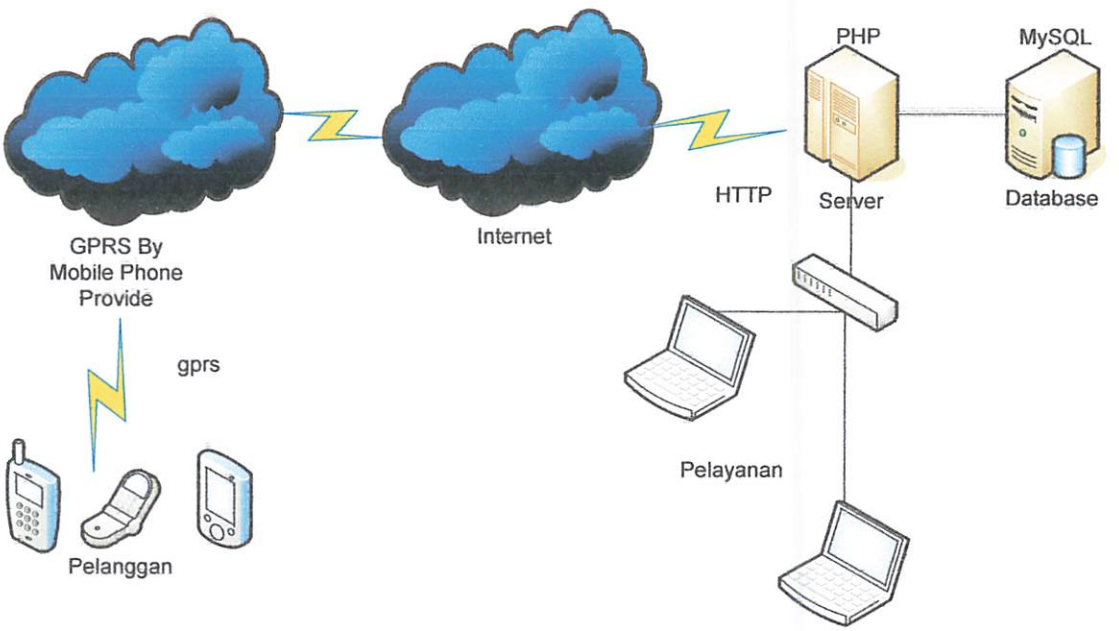
Aplikasi pemesanan lapangan futsal telah dibuat dan diuji coba menggunakan *emulator* maka aplikasi tersebut dijalankan pada *hardware* yang sesungguhnya, *hardware* yang digunakan adalah telepon seluler Sony Ericsson W200i yang nantinya akan diinstall aplikasi pemesanan lapangan futsal tersebut.Adapun spesifikasi hardware.

- CLDC Version : CLDC-1.1.
- MIDP Version : MIDP-2.0

Gambar 3.3. SONY ERICSSON W200i

3.2. DESAIN KESELURUHAN SISTEM

Perancangan desain aplikasi pemesanan lapangan futsal agar dapat dilakukan secara sistematis dan terstuktur maka perlu dibuat blok sistem yang menjelaskan sistem yang dirancang. Secara garis besar perancangan sistem ditunjukan pada gambar berikut :



Gambar 3.4. Desain Keseluruhan Aplikasi Pemesanan Lapangan Futsal

Keterangan :

1. Aplikasi user yang diinstall di handphone pelanggan akan melakukan *request* dan akan menerima *respone* dari *server*. Dalam melakukan *request* aplikasi ini akan membuat sebuah objek dalam hal ini *httpconnection* dengan mengiriman menggunakan sinyal GPRS yang sudah *include* didalam telepon seluler saat ini.
2. Provider GSM menterjemahkan sinyal GPRS menjadi *request* HTTP. Request tersebut diteruskan ke URL yang di tuju dalam hal ini akan berkomunikasi dengan *Internet*.
3. Wap adalah protokol yang digunakan untuk berkomunikasi dengan internet.
4. Server, *web server apache* akan menerima *request* dari *user* dan memproses menggunakan PHP dan akan mengirimkan *respone* menggunakan HTTP ke Provider dan akan diteruskan ke user menggunakan GPRS.

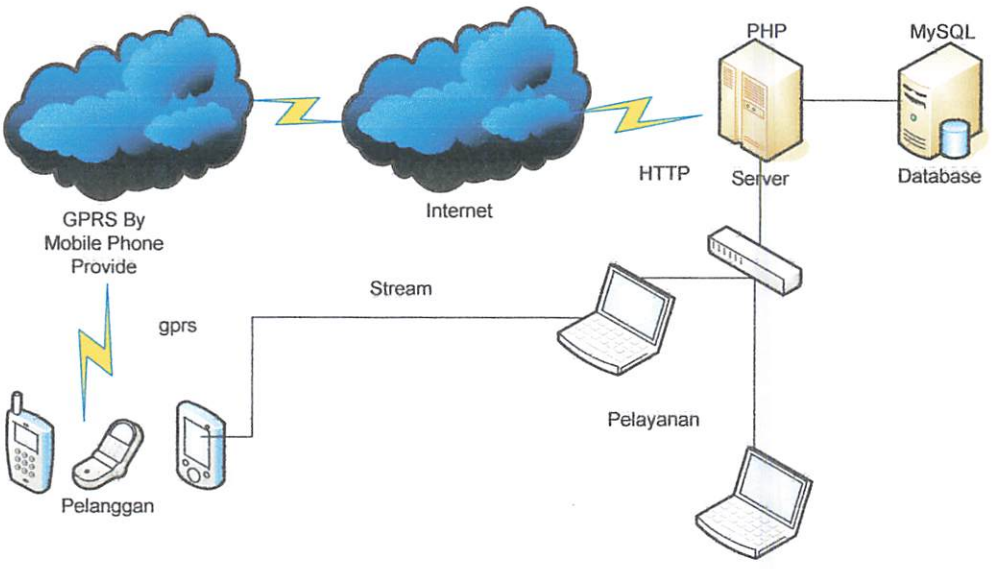
5. PHP mengolah dan penghubung database MySQL. PHP akan melakukan proses sesuai dengan *request* yang dikirimkan dan akan melakukan koneksi ke *database* untuk melakukan perbandingan dengan data dalam database jika sesuai maka akan mengirimkan data tersebut melalui *server* ke *user* jika tidak sesuai maka akan mengirimkan peringatan sesuai fungsinya.
6. Database MySQL untuk menyimpan data user, transaksi – transaksi, dan informasi Lapangan –Lapangan yang akan dilihat oleh user.

3.2.1 DESAIN KOMUNIKASI HTTP

Web server membuka *port* 80 untuk berkomunikasi dengan dunia luar. Maka dalam melakukan komunikasi dengan *web server* aplikasi pemesanan lapangan futsal memanfaatkan *port* tersebut, request dilakukan dengan *url* yang telah ditanaman pada *sricpt java* dan melakukan *request* dengan metode *get.*

- **Analisa Hardware**

  Analisa *hardware* ini membahas aliran koneksi secara *hardware* dimana aliran data melewati beberapa proses sebelum diterima *server*, adapun aliran data. Gambaran aliran data sebagai berikut:



Gambar 3.5. Desain Komunikasi Hardware

Keterangan dari desain sistem diatas.

Cara kerja diatas secara hardware :

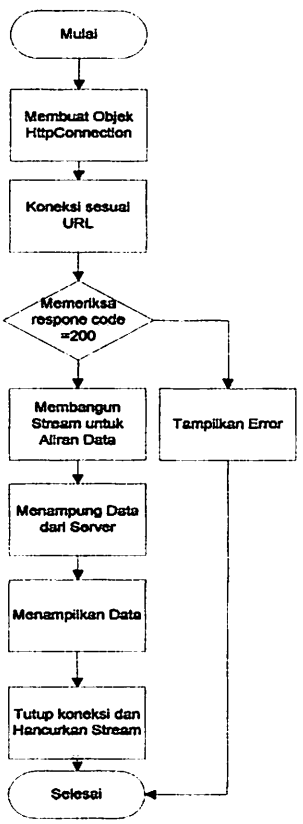
1. User dengan ponsel yang mendukung J2ME diinstall aplikasi pemesanan Lapangan Futsal kemudian melakukan *request* ke *server*, dengan mengakses *port* HTTP 80 dalam komunikasi dengan *server*.Teknologi transfer yang digunakan pada provider GSM adalah GPRS, awalnya *request* dari user adalah berupa sinyal GPRS.
2. Pada provider GSM *request* dari user berupa sinyal GPRS akan diterjemahkan menjadi *request* dengan format HTTP yang nantinya berkomunikasi dengan Protokol Wireles(WAP).
3. Request dari user yang telah diinstall diterjemahkan oleh provider ke dalam format HTTP lalu dialamatkan sesuai URL yang telah dikodekan didalam aplikasi user, dalam hal ini user melakukan *request* ke *server web* yang telah disiapkan dalam *server* local.

4. Server HTTP membuka port 80 untuk berkomunikasi dengan pihak lain, dalam kondisi normal maka *server* bersikap *listening*, menunggu paket yang masuk setelah paket user diterima *server* maka *server* akan memberikan *respon* berupa koneksi (HTTP_OK 200), setelah aplikasi menerima *respon* tersebut maka aplikasi user pada java membentuk hubungan khusus dalam hal ini *Stream*, untuk aliran data khusus antara *user* dan *server*.
5. Lalu *server* melakukan *reply* kepada *user* melalui *route internet* untuk dikembalikan pada alamat pengirim dalam hal ini provider GSM, masih lewat hubungan *Stream* yang diciptakan java, lalu dari provider GSM selanjutnya diterjemahkan kembali dalam bentuk sinyal GPRS dan dikembalikan ke user, setelah user menerima *reply* dari *server* maka hubungan *Stream* tadi dihancurkan oleh *Destroyapp* dalam paket java.

❖ **Analisa Software**

Komunikasi dengan *web server* dilakukan oleh aplikasi *user* dalam hal ini lebih mengacu pada pembangunan *software* J2ME, dalam melakukanya komunikasi dengan server algoritma yang digunakan adalah dengan melemparkan paket *server* sesuai dengan *url* aplikasi setelah itu jika *server* itu ada maka server tersebut akan memberikan nilai kembalian berupa *respone code* jika *respone code* = 200 itu berarti koneksi yang dilakukan berhasil, bila bukan 200 maka aplikasi memanfaatkan *API* dari Java yang support dalam melakukan *httpconnection*, dimana koneksi akan diciptakan

melalui objek untuk mengakses *API*. Komunikasi *API* inilah yang nantinya berkomuniksai dengan *http* seperti pada flowchart berikut :



Gambar 3.6. Flowchart HTTP

Keterangan :

Tahapan dalam melakukan komunikasi dengan server http :

1. Start dan Inisialisaisi system.
2. Membangun objek dari kelas *httpconenction* di java kelas ini berfungsi untuk mengatur koneksi HTTP.
3. Membangun koneksi kesuatu alamat sesuai dengan URL.
4. Dalam melakukan komunikasi dengan *server* maka sebelum melakukan komunikasi sesungguhnya ada yang dinamakan pengecekan untuk mengecek aapkah alamat yang dituju sesuai URL benar – benar ada, selain

itu juga akan mengecek kondisi jaringan, biasanya *server* memberikan respon berupa kode untuk menggambarkan kondisi yang ada.

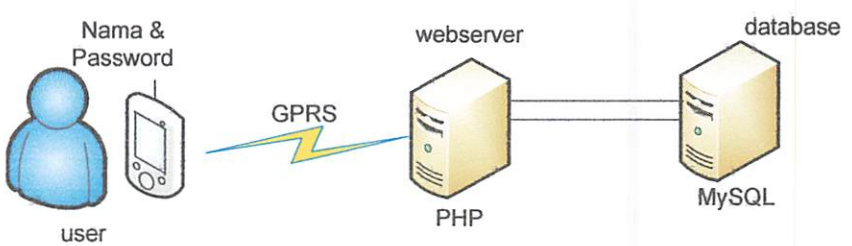
5. Jika Responcode yang dikirm oleh *server* bukan 200 maka dikatakan koneksi yang terajadi tidak bisa dilanjutkan karena bermasalah, kita tidak perlu mengetahui apa masalahnya yang jelas jika respone kode bukan 200 maka koneksi putus dan java membangkitkan *error*.Jika responcode yang dirim oleh *server* adalah 200 maka koneksi bisa dilakukan dan aplikasi java pada user bisa berkomunikasi dengan *web server.*
6. Setelah *server* mengirim responcode dan kode itu bernilai 200 maka java membuat suatu *Stream* dari kelas *inputStreamReader* yang berfungsi untuk melakukan aliran data khusus koneksi ini saja.
7. Setelah *Stream* terbentuk maka *server* menerima *request* dari user dan memprosesnya, setelah *server* memproses maka *server* mengirimkan hasil dari proses, yang mana kiriman dari *server* tersebut ditampung dalam *buffer* yang teradapat dalam inputStreamReader.
8. Pemerikasaan terhadap kiriman *server*, apakah kiriman *server* berisi data atau tidak.
9. Jika hasil dari *server* kosong atau tidak berisi apa bisa saja akses terhadap file PHP tujuan user tidak ada dan pada saat itu akan membangkitkan *error message* oleh java, sedangkan jika kiriman dari *server* berisi data maka ditampilkan pada form Midlet.

10. Setelah *request* dari user diterima dan proses *server* telah didapatkan oleh apliaksi user maka hubungan untuk satu koneksi ditutup dan *Stream* otomatis dihancurkan.

3.2.2 DESAIN TRANSAKSI

❖ **Analisa Hardware**

Analisa hardware ini menjelaskan aliran transaksi secara hardware , adapun desain sebagai berikut :



Gambar 3.7. Desain Login

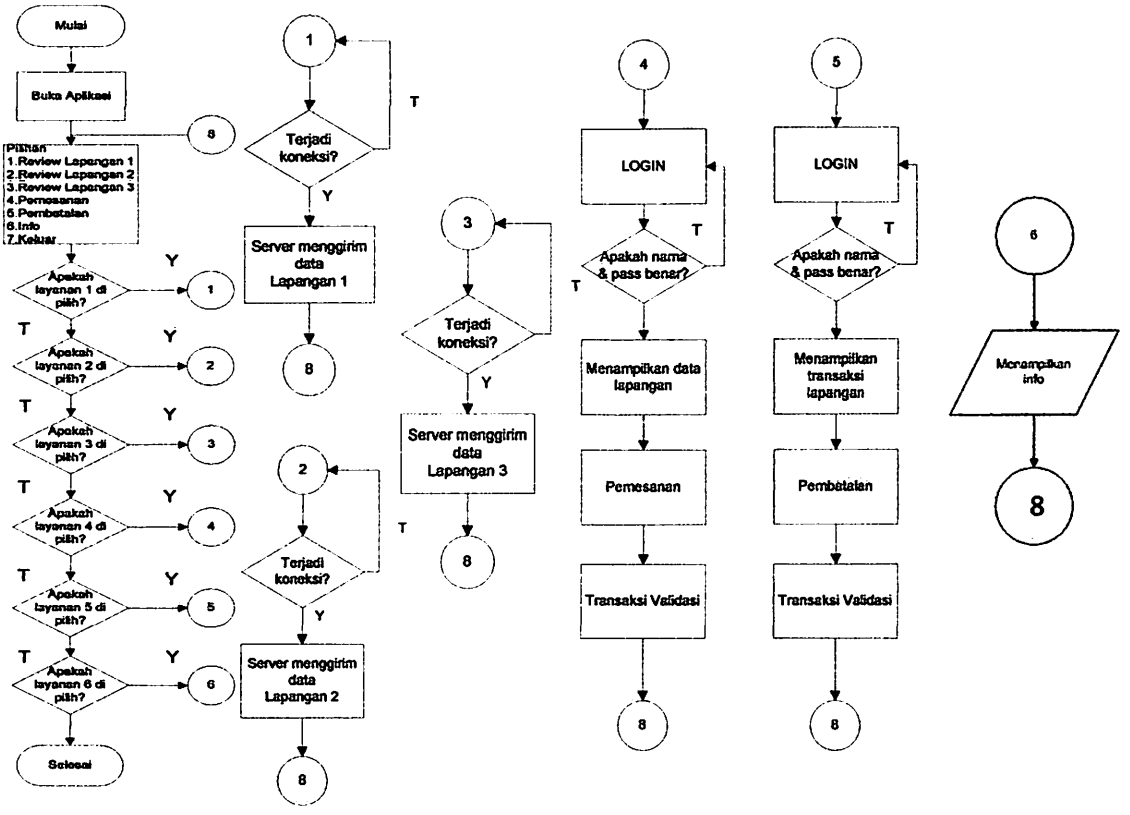
Keterangan :

1. Untuk melakukan transaksi pemesanan lapangan futsal *mobile* sebelumnya *user* harus daftar dahulu dan mendapatkan nama dan *password*..
2. Untuk melakukan transaksi pemesanan ataupun pemabatalan Lapangan Futsal sebelumnya *user* akan mengirimkan nama dan *password* ke *server,* mengirimkan menggunakan GPRS.

3. PHP S*cript* berfungsi sebagai aplikasi di *server* melakukan pemrosesan dengan *database.*
4. Nama dan password yang dikirimkan akan dibandingkan ke *database*, kemudian PHP memproses hasilnya.
5. Setelah database diproses oleh PHP maka *server* mengirimkan kode sesuai dengan hasil proses, jika data Nama dan *password* kirim tidak sesuai maka akan mengirimkan pesan gagal, sedangkan jika nama dan *password* sesuai kode maka akan mengirimkan pesan sukses. Jika pesan berisi gagal maka user tidak bias mengakses bagian transksi selanjutnya, sedangkan jika kode yang dikirim berisi pesan sukses maka *user* bisa mengakses bagian transaksi pada aplikasi Midlet user.

❖ **Analisa Software**

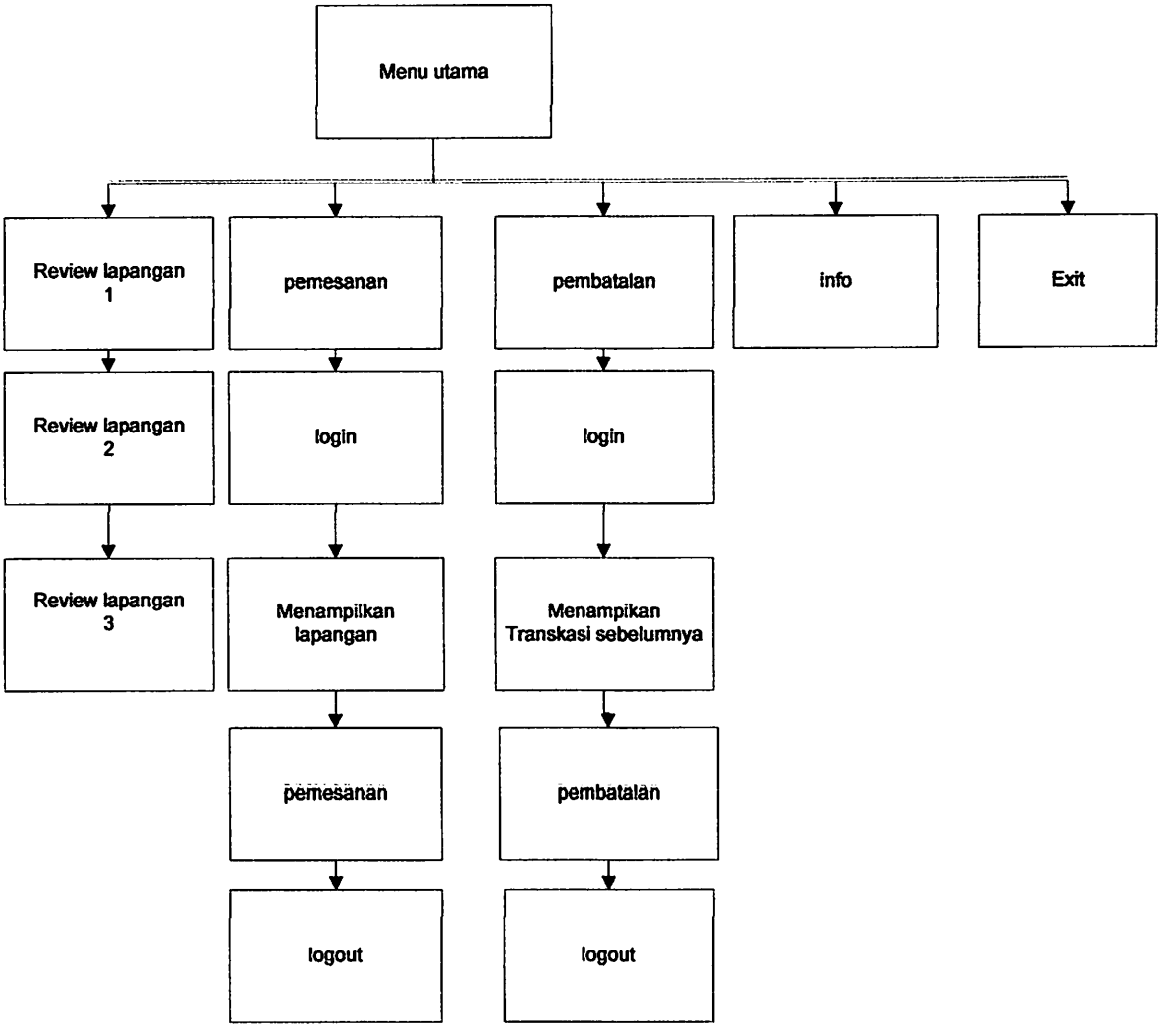
Analisa Software pada aplikasi pemesanan lapangan futsal mobile ini membutuhkan keefisianisasi dikarenakan nantinya aplikasi ini akan menggunakan GPRS yang terkoneksi ke *server* dan pastinya akan membutuhkan biaya setiap transaksi – transaksi yang dilakukan walaupun saat aplikasi ini dibuat perang tarif GPRS antar provider GSM membuat harga GPRS semakin murah , adapun flowchart dari proses transaksi adalah sebagai berikut :



Gambar 3.8. Desain Flowchart Program

3.2.3 DESAIN KONTENT

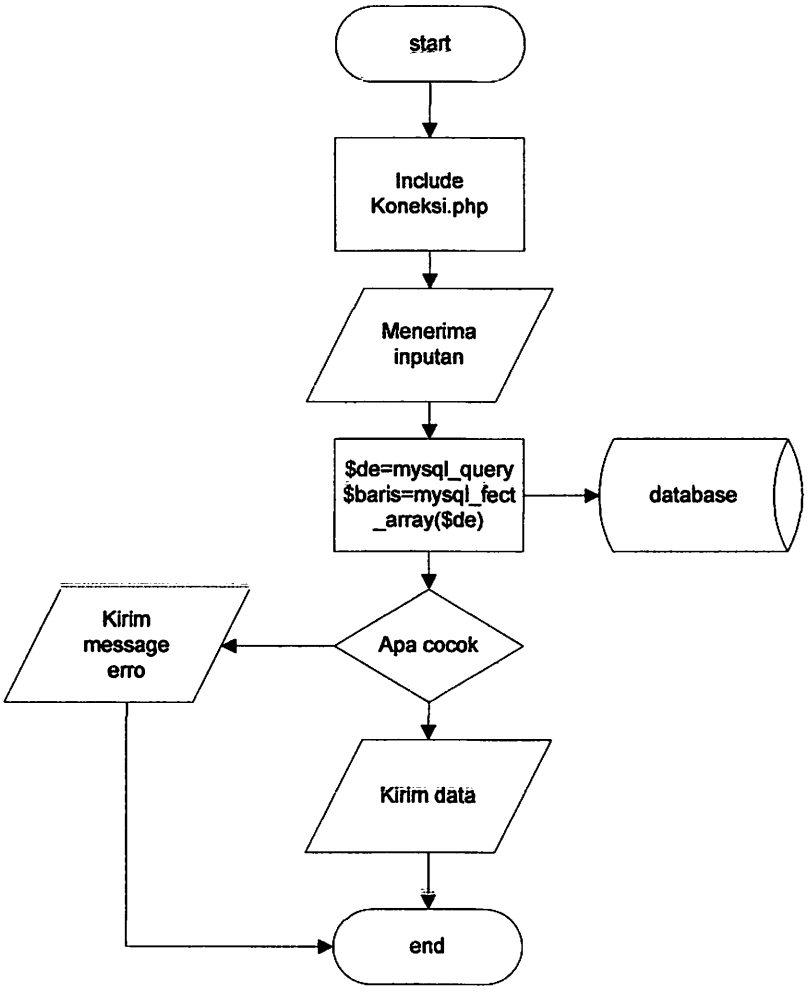
Dalam pembuatan aplikasi pemesanan lapangan futsal *mobile* harus diperhatikan menu menu yang ada pada aplikasi benar – benar diperlukan oleh konsumen. Menu – menu tersebut harus direncanakan dengan baik dan benar, oleh karena itu dirancanglah konten aplikasi yang dapat memenuhi segala kegiatan transksi pemesanan lapangan futsal. Dengan kontent yang direncakana dapat diketahui kekurangan serta kelebihan sehingga pengaplikasianya lebih mudah dan terarah sesuai . Rancangan kontent dalam aplikasi pemesanan adalah sebagai berikut



Gambar 3.9. Diagram Konten

3.2.4 DESAIN SERVER

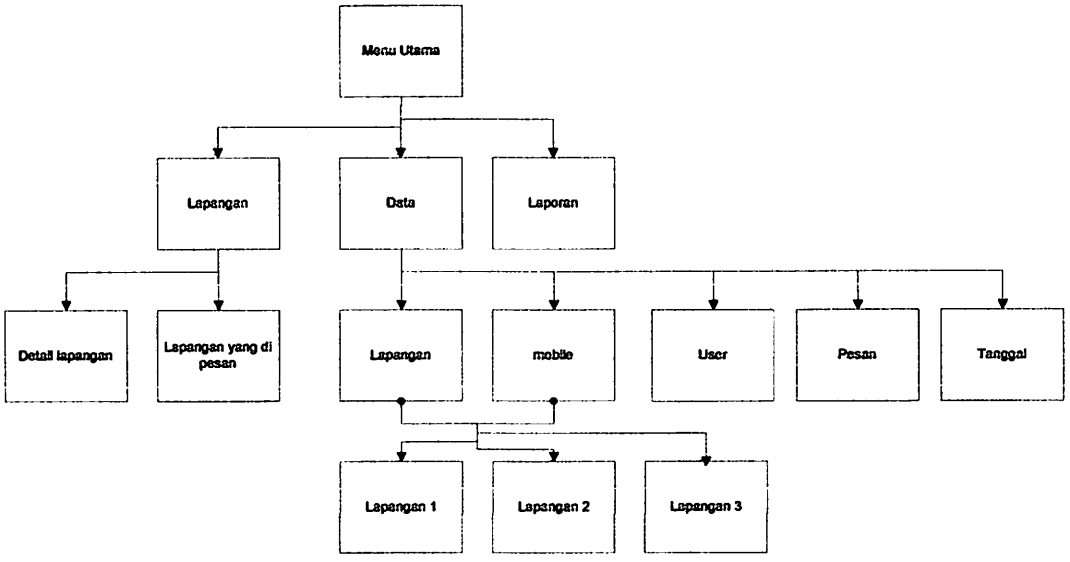
Dalam melakukan komunikasi dengan *server*, aplikasi pemesanan melakukan *request* pada *server* dan *server* memberikan *reply* pada aplikasi, *request* yang dilakukan oleh aplikasi pemesanan lapangan futsal diolah terlebih dahulu oleh *script* PHP dan dilakukan koneksi ke *database*. Adapun flowchart PHP disisi *server* adalah sebagai berikut :



Gambar 3.10. Flowchart Server

3.2.5 DESAIN APLIKASI DI SERVER

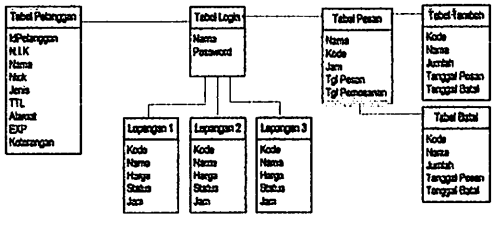
Dalam aplikasi di *server* ini nantinya pemesan dapat mengetahui berbagai informasi berhubungan dengan datalapangan yang akan dipesan oleh user, dibatalkan dan memasukkan data – data pemesanan Lapangan untuk *user*. Aplikasi di *server* mengunakan PHP dan MySQL. Aplikasi berbasis *web* ini untuk pelayan selanjutnya kepada konsumen, dengan mengetahui tanggal pesan ataupun Lapangan yang dipesan memudahkan pelayan untuk meneruskan data Lapangan kepada konsumen. Berikut merupakan desain konten dari aplikasi di *server*



Gambar 3.11. Diagram Konten

3.2.6 DESAIN DATABASE

Dalam mengolah *request* dari aplikasi pemesanan lapangan futsal, *server* melakukan komunikasi dengan *database* untuk menyimpan data dan mengambil data dari desain konten yang ada maka dibutuhkan beberapa database untuk memenuhi kebutuhan dari desain kontent dan transaksi pemesanan lapangan futsal.



Gambar 3.12. Database

Tabel3.1. Tabel Login

No.	Nama Kolom	Tipe Data	Keterangan
1.	Nama	Varchar(30)	*Primary Key (PK)*, Not null
2.	Password	Varchar(20)	Not null

Tabel3.2. Tabel Pesan

No.	Nama Kolom	Tipe Data	Keterangan
1.	Nama	Varchar(30)	*Primary Key (PK)*, Not null
2.	Kode	BigInt(10)	Not null
3.	Jumlah jam	Varchar(15)	Not null
4.	Tgl pesan	Date(0)	Not null
5.	Tanggal Pesan	timestamp(14)	Not null

Tabel3.3. Tabel Lapangan 1

No.	Nama Kolom	Tipe Data	Keterangan
1.	Kode	BigInt(10)	*Primary Key (PK)*, Not null
2.	Nama	Varchar(30)	Not null
3.	Harga	Int (10)	Not null
4.	Status	Varchar(30)	Not null
5.	Jam	Varchar(50)	Not null

Tabel3.4. Tabel Lapangan 2

No.	Nama Kolom	Tipe Data	Keterangan
1.	Kode	BigInt(10)	*Primary Key (PK)*, Not null
2.	Nama	Varchar(30)	Not null
3.	Harga	Int (10)	Not null
4.	Status	Varchar(30)	Not null
5.	Jam	Varchar(50)	Not null

Tabel3.5. Tabel Lapangan 3

No.	Nama Kolom	Tipe Data	Keterangan
1.	Kode	Bigint(10)	*Primary Key (PK)*, Not null
2.	Nama	Varchar(30)	Not null
3.	Harga	Int(10)	Not null
4.	Status	Varchar(30)	Not null
5.	Jam	Varchar(50)	Not null

Tabel3.6. Tabel Tanggal

No.	Nama Kolom	Tipe Data	Keterangan
1.	tanggal	Date(0)	*Primary Key (PK)*, Not null

Tabel3.7. Tabel Pelanggan

No.	Nama Kolom	Tipe Data	Keterangan
1.	IdPelanggan	Bigint(10)	*Primary Key (PK)*, Not null
2.	N.I.K	Varchar(30)	Not null
3.	Nama	Varchar(30)	Not null
4.	Nick	Varchar(30)	Not null
5.	Jenis	Varchar(30)	Not null
6.	TTL	Varchar(30)	Not null
7.	Alamat	Varchar(30)	Not null
8.	EXP	Varchar(30)	Not null
9.	Keterangan	Vardhar(30)	Not null

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Tahap implementasi desain perangkat lunak merupakan proses pengubahan spesifikasi sistem menjadi sistem yang dapat dijalankan. Tahap ini merupakan lanjutan dari proses perancangan, yaitu proses pemrograman perangkat lunak sesuai dengan spesifikasi dan desain sistem.

Aplikasi pemesanan lapangan futsal berbasis telepon seluler ini menggunakan pemograman J2ME dan menggunakan basis data MySQL yang berfungsi sebagai media penyimpanan data atau informasi yang terkumpul, yang terdiri dari beberapa tabel yang saling berhubungan. Sedangkan untuk menjembatani antara informasi yang akan dibuat dengan basis data yang ada, digunakan PHP.

4.1.1 Login

Login merupakan awal dari transaksi pemesanan lapangan futsal agar antara pemesan dan jasa pemesanan sama – sama mengetahui dan ahirnya timbul kepercayaan dalam melakukan transaksi selanjutnya. Aplikasi pemesanan lapangan futsal ini dalam menerapkan *login* tidak melalui media penyimpan *persisten* (RMS) dalam telepon seluler melainkan data berada pada *server* sehinggan memerlukan koneksi GPRS sebelum mengirim data berupa nama dan *password* yang telah dibuat sebelumnya. Berikut tampilan Login:

Gambar 4.1. form login

Gambar dibawah merupakan *Script* PHP yang memproses *Login* dengan membandingkan data yang dikirimkan oleh *user* berupa *variabel* n dan *variabel* m dengan data yang berada pada *database* MySQL.

```php
<?php
include "koneksi.php";

mysql_connect("localhost","root","triadpass");
mysql_select_db("data");

//=============================================
// Bagian ini untuk menangani proses Login
// 1. Periksa apakah nama telah pernah terdaftar
// 2. Periksa apakah nama telah pernah terdaftar
// 3. Jika Nama dan Password terdaftar maka akan mengirim 2
// 4. Jika Nama dan Password tidak terdaftar maka akan mengirim 1
//=============================================
if ($a=="a"){
    $sql = "SELECT * FROM tbllogin Where nama = '$n' AND password='$m'";
    $result = mysql_query($sql);
    $row= mysql_fetch_array($result);
    if($row[nama]==$n AND $row[password]==$m){

                echo "2";}

     else {

                 echo "1";
    }
}
?>
```

Gambar 4.2. pemogram login PHP pada server

Gambar dibawah ini merupakan *script* java yang berfungsi untuk memeriksa nama dan *password* sebelum dikirim ke *server* agar nantinya koneksi benar – benar efisien karena menggunakan GPRS yang berbayar sehingga waktu

pemrosesan pun perlu diperhitungkan. Jika Nama dan *Password* kosong maka aplikasi akan membuka *form* Konfirmasi sehingga tidak melakukan koneksi ke *server*, namun jika Nama dan *Password* tidak kosong akan dikirimkan ke *server* dan akan diperiksa oleh *script* PHP diatas.

```
String wrn="";
if(txtNama_1.size()==0){
    wrn="Nama Tidak Boleh Kosong\n";}
if (txtPass_1.size()==0){
  wrn += "Password Tidak Boleh Kosong\n";
}


if (!txtPass_1.getString().equals(txtPass_1.getString())){
 wrn += "Password must be equal\n ";
}


if (!wrn.equals("")){
  getDisplay().setCurrent(getFrmKonv_B1());
  strKon_B1.setText(wrn);
}
if(txtNama_1.size()>0){
  if(txtPass_1.size()>0){
   getDisplay().setCurrent(getFrmKonv_B2());}

 }
```

Gambar 4.3. pemograman login java

4.2 Pengujian Hasil

Pengujian dilakukan menggunakan *emulator* SonyEricsson dengan versi w200i yang sudah mendukung java. Pengujian nantinya meliputi proses melihat lapangan, pemesanan dan pembatalan. Berikut merupakan gambar dari menu utama aplikasi pemesanan lapangan futsal:

Gambar 4.4. form menu utama

4.2.1 Pengujian Lapangan

Menu Lapangan merupakan menu yang berfungsi sebagai media untuk *user* dalam hal ini pelanggan yang hanya ingin melihat – lihat lapangan yang ditawarkan tanpa melakukan transaksi pemesanan maupun pembatalan. Sehingga konsumen dapat memiliki keleluasaan dalam menentukan pilihan pemesanan dalam transaksi pemesanan lapangan futsal.

Pengujian dilakukan dengan menekan tombol *next* atau *back* untuk melihat daftar lapangan. Jika tombol *next* ditekan maka *record* selanjutnya akan terbuka dan jika tombol *back* ditekan tombol sebelumnya akan tampil sehingga memudahkan pengguna dalam menggunakanya. Gambar berikut merupakan menu lapangan:

Gambar 4.5. form lihat lapangan

4.2.2 Pengujian Pemesanan

Dalam melakukan transaksi pemesanan user harus melakukan login terlebih dahulu selanjutnya dapat melakukan transaksi pemesanan . Ada beberapa proses pengujian dalam menu pemesanan berupa *login*, proses melihat lapangan , proses pemesanan dengan memasukkan kode , jumlah jam ,tanggal yang akan dipesan dan *report* yang menandakan *server* telah menerima transaksi oleh *user*.

4.2.2.1 Pengujian Login

Pengujian login ini dilakukan dengan memasukkan nama dan *password* yang diketik oleh user dalam *textField* lalu dikirimkan ke *server* untuk mengecek keabsahan user dalam melakukan transaksi selanjuntya. Aplikasi pemesanan lapangan futsal ini akan menolak mengirimkan data ke *server* bila nama dan *password* kosong, dan apabila nama dan *password* tidak kosong maka akan mengirim data ke *server*. Dan disini *server* akan membandingkan data yang dikirimkan dengan *database* jika salah maka akan memanggil *from* konfirmasi kesalahan namun jika data benar maka dapat melakukan transaksi selanjutnya.

Gambar 4.6. form login pada aplikasi

Gambar 4.7. konfirmasi kesalahan menulis nama dan password

Gambar 4.8. persetujuan mengirim data dan menerima dari dan ke server

Gambar 4.9. nama dan password telah divalidasi oleh server

4.2.2.2 Pengujian Pemilihan Lapangan

Setelah *login* sukses maka *server* akan menunggu transaksi selanjutnya dari *user*. Dalam aplikasi pemesanan lapangan futsal ini setelah *user* sukses melakukan *login* maka akan masuk ke menu lapangan. Di menu lapangan ini berbeda dengan menu *main* awal aplikasi disini *user* dapat memilih untuk melihat lapangan sesuai kriteria yang disediakan oleh jasa pemesanan sehingga dalam melakukan proses pemesanan user lebih mudah dalam melakukan transaksi pemesanan selanjutnya. Berikut merupakan menu pilihan lapangan yang disediakan oleh jasa pemesanan :

Gambar 4.10. pemilihan lapangan yang dipesan

Gambar 4.11. lihat lapangan pada transaki pemesanan

4.2.2.3 Pengujian Pemesanan

Setelah memilih *user* dapat melihat kembali lapangan beserta kode dan harga dan selanjutnya dapat melakukan transaksi pemesanan dengan memasukkan kode sesuai kode pada lapangan yang ada pada *form* sebelumnya. Dan jika *user* lupa atau kesulitan dalam menghafal kode, user dapat menekan tombol *back* untuk melihat kembali kode yang terdapat pada lapangan tersebut.

Gambar 4.12. form pemesanan

Setelah menekan Ok berarti *user* setuju dan aplikasi akan mengirimkan kode ke *server* dan jika *server* menerima kode tersebut *server* akan mengirimkan *report* bahwa transaksi diterima oleh *server*.Berikut gambar yang menyatakan kode telah diterima oleh *server* :

Gambar 4.13. form report

Gambar 4.14. form jika salah memasukkan kode

Gambar 4.15. form jika Lapangan di pesan

4.2.3 Pengujian Pembatalan

Pada menu pembatalan ini awalnya user harus *login* terlebih dahulu agar identitasnya diketahui. Setelah *user* berhasil *login* maka akan tampil *menu* hasil pemesanan yang telah dilakukan oleh *user* sebelumnya. Disini *user* mengetahui kode hasil transaksi sebelumnya. Jika ingin membatalkan dapat menginputkan data yang ingin di batalkan pada menu pembatalan ini sehingga memudahkan pengguna.

4.2.3.1 Pengujian Login Pembatalan

Pengujian login ini dilakukan dengan memasukkan nama dan *password* yang diketik oleh *user* dalam *textField* lalu dikirimkan ke *server* untuk mengecek keabsahan *user* dalam melakukan transaksi selanjuntya. Aplikasi pemesanan lapangan futsal ini akan menolak mengirimkan data ke *server* bila nama dan *password* kosong, dan apabila nama dan *password* tidak kosong maka akan mengirim data ke *server*. Dan disini *server* akan membandingkan data yang dikirimkan dengan *database* jika salah maka akan memanggil *from* konfirmasi kesalahan namun jika data benar maka dapat melakukan transaksi selanjutnya.

Gambar 4.16. login pada pembatalan

4.2.3.2 Pengujian Penampilan Lapangan Dipesan

Pada pengujian ini akan dimunculkan lapangan yang sudah dipesan. *User* akan terhubung ke *database server* dan akan menerima kode – kode yang sudah dikirimkan dan akan diproses oleh *server* berikut merupakan gambar hasil penampilan transaksi yang telah dilakukan oleh *user*.

Gambar 4.17. form lapangan yang sudah dipesan

4.2.3.3 Pengujian Pembatalan

User dapat melakukan pembatalan transaksi pemesanan dengan mengisi data pada *form* pembatalan berikut, Dengan ini *user* akan dimudahkan jika terjadi kekeliruan dalam memasukkan kode dalam transaksi pemesanan. Berikut merupakan tampilan *form* pembatalan.

Gambar 4.18. form pembatalan

Dari gambar di atas dapat diketahui bahwa *user* dapat mengisi data dimana kode dan tanggal pesan yang ingin di batalkan dan data pemesanan akan dihapus oleh server.

4.2.3.4 Pengujian Report Pembatalan

Setelah form pembatalan telah dilakukan maka *user* akan mengirim data tersebut dan *server* akan membatalkan atau men*delete* berdasarkan nama *login,* kode lapangan dan tanggal pemesanan , sehingga *server* menerima kode lapangan dan tanggal berdasarkan dengan nama *login* yang telah dimasukkan oleh *user.* Berikut merupakan gambar *report* data yang telah diterima oleh *server* :

Gambar 4.19. konfirmasi server pembatalan sukses

4.2.4 Pengujian Menu Info

Menu info ini memuat cara – cara dalam melakukan transaksi pemesanan dan pemesanan lapangan futsal. Dalam menu ini juga terdapat pemberitahuan nama pembuat dan tujuan pembuat dalam medesain aplikasi pemesanan lapangan futsal ini. Berikut merupakan gambar menu info.

Gambar 4.20. menu info

4.2.5 Pengujian Login Aplikasi Web

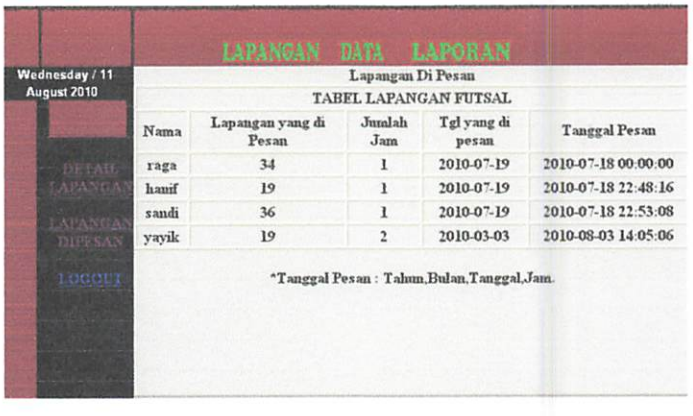
Sebelum masuk ke aplikasi *web* untuk mengetahui informasi yang ada berhubungan dengan konsumen dalam hal ini transaksi yang dilakukan pelanggan dalam pemesanan dan pembatalan dilakukan *login* terlebih dahulu. Berikut merupakan gambar menu Login.

Gambar 4.21. menu login

Gambar 4.22. konfirmasi login

4.2.6 Pengujian Menu Lapangan

Pada menu tampilan awal pada menu lapangan terdapat informasi mengenai lapangan yang dipesan.Pada menu lapangan ini juga terdapat menu detail lapangan dan menu lapangan dipesan. Detail lapangan digunakan untuk mengetahui Lapangan yang telah di pesan atau belum pada hari ini, ini akan memudahkan pelayan mengetahui lebih awal lapangan mana yang kosong dan lapangan mana yang telah di pesan . Lapangan dipesan digunakan untuk mengetahui Lapangan yang dipesan oleh konsumen saat ini, . Berikut merupakan gambar menu lapangan.

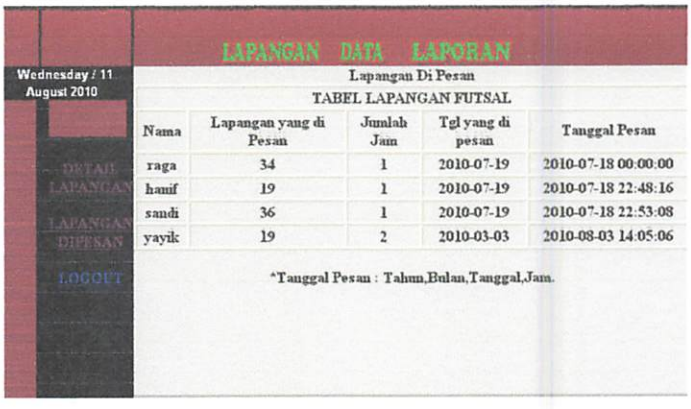


Gambar 4.23. menu lapangan

Pada menu detail dapat diketahui jika Lapangan telah di pesan maka pelayan harus mengupdate Lapangan yang telah di pesan dengan memasukan nama pemesan karena jika lapangan telah dipesan maka konsumen tidak akan memesan lapangan yang telah dipesan dan dapat melakukan transaksi pemesanan yang lain .

Gambar 4.24. detail lapangan

Pada menu lapangan dipesan ini dapat diketahui siapa yang melakukan transaksi pemesanan .



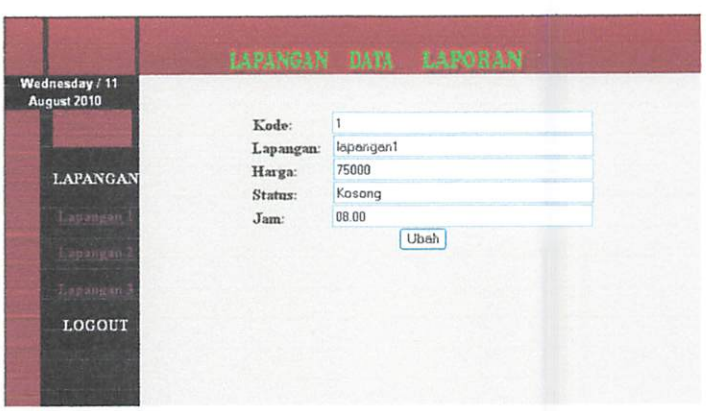
Gambar 4.25. lapangan dipesan

4.2.7 Pengujian Menu Data

Pada menu data ini pelayan dapat mengupdate data lapangan yang telah dipesan maupun belum karena dalam aplikasi di *handphone* jika lapangan telah dipesan maka *user* tidak dapat melakukan transaksi pemesanan pada lapangan yang telah dipesan. Berikut merupakan gambar menu Data.

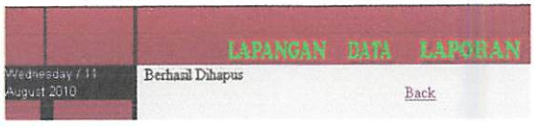
Gambar 4.26. data edit

Dengan menekan kolom *edit* maka akan pelayan dapat melakukan *edit* pada semua *field* pada data baru.



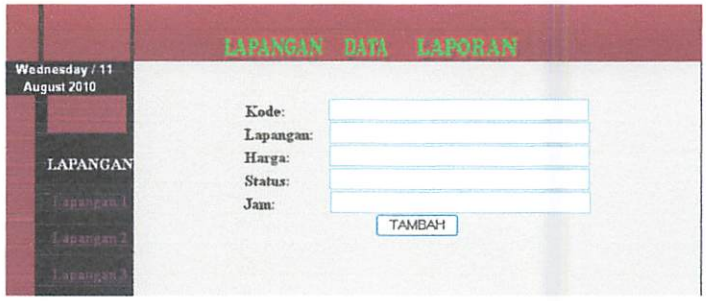
Gambar 4.27. form edit

Dengan menekan kolom hapus pelayan dapat menghapus data yang ingin di hapus dan data yang berhasil di hapus akan terdapat pemberitahuan "*berhasil dihapus*"



Gambar 4.28. konfirmasi data dihapus

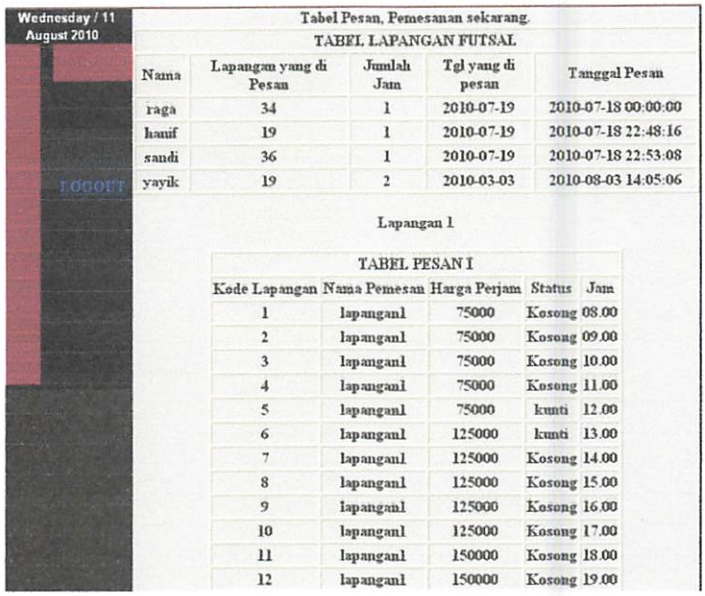
Form tambah ini digunakan untuk menambah lapangan pada data baru dengan memasukkan kode, lapangan, harga, status dan *jam.*



Gambar 4.29. form tambah

4.2.8 Pengujian Laporan

Pada menu laporan ini dapat diketahui melalui tabel tentang transaksi yang telah dipesan, Berikut merupakan gambar laporan.



Gambar 4.30. laporan

BAB V

PENUTUP

Berdasarkan pada hasil pengujian dan analisa terhadap hasil yang didapatkan, maka dapat diambil suatu kesimpulan dan saran untuk kemungkinan pengembangan sistem yaitu :

5.1. Kesimpulan

- Proses permintaan dan penerimaan pemesanan lapangan futsal menjadi lebih cepat dengan adanya info lapangan dan data pemesanan yang dihasilkan juga sesuai dengan keinginan konsumen.
- Dengan adanya aplikasi pemesanan lapangan futsal ini maka dapat memudahkan konsumen dalam melakukan transaksi pemesanan lapangan futsal .
- Konsumen dapat mengetahui lapangan futsal yang telah di pesan maupun belum pada hari tersebut

5.2. Saran

Untuk mengembangkan aplikasi ini dapat dilakukan dengan beberapa cara, diantaranya:

- Aplikasi ini hanya menampilkan transaksi pemesanan waktu itu saja,sehingga bisa dikembangkan dengan menampilkan transaksi transaksi pemesanan yang di lakukan di hari hari berikutnya
- Untuk mempercantik tampilan dapat menggunakan *library* lain, seperti J2ME *Polish*, LWUIT(*Light Weight User Interface Toolkit*), *Synclast, Thinlet,* dan *J4ME.*

DAFTAR PUSTAKA

1. **Dwi Prasetyo, Didik.** *150 Rahasia Pemrograman Java.* Jakarta 2007. PT. Elex Media Koputindo

2. **Budi, Imam, dan Arif.** *Tuntunan Pemrograman Java untuk Handphone.* Bandung 2007. PT. Informatika

3. **Michael Siregare, Ivan.** *Membangun Aplikasi Chat Lewat GPRS dengan J2ME Menggunakan NetBeans IDE 5.0.* Yogyakarta 2007. PT. Gava Media

4. **Riyanto, Suprapto, dan Hendi.** *Pengembangan Aplikasi Manajemen Database dengan Java 2.* Yogyakarta 2008. PT. Gava Media

5. **Firdaus.** *PHP & MySQL dengan Dreamweaver.* Palembang 2007. PT. Penerbit Maxikom

6. **Madcoms.** *Aplikasi Web Database Menggunakan Adobe Dreamweaver CS3 & Pemrograman PHP+Mysql.* Madiun 2008. PT Penerbit Andi

7. **M.Salahudin, dan Rosa A.S.** *Pemrograman J2ME Belajar Cepat Pemrograman Perangkat Telekomunikasi Mobile.* Bandung 2008. PT Infotmatika

LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAM TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
I (PERSERO) MALANG
NK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

AMA	:	**GEMA MARDHIKA WICAKSONO**
IM	:	**05.12.732**
URUSAN	:	**TEKNIK ELEKTRO S-1**
ONSENTRASI	:	**TEKNIK KOMPUTER DAN INFORMATIKA**
UDUL SKRIPSI	:	**PENGEMBANGAN APLIKASI SISTEM INFORMASI PEMESANAN LAPANGAN FUTSAL**

Dipertahankan di hadapan penguji skripsi jenjang program strata satu (S-1) pada :

lari	:	**Senin**
anggal	:	**16 Agustus 2010**
engan nilai	:	**84,5 (A)**

PANITIA UJIAN SKRIPSI

Mengetahui,
Ketua Majelis Penguji

Ir. Yusuf Ismail Nakhoda, MT
NIP. Y. 1018800189

ANGGOTA PENGUJI

Dosen Penguji I

Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP. P. 1030800417

Dosen Penguji II

Sotyohadi, ST
NIP. Y. 1039700309

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo 10 KM
MALANG

FORMULIR BIMBINGAN SKRIPSI

: Gema Mardhika Wicaksono

: 05.12.732

Bimbingan : 8 Desember 2009 – 8 Juni 2010

Skripsi : PENGEMBANGAN APLIKASI SISTEM INFORMASI PEMESANAN LAPANGAN FUTSAL

Tanggal	Uraian	Paraf Pembimbing
21 - 6 - 2010	BAB I, II Revisi	
24 - 6 - 2010	BAB I, II ACC	
12 - 7 - 2010	BAB III Revisi	
21 - 7 - 2010	BAB III ACC	
26 - 7 - 2010	BAB IV, V Revisi	
2 - 8 - 2010	BAB IV, V ACC	
03 - 8 - 10	ACC KOMPRE	

Malang,
Dosen Pembimbing

(JOSEPH DEDY IRAWAN ST, MT)
NIP. Y 1323125178

FORM S-4b

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package Mobile;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
import com.wingfoot.soap.*;
import com.wingfoot.soap.encoding.*;
import com.wingfoot.soap.transport.*;


import org.netbeans.microedition.lcdui.SplashScreen;
//import org.netbeans.microedition.lcdui.TableItem;

/**
 * @author gesa
 */
public class PMobile extends MIDlet implements CommandListener {

    private boolean midletPaused = false;

    //<editor-fold defaultstate="collapsed" desc=" Generated Fields ">
    private SplashScreen splashScreen;
    private Form frmA_1;
    private ChoiceGroup chcA_3;
    private Form frmP_1;
    private StringItem stringItem1;
    private Form frmBt_1;
    private TextField txtNama_2;
    private TextField txtPass_2;
    private StringItem strBt_1;
    private Form frmB_1;
    private TextField txtNama_1;
    private TextField txtPass_1;
    private StringItem strB_1;
    private Form frmIn_1;
    private StringItem stringItem2;
    private Form frmKonv_B2;
    private StringItem str_B2;
    private Form frmKonv_B1;
    private StringItem strKon_B1;
    private Form frmKonv_B3;
    private Form frmPesan_B5;
    private TextField txtLapangan_B5;
    private TextField txtLapangan1_B5;
    private TextField txtLapangan2_B5;
    private Form frmLihat_B4;
    private Form frmKonv_B6;
    private Form frmKonv_Bt2;
    private StringItem stringItem;
    private Form frmKonv_Bt1;
    private StringItem strBt_2;
    private Form frmList_Bt4;
    private Form frmKonv_Bt3;
    private Form frmBatal_Bt5;
    private StringItem stringItem3;
    private TextField txtB_1;
    private TextField txtB_2;
    private Form frmKonv_Bt6;
    private Form frm_B7;
    private StringItem str_B7;
    private ChoiceGroup chc_B7;
    private Form frmPb;
    private Form frmBPB;
    private StringItem stringItem4;
    private Form frmBMK;
    private TextField txtBMK;
    private TextField txtBMK1;
    private TextField txtBMK2;
    private Form frmBSM;
    private TextField txtLapanganBSM1;
    private TextField txtlapanganBSM;
    private TextField txtLapanganBSM2;
    private Form formMk;
    private Form frmSM;
    private Form frmBSM1;
    private Form frmBMK1;
    private Form frmP_2;
    private StringItem stringItem5;
    private Form frmP_3;
    private StringItem stringItem6;
    private Command cmdA_4;
    private Command cmdBack_B;
    private Command cmdOk_B;
    private Command cmdBack_Bt;
    private Command cmdOk_Bt;
    private Command cmdOk_In;
    private Command cmdOk_P;
    private Command cmdBack_In;
    private Command cmdBack_P;
    private Command cmdBack_B1;
    private Command cmdOk_B3;
    private Command cmdOk_B2;
    private Command cmdBack_B4;
    private Command cmdOk_B4;
    private Command cmdOk_B6;
    private Command cmdOk_B5;
    private Command cmdBack_B5;
    private Command cmdBack_Bt1;
    private Command cmdExit_B6;
    private Command cmdOk_Bt4;
    private Command cmdOk_Bt3;
    private Command cmdOk_Bt2;
    private Command cmdBack_Bt5;
    private Command cmdOk_Bt5;
    private Command cmdExit_Bt6;
    private Command cmdBack_Bt6;
    private Command cmdOk_B7;
    private Command cmdNext;
    private Command cmdBack_P1;
    private Command okCommand;
    private Command cmdOkPB;
    private Command okCommand1;
    private Command okCommand2;
    private Command okCommand3;
    private Command cmdBackBSM;
    private Command cmdOkBSM;
    private Command okCommand4;
    private Command backCommand;
    private Command cmdOkBMK;
    private Command cmdBackBMK;
    private Command cmdBack_P2;
    private Command cmdNext3;
    private Command cmdNext2;
    private Command okCommand5;
    private Command cmdBack_P3;
    private Image image1;
    private Image image2;
    private Image image3;
    private Image image4;
    private Image image5;
    private Image image6;
    private Image image7;
    private Image image8;
    //</editor-fold>
     public String nama1="";
     public String pass1;
     public String nama2="";
     public String pass2;
     public String lapangan="";
     public String lapangan1="";
     public String lapangan2="";
     public String lapangan3="";
     public String lapangan4="";
     public String lapangan5="";
     public String lapangan6="";
     public String lapangan7="";
     public String lapangan8="";
     public String lapangan9="";
     public String lapangan10="";
     public String lapangan11="";
     public String lapangan12="";
     public String lapangan13="";
     public String lapangan14="";
     private Object[] record;
          private int curRec;
     private boolean buttonShowed;
    /**
     * The PMobile constructor.
     */
    public PMobile() {
    }

    //<editor-fold defaultstate="collapsed" desc=" Generated Methods
">
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
initialize ">
    /**
     * Initilizes the application.
     * It is called only once when the MIDlet is started. The method
is called before the <code>startMIDlet</code> method.
     */
    private void initialize() {
        // write pre-initialize user code here

        // write post-initialize user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
startMIDlet ">
    /**
     * Performs an action assigned to the Mobile Device - MIDlet
Started point.
     */
    public void startMIDlet() {
        // write pre-action user code here
        switchDisplayable(null, getSplashScreen());
        // write post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
resumeMIDlet ">
    /**
     * Performs an action assigned to the Mobile Device - MIDlet
Resumed point.
     */
    public void resumeMIDlet() {
        // write pre-action user code here

        // write post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
switchDisplayable ">
    /**
     * Switches a current displayable in a display. The
<code>display</code> instance is taken from <code>getDisplay</code>
```

```
method. This method is used by all actions in the design for switching
displayable.
     * @param alert the Alert which is temporarily set to the display;
if <code>null</code>, then <code>nextDisplayable</code> is set
immediately
     * @param nextDisplayable the Displayable to be set
     */
    public void switchDisplayable(Alert alert, Displayable
nextDisplayable) {
        // write pre-switch user code here
        Display display = getDisplay();
        if (alert == null) {
            display.setCurrent(nextDisplayable);
        } else {
            display.setCurrent(alert, nextDisplayable);
        }
        // write post-switch user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
commandAction for Displayables ">
    /**
     * Called by a system to indicated that a command has been invoked
on a particular displayable.
     * @param command the Command that was invoked
     * @param displayable the Displayable where the command was
invoked
     */
    public void commandAction(Command command, Displayable
displayable) {
        // write pre-action user code here
        if (displayable == formMk) {
            if (command == okCommand1) {
                // write pre-action user code here
                switchDisplayable(null, getFrmBMK());
                // write post-action user code here
            }
        } else if (displayable == frmA_1) {
            if (command == cmdA_4) {
              switch(chcA_3.getSelectedIndex()){    // write pre-
action user code here

                  case 0 :

                      getDisplay().setCurrent(getFrmP_1());

                       break;
                  case 1 :

                      getDisplay().setCurrent(getFrmP_2());

                       break;
                  case 2 :

                      getDisplay().setCurrent(getFrmP_3());

                       break;

                  case 3 :
                      getDisplay().setCurrent(getFrmB_1());
                      break;
                  case 4 :
                      getDisplay().setCurrent(getFrmBt_1());
                      break;
                  case 5 :
                      getDisplay().setCurrent(getFrmIn_1());
                      break;
                  case 6 :
                      exitMIDlet();
                      break;
              }
                  // write post-action user code here
            }
        } else if (displayable == frmBMK) {
            if (command == cmdBackBMK) {
                // write pre-action user code here
                switchDisplayable(null, getFormMk());
                // write post-action user code here
            } else if (command == cmdOkBMK) {
                 lapangan10 = txtBMK .getString();
                 lapangan11 = txtBMK1.getString();
                 lapangan12 = txtBMK2.getString();

                 PesanLapangan2 reg2 = new PesanLapangan2();
                 reg2.setListener(this);
                 reg2.start();  // write pre-action user code here
                 switchDisplayable(null, getFrmBMK1());
                // write post-action user code here
            }
        } else if (displayable == frmBMK1) {
            if (command == okCommand4) {
                // write pre-action user code here
                switchDisplayable(null, getFrmA_1());
                // write post-action user code here
            }
        } else if (displayable == frmBP8) {
            if (command == backCommand) {
                // write pre-action user code here
                switchDisplayable(null, getFrm_87());
                // write post-action user code here
            }
        } else if (displayable == frmBSM) {
            if (command == cmdBackBSM) {
                // write pre-action user code here
                switchDisplayable(null, getFrmSM());
                // write post-action user code here
            } else if (command == cmdOkBSM) {
                 lapangan5 = txtlapanganBSM .getString();
                 lapangan6 = txtLapanganBSM1.getString();
                 lapangan7 = txtLapanganBSM2.getString();

                 PesanLapangan1 reg2 = new PesanLapangan1();
                 reg2.setListener(this);
                 reg2.start();// write pre-action user code here
                 switchDisplayable(null, getFrmBSM1());
                // write post-action user code here
            }
        } else if (displayable == frmBSM1) {
            if (command == okCommand3) {
                // write pre-action user code here
                switchDisplayable(null, getFrmA_1());
                // write post-action user code here
            }
        } else if (displayable == frmB_1) {
            if (command == cmdBack_B) {
                // write pre-action user code here
                switchDisplayable(null, getFrmA_1());
                // write post-action user code here
            } else if (command == cmdOk_B) {
                 String wrn="";
                if(txtNama_1.size()==0){
                    wrn="Nama Tidak Boleh Kosong\n";}
                 if (txtPass_1.size()==0){
                   wrn += "Password Tidak Boleh Kosong\n";
                 }


                if
(!txtPass_1.getString().equals(txtPass_1.getString())){
                   wrn += "Password must be equal\n ";
                }


                if (!wrn.equals("")){
                   getDisplay().setCurrent(getFrmKonv_B1());
                   strKon_B1.setText(wrn);
                }
                if(txtNama_1.size()>0){
                   if(txtPass_1.size()>0){
                    getDisplay().setCurrent(getFrmKonv_B2());}

                  }   // write pre-action user code here

                // write post-action user code here
            }
        } else if (displayable == frmBatal_Bt5) {
            if (command == cmdBack_Bt5) {
                // write pre-action user code here

                // write post-action user code here
            } else if (command == cmdOk_Bt5) {
                 lapangan = txtB_1.getString();
                 lapangan1 = txtB_2.getString();
                // lapangan2 = txtB_3.getString();
                 BatalPemesanan reg2 = new BatalPemesanan();
                 reg2.setListener(this);
                 reg2.start();  // write pre-action user code here
                 switchDisplayable(null, getFrmKonv_Bt6());
                // write post-action user code here
            }
        } else if (displayable == frmPesan_B5) {
            if (command == cmdBack_B5) {
                // write pre-action user code here
                switchDisplayable(null, getFrmLihat_B4());
                // write post-action user code here
            } else if (command == cmdOk_B5) {
                 lapangan = txtLapangan_B5.getString();
                 lapangan1 = txtLapangan1_B5.getString();
                 lapangan2 = txtLapangan2_B5.getString();



                 PesanLapangan reg2 = new PesanLapangan();
                 reg2.setListener
                         (this);
                 reg2.start();  // write pre-action user code here
                 switchDisplayable(null, getFrmKonv_B6());
                // write post-action user code here
            }
        } else if (displayable == frmBt_1) {
            if (command == cmdBack_Bt) {
                // write pre-action user code here
                switchDisplayable(null, getFrmA_1());
                // write post-action user code here
            } else if (command == cmdOk_Bt) {
             String wrn="";
                if(txtNama_2.size()==0){
                    wrn="\nNama Tidak Boleh Kosong\n";}
                 if (txtPass_2.size()==0){
                   wrn += "Password tidak boleh Kosong\n";
                 }


                if
(!txtPass_2.getString().equals(txtPass_2.getString())){
                   wrn += "Password must be equal\n ";
                }


                if (!wrn.equals("")){
                   getDisplay().setCurrent(getFrmKonv_Bt1());
                   strBt_2.setText(wrn); // write post-action user
code here
                }
                if(txtNama_2.size()>0){
                   if(txtPass_2.size()>0){
                    getDisplay().setCurrent(getFrmKonv_Bt2());}
```

```
            }    // write pre-action user code here

            // write post-action user code here
        }
    } else if (displayable == frmIn_1) {
        if (command == cmdBack_In) {
            // write pre-action user code here
            switchDisplayable(null, getFrmA_1());
            // write post-action user code here
        }
    } else if (displayable == frmKonv_B1) {
        if (command == cmdBack_B1) {
            // write pre-action user code here
            switchDisplayable(null, getFrmB_1());
            // write post-action user code here
        }
    } else if (displayable == frmKonv_B2) {
        if (command == cmdOk_B2) {
            nama1 = txtNama_1.getString();
            pass1 = txtPass_1.getString();
            login1 reg = new login1();
            reg.setListener(this);
            reg.start(); // write pre-action user code here

            // write post-action user code here
        }
    } else if (displayable == frmKonv_B3) {
        if (command == cmdOk_B3) {
            // write pre-action user code here
            switchDisplayable(null, getFrm_B7());
            // write post-action user code here
        }
    } else if (displayable == frmKonv_B6) {
        if (command == cmdExit_B6) {
            // write pre-action user code here
            switchDisplayable(null, getFrmA_1());
            // write post-action user code here
        } else if (command == cmdOk_B6) {
            // write pre-action user code here
            switchDisplayable(null, getFrmA_1());
            // write post-action user code here
        }
    } else if (displayable == frmKonv_Bt1) {
        if (command == cmdBack_Bt1) {
            // write pre-action user code here
            switchDisplayable(null, getFrmBt_1());
            // write post-action user code here
        }
    } else if (displayable == frmKonv_Bt2) {
        if (command == cmdOk_Bt2) {
            nama2 = txtNama_2.getString();
            pass2 = txtPass_2.getString();
            login2 reg = new login2();
            reg.setListener(this);
            reg.start();   // write pre-action user code here
            switchDisplayable(null, getFrmKonv_Bt3());
            // write post-action user code here
        }
    } else if (displayable == frmKonv_Bt3) {
        if (command == cmdOk_Bt3) {
            nama2 = txtNama_2.getString();
            ListPesan reg = new ListPesan();
            reg.setListener(this);
            reg.start(); // write pre-action user code here
            switchDisplayable(null, getFrmList_Bt4());
            // write post-action user code here
        }
    } else if (displayable == frmKonv_Bt6) {
        if (command == cmdBack_Bt6) {
            // write pre-action user code here
            switchDisplayable(null, getFrmA_1());
            // write post-action user code here
        } else if (command == cmdExit_Bt6) {
            // write pre-action user code here
            switchDisplayable(null, getFrmA_1());
            // write post-action user code here
        }
    } else if (displayable == frmLihat_B4) {
        if (command == cmdBack_B4) {
            // write pre-action user code here
            switchDisplayable(null, getFrmLihat_B4());
            // write post-action user code here
        } else if (command == cmdOk_B4) {
            // write pre-action user code here
            switchDisplayable(null, getFrmPesan_BS());
            // write post-action user code here
        }
    } else if (displayable == frmList_Bt4) {
        if (command == cmdOk_Bt4) {
            // write pre-action user code here
            switchDisplayable(null, getFrmBatal_Bt5());
            // write post-action user code here
        }
    } else if (displayable == frmP_1) {
        if (command == cmdBack_P) {

            // write pre-action user code here
            switchDisplayable(null, getFrmA_1());
            // write post-action user code here
        } else if (command == cmdBack_P1) {
            curRec--;
:tion user code here                    viewRecord(); // write pre-

            // write post-action user code here
        } else if (command == cmdNext) {
            curRec++;
                                        viewRecord();
```

```
                // write pre-action user code here

                // write post-action user code here
            }
        } else if (displayable == frmP_2) {
            if (command == cmdBack_P) {
                // write pre-action user code here
                switchDisplayable(null, getFrmA_1());
                // write post-action user code here
            } else if (command == cmdBack_P2) {
                curRec--;
    action user code here                    viewRecord2();// write pre-

                // write post-action user code here
            } else if (command == cmdNext2) {
                curRec++;
                                             viewRecord2();
                // write pre-action user code here

                // write post-action user code here
            }
        } else if (displayable == frmP_3) {
            if (command == cmdBack_P) {
                // write pre-action user code here
                switchDisplayable(null, getFrmA_1());
                // write post-action user code here
            } else if (command == cmdBack_P3) {
                curRec--;
    action user code here                    viewRecord3();// write pre-

                // write post-action user code here
            } else if (command == cmdNext3) {
                curRec++;
                                             viewRecord3();
                // write pre-action user code here

                // write post-action user code here
            }
        } else if (displayable == frmPb) {
            if (command == cmdOkPB) {
                // write pre-action user code here
                switchDisplayable(null, getFrmBPB());
                // write post-action user code here
            }
        } else if (displayable == frmSM) {
            if (command == okCommand2) {
                // write pre-action user code here
                switchDisplayable(null, getFrmBSM());
                // write post-action user code here
            }
        } else if (displayable == frm_B7) {
            if (command == cmdOk_B7) {

                switch(chc_B7.getSelectedIndex()){    // write pre-
    action user code here


                    case 0 :
                        getDisplay().setCurrent(getFrmLihat_B4());
                        LihatLapangan reg = new LihatLapangan ();
                        reg.setListener(this);
                        reg.start();
                        break;
                    case 1 :
                        getDisplay().setCurrent(getFormMk());
                        LihatLapangan2 reg2 = new LihatLapangan2 ();
                        reg2.setListener(this);
                        reg2.start();
                        break;
                    case 2 :
                        getDisplay().setCurrent(getFrmSM());
                        LihatLapangan3 reg3 = new LihatLapangan3 ();
                        reg3.setListener(this);
                        reg3.start();
                        break;

                }
                // write pre-action user code here

                // write post-action user code here
            }
        } else if (displayable == splashScreen) {
            if (command == SplashScreen.DISMISS_COMMAND) {
                // write pre-action user code here
                switchDisplayable(null, getFrmA_1());
                // write post-action user code here
            }
        }
        // write post-action user code here
    }
    //</editor-fold>


    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
splashScreen ">
    /**
     * Returns an initiliazed instance of splashScreen component.
     * @return the initialized component instance
     */
    public SplashScreen getSplashScreen() {
        if (splashScreen == null) {
            // write pre-init user code here
            splashScreen = new SplashScreen(getDisplay());
            splashScreen.setTitle("");
            splashScreen.setCommandListener(this);
            splashScreen.setImage(getImage6());
```

```
            splashScreen.setText("Silahkan Tunggu....");
            // write post-init user code here
        }
        return splashScreen;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmA_1 ">
    /**
     * Returns an initiliazed instance of frmA_1 component.
     * @return the initialized component instance
     */
    public Form getFrmA_1() {
        if (frmA_1 == null) {
            // write pre-init user code here
            frmA_1 = new Form("Menu Utama", new Item[] { getChcA_3()
});
            frmA_1.addCommand(getCmdA_4());
            frmA_1.setCommandListener(this);
            // write post-init user code here
        }
        return frmA_1;
    }
    //</editor-fold>



    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
chcA_3 ">
    /**
     * Returns an initiliazed instance of chcA_3 component.
     * @return the initialized component instance
     */
    public ChoiceGroup getChcA_3() {
        if (chcA_3 == null) {
            // write pre-init user code here
            chcA_3 = new ChoiceGroup("Pilihan Menu Utama",
Choice.EXCLUSIVE);
            chcA_3.append("Review Lapangan 1", getImage1());
            chcA_3.append("Review Lapangan 2", getImage1());
            chcA_3.append("Review Lapangan 3", getImage1());
            chcA_3.append("Pemesanan", getImage2());
            chcA_3.append("Pembatalan", getImage3());
            chcA_3.append("Info", getImage4());
            chcA_3.append("Keluar", getImage5());
            chcA_3.setFitPolicy(Choice.TEXT_WRAP_DEFAULT);
            chcA_3.setSelectedFlags(new boolean[] { false, false,
false, false, false, false, false });
            chcA_3.setFont(0, null);
            chcA_3.setFont(1, null);
            chcA_3.setFont(2, null);
            chcA_3.setFont(3, null);
            chcA_3.setFont(4, null);
            chcA_3.setFont(5, null);
            chcA_3.setFont(6, null);
            // write post-init user code here
        }
        return chcA_3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdA_4 ">
    /**
     * Returns an initiliazed instance of cmdA_4 component.
     * @return the initialized component instance
     */
    public Command getCmdA_4() {
        if (cmdA_4 == null) {
            // write pre-init user code here
            cmdA_4 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdA_4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmP_1 ">
    /**
     * Returns an initiliazed instance of frmP_1 component.
     * @return the initialized component instance
     */
    public Form getFrmP_1() {
        if (frmP_1 == null) {
            // write pre-init user code here
            frmP_1 = new Form("Lapangan 1", new Item[] {
getStringItem1() });
            frmP_1.addCommand(getCmdBack_P());
            frmP_1.addCommand(getCmdNext());
            frmP_1.addCommand(getCmdBack_P1());
            frmP_1.setCommandListener(this);
            // write post-init user code here
        }
        return frmP_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmB_1 ">
    /**
     * Returns an initiliazed instance of frmB_1 component.
     * @return the initialized component instance
     */
    public Form getFrmB_1() {
        if (frmB_1 == null) {
            // write pre-init user code here
            frmB_1 = new Form("Login", new Item[] { getStrB_1(),
getTxtNama_1(), getTxtPass_1() });
            frmB_1.addCommand(getCmdOk_B());
            frmB_1.addCommand(getCmdBack_B());
            frmB_1.setCommandListener(this);
            // write post-init user code here
        }
        return frmB_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmBt_1 ">
    /**
     * Returns an initiliazed instance of frmBt_1 component.
     * @return the initialized component instance
     */
    public Form getFrmBt_1() {
        if (frmBt_1 == null) {
            // write pre-init user code here
            frmBt_1 = new Form("Login", new Item[] { getStrBt_1(),
getTxtNama_2(), getTxtPass_2() });
            frmBt_1.addCommand(getCmdOk_Bt());
            frmBt_1.addCommand(getCmdBack_Bt());
            frmBt_1.setCommandListener(this);
            // write post-init user code here
        }
        return frmBt_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmIn_1 ">
    /**
     * Returns an initiliazed instance of frmIn_1 component.
     * @return the initialized component instance
     */
    public Form getFrmIn_1() {
        if (frmIn_1 == null) {
            // write pre-init user code here
            frmIn_1 = new Form("Menu Info", new Item[] {
getStringItem2() });
            frmIn_1.addCommand(getCmdBack_In());
            frmIn_1.setCommandListener(this);
            // write post-init user code here
        }
        return frmIn_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
strB_1 ">
    /**
     * Returns an initiliazed instance of strB_1 component.
     * @return the initialized component instance
     */
    public StringItem getStrB_1() {
        if (strB_1 == null) {
            // write pre-init user code here
            strB_1 = new StringItem("", " Selamat Datang, Sebelum
memulai transaksi pemesanan pelenggan diharapkan melakukan Login
terlebih dahulu  dengan memasukkan nama dan password.", Item.PLAIN);
            strB_1.setPreferredSize(-1, -1);
            strB_1.setLayout(ImageItem.LAYOUT_DEFAULT);
            // write post-init user code here
        }
        return strB_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtNama_1 ">
    /**
     * Returns an initiliazed instance of txtNama_1 component.
     * @return the initialized component instance
     */
    public TextField getTxtNama_1() {
        if (txtNama_1 == null) {
            // write pre-init user code here
            txtNama_1 = new TextField("Nama", null, 32,
TextField.ANY);
            // write post-init user code here
        }
        return txtNama_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtPass_1 ">
    /**
     * Returns an initiliazed instance of txtPass_1 component.
     * @return the initialized component instance
     */
    public TextField getTxtPass_1() {
        if (txtPass_1 == null) {
            // write pre-init user code here
            txtPass_1 = new TextField("Password", null, 32,
TextField.ANY | TextField.PASSWORD);
            txtPass_1.setLayout(ImageItem.LAYOUT_DEFAULT);
            txtPass_1.setInitialInputMode("UCB_BASIC_LATIN");
            // write post-init user code here
        }
        return txtPass_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
strBt_1 ">
    /**
```

```
     * Returns an initiliazed instance of strBt_1 component.
     * @return the initialized component instance
     */
    public StringItem getStrBt_1() {
        if (strBt_1 == null) {
            // write pre-init user code here
            strBt_1 = new StringItem("", "Untuk melakukan pembatalan
pemesan diharapkan login terlebih dahulu.");
            // write post-init user code here
        }
        return strBt_1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtNama_2 ">
    /**
     * Returns an initiliazed instance of txtNama_2 component.
     * @return the initialized component instance
     */
    public TextField getTxtNama_2() {
        if (txtNama_2 == null) {
            // write pre-init user code here
            txtNama_2 = new TextField("Nama", null, 32,
TextField.ANY);
            // write post-init user code here
        }
        return txtNama_2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtPass_2 ">
    /**
     * Returns an initiliazed instance of txtPass_2 component.
     * @return the initialized component instance
     */
    public TextField getTxtPass_2() {
        if (txtPass_2 == null) {
            // write pre-init user code here
            txtPass_2 = new TextField("Password", null, 32,
TextField.ANY | TextField.PASSWORD);
            // write post-init user code here
        }
        return txtPass_2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_B ">
    /**
     * Returns an initiliazed instance of cmdOk_B component.
     * @return the initialized component instance
     */
    public Command getCmdOk_B() {
        if (cmdOk_B == null) {
            // write pre-init user code here
            cmdOk_B = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_B;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_B ">
    /**
     * Returns an initiliazed instance of cmdBack_B component.
     * @return the initialized component instance
     */
    public Command getCmdBack_B() {
        if (cmdBack_B == null) {
            // write pre-init user code here
            cmdBack_B = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_B;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_Bt ">
    /**
     * Returns an initiliazed instance of cmdOk_Bt component.
     * @return the initialized component instance
     */
    public Command getCmdOk_Bt() {
        if (cmdOk_Bt == null) {
            // write pre-init user code here
            cmdOk_Bt = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_Bt;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_Bt ">
    /**
     * Returns an initiliazed instance of cmdBack_Bt component.
     * @return the initialized component instance
     */
    public Command getCmdBack_Bt() {
        if (cmdBack_Bt == null) {
            // write pre-init user code here
            cmdBack_Bt = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_Bt;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_In ">
    /**
     * Returns an initiliazed instance of cmdOk_In component.
     * @return the initialized component instance
     */
    public Command getCmdOk_In() {
        if (cmdOk_In == null) {
            // write pre-init user code here
            cmdOk_In = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_In;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_In ">
    /**
     * Returns an initiliazed instance of cmdBack_In component.
     * @return the initialized component instance
     */
    public Command getCmdBack_In() {
        if (cmdBack_In == null) {
            // write pre-init user code here
            cmdBack_In = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_In;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_P ">
    /**
     * Returns an initiliazed instance of cmdOk_P component.
     * @return the initialized component instance
     */
    public Command getCmdOk_P() {
        if (cmdOk_P == null) {
            // write pre-init user code here
            cmdOk_P = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_P;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_P ">
    /**
     * Returns an initiliazed instance of cmdBack_P component.
     * @return the initialized component instance
     */
    public Command getCmdBack_P() {
        if (cmdBack_P == null) {
            // write pre-init user code here
            cmdBack_P = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_P;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_B1 ">
    /**
     * Returns an initiliazed instance of frmKonv_B1 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_B1() {
        if (frmKonv_B1 == null) {
            // write pre-init user code here
            frmKonv_B1 = new Form("Konvirmasi Kesalahan", new Item[] {
getStrKon_B1() });
            frmKonv_B1.addCommand(getCmdBack_B1());
            frmKonv_B1.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_B1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
strKon_B1 ">
    /**
     * Returns an initiliazed instance of strKon_B1 component.
     * @return the initialized component instance
     */
    public StringItem getStrKon_B1() {
        if (strKon_B1 == null) {
            // write pre-init user code here
            strKon_B1 = new StringItem("", null);
            // write post-init user code here
        }
        return strKon_B1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_B2 ">
    /**
     * Returns an initiliazed instance of frmKonv_B2 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_B2() {
```

```
        if (frmKonv_B2 == null) {
            // write pre-init user code here
            frmKonv_B2 = new Form("Konfirmasi Koneksi", new Item[] {
getStr_B2() });
            frmKonv_B2.addCommand(getCmdOk_B2());
            frmKonv_B2.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_B2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_B3 ">
    /**
     * Returns an initiliazed instance of frmKonv_B3 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_B3() {
        if (frmKonv_B3 == null) {
            // write pre-init user code here
            frmKonv_B3 = new Form("Berhasil");
            frmKonv_B3.addCommand(getCmdOk_B3());
            frmKonv_B3.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_B3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmLihat_B4 ">
    /**
     * Returns an initiliazed instance of frmLihat_B4 component.
     * @return the initialized component instance
     */
    public Form getFrmLihat_B4() {
        if (frmLihat_B4 == null) {
            // write pre-init user code here
            frmLihat_B4 = new Form("Lapangan");
            frmLihat_B4.addCommand(getCmdOk_B4());
            frmLihat_B4.addCommand(getCmdBack_B4());
            frmLihat_B4.setCommandListener(this);
            // write post-init user code here
        }
        return frmLihat_B4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmPesan_B5 ">
    /**
     * Returns an initiliazed instance of frmPesan_B5 component.
     * @return the initialized component instance
     */
    public Form getFrmPesan_B5() {
        if (frmPesan_B5 == null) {
            // write pre-init user code here
            frmPesan_B5 = new Form("Lapangan", new Item[] {
getTxtLapangan_B5(), getTxtLapangan1_B5(), getTxtLapangan2_B5() });
            frmPesan_B5.addCommand(getCmdOk_B5());
            frmPesan_B5.addCommand(getCmdBack_B5());
            frmPesan_B5.setCommandListener(this);
            // write post-init user code here
        }
        return frmPesan_B5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_B6 ">
    /**
     * Returns an initiliazed instance of frmKonv_B6 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_B6() {
        if (frmKonv_B6 == null) {
            // write pre-init user code here
            frmKonv_B6 = new Form("Report Server");
            frmKonv_B6.addCommand(getCmdOk_B6());
            frmKonv_B6.addCommand(getCmdExit_B6());
            frmKonv_B6.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_B6;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_Bt1 ">
    /**
     * Returns an initiliazed instance of frmKonv_Bt1 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_Bt1() {
        if (frmKonv_Bt1 == null) {
            // write pre-init user code here
            frmKonv_Bt1 = new Form("Konfirmasi Kesalahan", new Item[]
{ getStrBt_2() });
            frmKonv_Bt1.addCommand(getCmdBack_Bt1());
            frmKonv_Bt1.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_Bt1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_Bt2 ">
    /**
     * Returns an initiliazed instance of frmKonv_Bt2 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_Bt2() {
        if (frmKonv_Bt2 == null) {
            // write pre-init user code here
            frmKonv_Bt2 = new Form("Konfirmasi Sukses", new Item[] {
getStringItem() });
            frmKonv_Bt2.addCommand(getCmdOk_Bt2());
            frmKonv_Bt2.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_Bt2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_Bt3 ">
    /**
     * Returns an initiliazed instance of frmKonv_Bt3 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_Bt3() {
        if (frmKonv_Bt3 == null) {
            // write pre-init user code here
            frmKonv_Bt3 = new Form("Berhasil");
            frmKonv_Bt3.addCommand(getCmdOk_Bt3());
            frmKonv_Bt3.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_Bt3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmList_Bt4 ">
    /**
     * Returns an initiliazed instance of frmList_Bt4 component.
     * @return the initialized component instance
     */
    public Form getFrmList_Bt4() {
        if (frmList_Bt4 == null) {
            // write pre-init user code here
            frmList_Bt4 = new Form("Lapangan yang anda Pesan");
            frmList_Bt4.addCommand(getCmdOk_Bt4());
            frmList_Bt4.setCommandListener(this);
            // write post-init user code here
        }
        return frmList_Bt4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmBatal_Bt5 ">
    /**
     * Returns an initiliazed instance of frmBatal_Bt5 component.
     * @return the initialized component instance
     */
    public Form getFrmBatal_Bt5() {
        if (frmBatal_Bt5 == null) {
            // write pre-init user code here
            frmBatal_Bt5 = new Form("Batal", new Item[] { getTxtB_1(),
getTxtB_2(), getStringItem3() });
            frmBatal_Bt5.addCommand(getCmdOk_Bt5());
            frmBatal_Bt5.addCommand(getCmdBack_Bt5());
            frmBatal_Bt5.setCommandListener(this);
            // write post-init user code here
        }
        return frmBatal_Bt5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmKonv_Bt6 ">
    /**
     * Returns an initiliazed instance of frmKonv_Bt6 component.
     * @return the initialized component instance
     */
    public Form getFrmKonv_Bt6() {
        if (frmKonv_Bt6 == null) {
            // write pre-init user code here
            frmKonv_Bt6 = new Form("Konfirmasi Server");
            frmKonv_Bt6.addCommand(getCmdBack_Bt6());
            frmKonv_Bt6.addCommand(getCmdExit_Bt6());
            frmKonv_Bt6.setCommandListener(this);
            // write post-init user code here
        }
        return frmKonv_Bt6;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_B1 ">
    /**
     * Returns an initiliazed instance of cmdBack_B1 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_B1() {
        if (cmdBack_B1 == null) {
            // write pre-init user code here
            cmdBack_B1 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_B1;
    }
    //</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_B2 ">
    /**
     * Returns an initiliazed instance of cmdOk_B2 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_B2() {
        if (cmdOk_B2 == null) {
            // write pre-init user code here
            cmdOk_B2 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_B2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_B3 ">
    /**
     * Returns an initiliazed instance of cmdOk_B3 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_B3() {
        if (cmdOk_B3 == null) {
            // write pre-init user code here
            cmdOk_B3 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_B3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_B4 ">
    /**
     * Returns an initiliazed instance of cmdOk_B4 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_B4() {
        if (cmdOk_B4 == null) {
            // write pre-init user code here
            cmdOk_B4 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_B4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_B4 ">
    /**
     * Returns an initiliazed instance of cmdBack_B4 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_B4() {
        if (cmdBack_B4 == null) {
            // write pre-init user code here
            cmdBack_B4 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_B4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_B5 ">
    /**
     * Returns an initiliazed instance of cmdOk_B5 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_B5() {
        if (cmdOk_B5 == null) {
            // write pre-init user code here
            cmdOk_B5 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_B5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_B5 ">
    /**
     * Returns an initiliazed instance of cmdBack_B5 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_B5() {
        if (cmdBack_B5 == null) {
            // write pre-init user code here
            cmdBack_B5 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_B5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_B6 ">
    /**
     * Returns an initiliazed instance of cmdOk_B6 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_B6() {
        if (cmdOk_B6 == null) {
            // write pre-init user code here
            cmdOk_B6 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_B6;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdExit_B6 ">
    /**
     * Returns an initiliazed instance of cmdExit_B6 component.
     * @return the initialized component instance
     */
    public Command getCmdExit_B6() {
        if (cmdExit_B6 == null) {
            // write pre-init user code here
            cmdExit_B6 = new Command("Exit", Command.EXIT, 0);
            // write post-init user code here
        }
        return cmdExit_B6;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_Bt1 ">
    /**
     * Returns an initiliazed instance of cmdBack_Bt1 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_Bt1() {
        if (cmdBack_Bt1 == null) {
            // write pre-init user code here
            cmdBack_Bt1 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_Bt1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_Bt2 ">
    /**
     * Returns an initiliazed instance of cmdOk_Bt2 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_Bt2() {
        if (cmdOk_Bt2 == null) {
            // write pre-init user code here
            cmdOk_Bt2 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_Bt2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_Bt3 ">
    /**
     * Returns an initiliazed instance of cmdOk_Bt3 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_Bt3() {
        if (cmdOk_Bt3 == null) {
            // write pre-init user code here
            cmdOk_Bt3 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_Bt3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_Bt4 ">
    /**
     * Returns an initiliazed instance of cmdOk_Bt4 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_Bt4() {
        if (cmdOk_Bt4 == null) {
            // write pre-init user code here
            cmdOk_Bt4 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_Bt4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_Bt5 ">
    /**
     * Returns an initiliazed instance of cmdOk_Bt5 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_Bt5() {
        if (cmdOk_Bt5 == null) {
            // write pre-init user code here
            cmdOk_Bt5 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_Bt5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_Bt5 ">
    /**
     * Returns an initiliazed instance of cmdBack_Bt5 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_Bt5() {
        if (cmdBack_Bt5 == null) {
            // write pre-init user code here
            cmdBack_Bt5 = new Command("Back", Command.BACK, 0);
```

```
            // write post-init user code here
        }
        return cmdBack_Bt5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_Bt6 ">
    /**
     * Returns an initiliazed instance of cmdBack_Bt6 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_Bt6() {
        if (cmdBack_Bt6 == null) {
            // write pre-init user code here
            cmdBack_Bt6 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_Bt6;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdExit_Bt6 ">
    /**
     * Returns an initiliazed instance of cmdExit_Bt6 component.
     * @return the initialized component instance
     */
    public Command getCmdExit_Bt6() {
        if (cmdExit_Bt6 == null) {
            // write pre-init user code here
            cmdExit_Bt6 = new Command("Exit", Command.EXIT, 0);
            // write post-init user code here
        }
        return cmdExit_Bt6;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
str_B2 ">
    /**
     * Returns an initiliazed instance of str_B2 component.
     * @return the initialized component instance
     */
    public StringItem getStr_B2() {
        if (str_B2 == null) {
            // write pre-init user code here
            str_B2 = new StringItem("Koneksi", "\nTekan Ok untuk
terhubung ke server...");
            // write post-init user code here
        }
        return str_B2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtLapangan_B5 ">
    /**
     * Returns an initiliazed instance of txtLapangan_B5 component.
     * @return the initialized component instance
     */
    public TextField getTxtLapangan_B5() {
        if (txtLapangan_B5 == null) {
            // write pre-init user code here
            txtLapangan_B5 = new TextField("Masukkan Kode Lapangan",
null, 32, TextField.ANY);
            // write post-init user code here
        }
        return txtLapangan_B5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtLapangan1_B5 ">
    /**
     * Returns an initiliazed instance of txtLapangan1_B5 component.
     * @return the initialized component instance
     */
    public TextField getTxtLapangan1_B5() {
        if (txtLapangan1_B5 == null) {
            // write pre-init user code here
            txtLapangan1_B5 = new TextField("Jumlah Jam", "1", 32,
TextField.ANY);
            // write post-init user code here
        }
        return txtLapangan1_B5;
    }
    //</editor-fold>









    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
strBt_2 ">
    /**
     * Returns an initiliazed instance of strBt_2 component.
     * @return the initialized component instance
     */
    public StringItem getStrBt_2() {
        if (strBt_2 == null) {
            // write pre-init user code here
            strBt_2 = new StringItem("", null, Item.PLAIN);
            // write post-init user code here
        }
        return strBt_2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
stringItem ">
    /**
     * Returns an initiliazed instance of stringItem component.
     * @return the initialized component instance
     */
    public StringItem getStringItem() {
        if (stringItem == null) {
            // write pre-init user code here
            stringItem = new StringItem("Koneksi", "Tekan Ok untuk
terhubung ke Server");
            // write post-init user code here
        }
        return stringItem;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frm_B7 ">
    /**
     * Returns an initiliazed instance of frm_B7 component.
     * @return the initialized component instance
     */
    public Form getFrm_B7() {
        if (frm_B7 == null) {
            // write pre-init user code here
            frm_B7 = new Form("Pilihan Lapangan", new Item[] {
getStr_B7(), getChc_B7() });
            frm_B7.addCommand(getCmdOk_B7());
            frm_B7.setCommandListener(this);
            // write post-init user code here
        }
        return frm_B7;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
str_B7 ">
    /**
     * Returns an initiliazed instance of str_B7 component.
     * @return the initialized component instance
     */
    public StringItem getStr_B7() {
        if (str_B7 == null) {
            // write pre-init user code here
            str_B7 = new StringItem("Pilhan Lapangan", " Dibawah ini
merupakan pilihan Lapangan yang disediakan oleh server berdasarkan
Tempat.");
            // write post-init user code here
        }
        return str_B7;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
chc_B7 ">
    /**
     * Returns an initiliazed instance of chc_B7 component.
     * @return the initialized component instance
     */
    public ChoiceGroup getChc_B7() {
        if (chc_B7 == null) {
            // write pre-init user code here
            chc_B7 = new ChoiceGroup("Pilihan :", Choice.EXCLUSIVE);
            chc_B7.append("Lapangan 1", null);
            chc_B7.append("Lapangan 2", null);
            chc_B7.append("Lapangan 3", null);
            chc_B7.setSelectedFlags(new boolean[] { false, false,
false });
            chc_B7.setFont(0, null);
            chc_B7.setFont(1, null);
            chc_B7.setFont(2, null);
            // write post-init user code here
        }
        return chc_B7;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOk_B7 ">
    /**
     * Returns an initiliazed instance of cmdOk_B7 component.
     * @return the initialized component instance
     */
    public Command getCmdOk_B7() {
        if (cmdOk_B7 == null) {
            // write pre-init user code here
            cmdOk_B7 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOk_B7;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdNext ">
    /**
     * Returns an initiliazed instance of cmdNext component.
     * @return the initialized component instance
     */
    public Command getCmdNext() {
        if (cmdNext == null) {
           nu reg= new nu();

               reg.start();   // write pre-init user code here
```

```
            cmdNext = new Command("Next", Command.OK, 0);
            // write post-init user code here
        }
        return cmdNext;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_P1 ">
    /**
     * Returns an initiliazed instance of cmdBack_P1 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_P1() {
        if (cmdBack_P1 == null) {
            // write pre-init user code here
            cmdBack_P1 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_P1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
stringItem1 ">
    /**
     * Returns an initiliazed instance of stringItem1 component.
     * @return the initialized component instance
     */
    public StringItem getStringItem1() {
        if (stringItem1 == null) {
            // write pre-init user code here
            stringItem1 = new StringItem("", null);
            // write post-init user code here
        }
        return stringItem1;
    }
    //</editor-fold>



    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
okCommand ">
    /**
     * Returns an initiliazed instance of okCommand component.
     * @return the initialized component instance
     */
    public Command getOkCommand() {
        if (okCommand == null) {
            nu reg= new nu();
            reg.start();
                // write pre-init user code here
            okCommand = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return okCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image1 ">
    /**
     * Returns an initiliazed instance of image1 component.
     * @return the initialized component instance
     */
    public Image getImage1() {
        if (image1 == null) {
            // write pre-init user code here
            try {
                image1 = Image.createImage("/Users1.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return image1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image2 ">
    /**
     * Returns an initiliazed instance of image2 component.
     * @return the initialized component instance
     */
    public Image getImage2() {
        if (image2 == null) {
            // write pre-init user code here
            try {
                image2 = Image.createImage("/pemesan.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return image2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image3 ">
    /**
     * Returns an initiliazed instance of image3 component.
     * @return the initialized component instance
     */
    public Image getImage3() {
        if (image3 == null) {
            // write pre-init user code here
            try {
                image3 = Image.createImage("/Stop.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return image3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image4 ">
    /**
     * Returns an initiliazed instance of image4 component.
     * @return the initialized component instance
     */
    public Image getImage4() {
        if (image4 == null) {
            // write pre-init user code here
            try {
                image4 = Image.createImage("/Help book.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return image4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image5 ">
    /**
     * Returns an initiliazed instance of image5 component.
     * @return the initialized component instance
     */
    public Image getImage5() {
        if (image5 == null) {
            // write pre-init user code here
            try {
                image5 = Image.createImage("/Exit.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return image5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image6 ">
    /**
     * Returns an initiliazed instance of image6 component.
     * @return the initialized component instance
     */
    public Image getImage6() {
        if (image6 == null) {
            // write pre-init user code here
            try {
                image6 = Image.createImage("/Dep.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return image6;
    }
    //</editor-fold>



    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image7 ">
    /**
     * Returns an initiliazed instance of image7 component.
     * @return the initialized component instance
     */
    public Image getImage7() {
        if (image7 == null) {
            // write pre-init user code here
            image7 = Image.createImage(1, 1);
            // write post-init user code here
        }
        return image7;
    }
    //</editor-fold>



    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtB_1 ">
    /**
     * Returns an initiliazed instance of txtB_1 component.
     * @return the initialized component instance
     */
    public TextField getTxtB_1() {
        if (txtB_1 == null) {
            // write pre-init user code here
            txtB_1 = new TextField("Masukkan Kode Lapangan", null, 32,
TextField.ANY);
            // write post-init user code here
        }
        return txtB_1;
    }
    //</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
stringItem2 ">
    /**
     * Returns an initiliazed instance of stringItem2 component.
     * @return the initialized component instance
     */
    public StringItem getStringItem2() {
        if (stringItem2 == null) {
            // write pre-init user code here
            stringItem2 = new StringItem("Info", "Selamat Datang, Jika
ingin melihat keseluruhan lapangan futsal tanpa mempunyai keinginan
untuk memesan dapat masuk ke menu Review Lapangan Jika inggin memulai
transaksi pemesanan dapat masuk ke menu pemesanan. Jika ingin memulai
transaksi pembatalan dapat masuk ke menu pembatalan. Terima kasih.",
Item.PLAIN);
            // write post-init user code here
        }
        return stringItem2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
stringItem3 ">
    /**
     * Returns an initiliazed instance of stringItem3 component.
     * @return the initialized component instance
     */
    public StringItem getStringItem3() {
        if (stringItem3 == null) {
            // write pre-init user code here
            stringItem3 = new StringItem("*Peringantan", "Isikan
dengan data yang ingin di batalkan");
            // write post-init user code here
        }
        return stringItem3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmPb ">
    /**
     * Returns an initiliazed instance of frmPb component.
     * @return the initialized component instance
     */
    public Form getFrmPb() {
        if (frmPb == null) {
            // write pre-init user code here
            frmPb = new Form("Lapangan");
            frmPb.addCommand(getCmdOkPB());
            frmPb.setCommandListener(this);
            // write post-init user code here
        }
        return frmPb;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOkPB ">
    /**
     * Returns an initiliazed instance of cmdOkPB component.
     * @return the initialized component instance
     */
    public Command getCmdOkPB() {
        if (cmdOkPB == null) {
            // write pre-init user code here
            cmdOkPB = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOkPB;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
formMk ">
    /**
     * Returns an initiliazed instance of formMk component.
     * @return the initialized component instance
     */
    public Form getFormMk() {
        if (formMk == null) {
            // write pre-init user code here
            formMk = new Form("Lapangan", new Item[] { });
            formMk.addCommand(getOkCommand1());
            formMk.setCommandListener(this);
            // write post-init user code here
        }
        return formMk;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmSM ">
    /**
     * Returns an initiliazed instance of frmSM component.
     * @return the initialized component instance
     */
    public Form getFrmSM() {
        if (frmSM == null) {
            // write pre-init user code here
            frmSM = new Form("Form");
            frmSM.addCommand(getOkCommand2());
            frmSM.setCommandListener(this);
            // write post-init user code here
        }
        return frmSM;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
okCommand1 ">
    /**
     * Returns an initiliazed instance of okCommand1 component.
     * @return the initialized component instance
     */
    public Command getOkCommand1() {
        if (okCommand1 == null) {
            // write pre-init user code here
            okCommand1 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return okCommand1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
okCommand2 ">
    /**
     * Returns an initiliazed instance of okCommand2 component.
     * @return the initialized component instance
     */
    public Command getOkCommand2() {
        if (okCommand2 == null) {
            // write pre-init user code here
            okCommand2 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return okCommand2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmBPB ">
    /**
     * Returns an initiliazed instance of frmBPB component.
     * @return the initialized component instance
     */
    public Form getFrmBPB() {
        if (frmBPB == null) {
            // write pre-init user code here
            frmBPB = new Form("Lapangan", new Item[] {
getStringItem4() });
            frmBPB.addCommand(getBackCommand());
            frmBPB.setCommandListener(this);
            // write post-init user code here
        }
        return frmBPB;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmBMK ">
    /**
     * Returns an initiliazed instance of frmBMK component.
     * @return the initialized component instance
     */
    public Form getFrmBMK() {
        if (frmBMK == null) {
            // write pre-init user code here
            frmBMK = new Form("Lapangan", new Item[] { getTxtBMK(),
getTxtBMK1(), getTxtBMK2() });
            frmBMK.addCommand(getCmdOkBMK());
            frmBMK.addCommand(getCmdBackBMK());
            frmBMK.setCommandListener(this);
            // write post-init user code here
        }
        return frmBMK;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmBSM ">
    /**
     * Returns an initiliazed instance of frmBSM component.
     * @return the initialized component instance
     */
    public Form getFrmBSM() {
        if (frmBSM == null) {
            // write pre-init user code here
            frmBSM = new Form("Lapangan", new Item[] {
getTxtlapanganBSM(), getTxtLapanganBSM1(), getTxtLapanganBSM2() });
            frmBSM.addCommand(getCmdOkBSM());
            frmBSM.addCommand(getCmdBackBSM());
            frmBSM.setCommandListener(this);
            // write post-init user code here
        }
        return frmBSM;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtlapanganBSM ">
    /**
     * Returns an initiliazed instance of txtlapanganBSM component.
     * @return the initialized component instance
     */
    public TextField getTxtlapanganBSM() {
        if (txtlapanganBSM == null) {
            // write pre-init user code here
            txtlapanganBSM = new TextField("Masukkan Kode Lapangan",
null, 32, TextField.ANY);
            // write post-init user code here
        }
        return txtlapanganBSM;
    }
```

```
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtLapanganBSM1 ">
    /**
     * Returns an initiliazed instance of txtLapanganBSM1 component.
     * @return the initialized component instance
     */
    public TextField getTxtLapanganBSM1() {
        if (txtLapanganBSM1 == null) {
            // write pre-init user code here
            txtLapanganBSM1 = new TextField("Jumlah Jam", "1", 32,
TextField.ANY);
            // write post-init user code here
        }
        return txtLapanganBSM1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtLapanganBSM2 ">
    /**
     * Returns an initiliazed instance of txtLapanganBSM2 component.
     * @return the initialized component instance
     */
    public TextField getTxtLapanganBSM2() {
        if (txtLapanganBSM2 == null) {
            // write pre-init user code here
            txtLapanganBSM2 = new TextField("Pesan untuk tanggal",
"tttt-bb-hh", 32, TextField.ANY);
            // write post-init user code here
        }
        return txtLapanganBSM2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOkBSM ">
    /**
     * Returns an initiliazed instance of cmdOkBSM component.
     * @return the initialized component instance
     */
    public Command getCmdOkBSM() {
        if (cmdOkBSM == null) {
            // write pre-init user code here
            cmdOkBSM = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOkBSM;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBackBSM ">
    /**
     * Returns an initiliazed instance of cmdBackBSM component.
     * @return the initialized component instance
     */
    public Command getCmdBackBSM() {
        if (cmdBackBSM == null) {
            // write pre-init user code here
            cmdBackBSM = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBackBSM;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmBSM1 ">
    /**
     * Returns an initiliazed instance of frmBSM1 component.
     * @return the initialized component instance
     */
    public Form getFrmBSM1() {
        if (frmBSM1 == null) {
            // write pre-init user code here
            frmBSM1 = new Form("Report Server");
            frmBSM1.addCommand(getOkCommand3());
            frmBSM1.setCommandListener(this);
            // write post-init user code here
        }
        return frmBSM1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
okCommand3 ">
    /**
     * Returns an initiliazed instance of okCommand3 component.
     * @return the initialized component instance
     */
    public Command getOkCommand3() {
        if (okCommand3 == null) {
            // write pre-init user code here
            okCommand3 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return okCommand3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtBMK ">
    /**
     * Returns an initiliazed instance of txtBMK component.
     * @return the initialized component instance
     */
    public TextField getTxtBMK() {
        if (txtBMK == null) {
            // write pre-init user code here
            txtBMK = new TextField("Masukkan Kode Lapangan", null, 32,
TextField.ANY);
            // write post-init user code here
        }
        return txtBMK;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtBMK1 ">
    /**
     * Returns an initiliazed instance of txtBMK1 component.
     * @return the initialized component instance
     */
    public TextField getTxtBMK1() {
        if (txtBMK1 == null) {
            // write pre-init user code here
            txtBMK1 = new TextField("Jumlah Jam", "1", 32,
TextField.ANY);
            // write post-init user code here
        }
        return txtBMK1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtBMK2 ">
    /**
     * Returns an initiliazed instance of txtBMK2 component.
     * @return the initialized component instance
     */
    public TextField getTxtBMK2() {
        if (txtBMK2 == null) {
            // write pre-init user code here
            txtBMK2 = new TextField("Pesan untuk tanggal", "tttt-bb-
hh", 32, TextField.ANY);
            // write post-init user code here
        }
        return txtBMK2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmBMK1 ">
    /**
     * Returns an initiliazed instance of frmBMK1 component.
     * @return the initialized component instance
     */
    public Form getFrmBMK1() {
        if (frmBMK1 == null) {
            // write pre-init user code here
            frmBMK1 = new Form("Report Server");
            frmBMK1.addCommand(getOkCommand4());
            frmBMK1.setCommandListener(this);
            // write post-init user code here
        }
        return frmBMK1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdOkBMK ">
    /**
     * Returns an initiliazed instance of cmdOkBMK component.
     * @return the initialized component instance
     */
    public Command getCmdOkBMK() {
        if (cmdOkBMK == null) {
            // write pre-init user code here
            cmdOkBMK = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return cmdOkBMK;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBackBMK ">
    /**
     * Returns an initiliazed instance of cmdBackBMK component.
     * @return the initialized component instance
     */
    public Command getCmdBackBMK() {
        if (cmdBackBMK == null) {
            // write pre-init user code here
            cmdBackBMK = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBackBMK;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
stringItem4 ">
    /**
     * Returns an initiliazed instance of stringItem4 component.
```

```
     * @return the initialized component instance
     */
    public StringItem getStringItem4() {
        if (stringItem4 == null) {
            // write pre-init user code here
            stringItem4 = new StringItem("Peringatan!", "Transaksi
Tidak ada.");
            // write post-init user code here
        }
        return stringItem4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
okCommand4 ">
    /**
     * Returns an initiliazed instance of okCommand4 component.
     * @return the initialized component instance
     */
    public Command getOkCommand4() {
        if (okCommand4 == null) {
            // write pre-init user code here
            okCommand4 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return okCommand4;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
backCommand ">
    /**
     * Returns an initiliazed instance of backCommand component.
     * @return the initialized component instance
     */
    public Command getBackCommand() {
        if (backCommand == null) {
            // write pre-init user code here
            backCommand = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return backCommand;
    }
    //</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmP_2 ">
    /**
     * Returns an initiliazed instance of frmP_2 component.
     * @return the initialized component instance
     */
    public Form getFrmP_2() {
        if (frmP_2 == null) {
            // write pre-init user code here
            frmP_2 = new Form("Lapangan 2", new Item[] {
getStringItem5() });
            frmP_2.addCommand(getCmdBack_P());
            frmP_2.addCommand(getCmdNext2());
            frmP_2.addCommand(getCmdBack_P2());
            frmP_2.setCommandListener(this);
            // write post-init user code here
        }
        return frmP_2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
image8 ">
    /**
     * Returns an initiliazed instance of image8 component.
     * @return the initialized component instance
     */
    public Image getImage8() {
        if (image8 == null) {
            // write pre-init user code here
            image8 = Image.createImage(1, 1);
            // write post-init user code here
        }
        return image8;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
frmP_3 ">
    /**
     * Returns an initiliazed instance of frmP_3 component.
     * @return the initialized component instance
     */
    public Form getFrmP_3() {
        if (frmP_3 == null) {
            // write pre-init user code here
            frmP_3 = new Form("Lapangan 3", new Item[] {
getStringItem8() });
            frmP_3.addCommand(getCmdBack_P());
            frmP_3.addCommand(getCmdNext3());
            frmP_3.addCommand(getCmdBack_P3());
            frmP_3.setCommandListener(this);
            // write post-init user code here
        }
        return frmP_3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
okCommand5 ">
    /**
     * Returns an initiliazed instance of okCommand5 component.
     * @return the initialized component instance
     */
    public Command getOkCommand5() {
        if (okCommand5 == null) {
            // write pre-init user code here
            okCommand5 = new Command("Ok", Command.OK, 0);
            // write post-init user code here
        }
        return okCommand5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdNext2 ">
    /**
     * Returns an initiliazed instance of cmdNext2 component.
     * @return the initialized component instance
     */
    public Command getCmdNext2() {
        if (cmdNext2 == null) {
            nu2 reg= new nu2();
            reg.start();// write pre-init user code here
            cmdNext2 = new Command("Next", Command.OK, 0);
            // write post-init user code here
        }
        return cmdNext2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
stringItem5 ">
    /**
     * Returns an initiliazed instance of stringItem5 component.
     * @return the initialized component instance
     */
    public StringItem getStringItem5() {
        if (stringItem5 == null) {
            // write pre-init user code here
            stringItem5 = new StringItem("", null);
            // write post-init user code here
        }
        return stringItem5;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
stringItem6 ">
    /**
     * Returns an initiliazed instance of stringItem6 component.
     * @return the initialized component instance
     */
    public StringItem getStringItem6() {
        if (stringItem6 == null) {
            // write pre-init user code here
            stringItem6 = new StringItem("", null);
            // write post-init user code here
        }
        return stringItem6;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdNext3 ">
    /**
     * Returns an initiliazed instance of cmdNext3 component.
     * @return the initialized component instance
     */
    public Command getCmdNext3() {
        if (cmdNext3 == null) {
           nu3 reg= new nu3();
            reg.start();// write pre-init user code here
            cmdNext3 = new Command("Next", Command.OK, 0);
            // write post-init user code here
        }
        return cmdNext3;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_P2 ">
    /**
     * Returns an initiliazed instance of cmdBack_P2 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_P2() {
        if (cmdBack_P2 == null) {
            // write pre-init user code here
            cmdBack_P2 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_P2;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cmdBack_P3 ">
    /**
     * Returns an initiliazed instance of cmdBack_P3 component.
     * @return the initialized component instance
     */
    public Command getCmdBack_P3() {
```

```
        if (cmdBack_P3 == null) {
            // write pre-init user code here
            cmdBack_P3 = new Command("Back", Command.BACK, 0);
            // write post-init user code here
        }
        return cmdBack_P3;
    }
    //</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtLapangan2_B5 ">
    /**
     * Returns an initiliazed instance of txtLapangan2_B5 component.
     * @return the initialized component instance
     */
    public TextField getTxtLapangan2_B5() {
        if (txtLapangan2_B5 == null) {
            // write pre-init user code here
            txtLapangan2_B5 = new TextField("Tanggal Yang Dipesan",
"tttt-bb-hh", 32, TextField.ANY);
            // write post-init user code here
        }
        return txtLapangan2_B5;
    }
    //</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
txtB_2 ">
    /**
     * Returns an initiliazed instance of txtB_2 component.
     * @return the initialized component instance
     */
    public TextField getTxtB_2() {
        if (txtB_2 == null) {
            // write pre-init user code here
            txtB_2 = new TextField("Tanggal yang di pesan", "tttt-bb-
hh", 32, TextField.ANY);
            // write post-init user code here
        }
        return txtB_2;
    }
    //</editor-fold>
```

```
    /**
     * Returns a display instance.
     * @return the display instance.
     */
    public Display getDisplay () {
        return Display.getDisplay(this);
    }

    /**
     * Exits MIDlet.
     */
    public void exitMIDlet() {
        switchDisplayable (null, null);
        destroyApp(true);
        notifyDestroyed();
    }

    /**
     * Called when MIDlet is started.
     * Checks whether the MIDlet have been already started and
initialize/starts or resumes the MIDlet.
     */
    public void startApp() {
        if (midletPaused) {
            resumeMIDlet ();
        } else {
            initialize ();
            startMIDlet ();
        }
        midletPaused = false;
    }

    /**
     * Called when MIDlet is paused.
     */
    public void pauseApp() {
        midletPaused = true;
    }
```

```
    /**
     * Called to signal the MIDlet to terminate.
     * @param unconditional if true, then the MIDlet has to be
unconditionally terminated and all resources has to be released.
     */
    public void destroyApp(boolean unconditional) {
    }
    private void viewRecord(){
                        if(curRec<0){
                                curRec=0;
                        }


                        if(curRec>record.length-1){
                                curRec=record.length-1;
                        }

                        UntypedObject uo=(UntypedObject)record[curRec];
                        stringItem1.setText("Kode       : " +
uo.getPropertyValue(0) + "\nStatus    : " + uo.getPropertyValue(1)  +
"\nJam       : " + uo.getPropertyValue(3) + "\nHarga     : " +
uo.getPropertyValue(4));
                }
    private void viewRecord2(){
                        if(curRec<0){
                                curRec=0;
                        }


                        if(curRec>record.length-1){
                                curRec=record.length-1;
                        }

                        UntypedObject uo=(UntypedObject)record[curRec];
                        stringItem5.setText("Kode       : " +
uo.getPropertyValue(0) + "\nStatus    : " + uo.getPropertyValue(1)  +
"\nJam       : " + uo.getPropertyValue(3) + "\nHarga     : " +
uo.getPropertyValue(4));
                }
    private void viewRecord3(){
                        if(curRec<0){
                                curRec=0;
                        }


                        if(curRec>record.length-1){
                                curRec=record.length-1;
                        }

                        UntypedObject uo=(UntypedObject)record[curRec];
                        stringItem6.setText("Kode       : " +
uo.getPropertyValue(0) + "\nStatus    : " + uo.getPropertyValue(1)  +
"\nJam       : " + uo.getPropertyValue(3) + "\nHarga     : " +
uo.getPropertyValue(4));
                }

class nu extends Thread {



public void run(){
                        try{
                                String
url="http://localhost/futsalOrder/WebService/Lapangan1.php";
                                curRec=0;

                                Call c=new Call();
                                c.setMethodName("getFilmsList");

                                HTTPTransport transport=new
HTTPTransport(url,null);
                                transport.getResponse(true);

                                Envelope res=c.invoke(transport);

                                record=(Object[])res.getParameter(0);



                                viewRecord();
                        }
                        catch(Exception ex){
                                stringItem.setText(ex.toString());
                        }
        finally{
        }
           }


}

class nu2 extends Thread {



public void run(){
                        try{
                                String
url="http://localhost/futsalOrder/WebService/Lapangan21.php";
                                curRec=0;

                                Call c=new Call();
```

```
                                c.setMethodName("getLapangan2List");

                                HTTPTransport transport=new
HTTPTransport(url,null);
                                transport.getResponse(true);

                                Envelope res=c.invoke(transport);

                                record=(Object[])res.getParameter(0);



                                viewRecord2();
                        }
                        catch(Exception ex){
                                stringItem.setText(ex.toString());
                        }
        finally{
        }
           }


}

class nu3 extends Thread {



public void run(){
                        try{
                                String
url="http://localhost/futsalOrder/WebService/Lapangan3.php";
                                curRec=0;

                                Call c=new Call();
                                c.setMethodName("getLapangan3List");

                                HTTPTransport transport=new
HTTPTransport(url,null);
                                transport.getResponse(true);

                                Envelope res=c.invoke(transport);

                                record=(Object[])res.getParameter(0);



                                viewRecord3();
                        }
                        catch(Exception ex){
                                stringItem.setText(ex.toString());
                        }
        finally{
        }
           }


}
    public String koneksi1 (String n,String m) throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url =
"http://localhost/futsalOrder/ag.php?a=c&n="+n+"&m="+m;
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 1;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[size];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[size];
            }

            input.close();

             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
               conn.close();
          }
        }
    }

   // Thread registrasi untuk melakukan koneksi dengan web server
   // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class login1 extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String pm = "";
            try {

               pm = koneksi1(nama1,pass1);
                frmKonv_B3.deleteAll();
                frmKonv_B3.append(pm);
              }
            catch(Exception err) {
                System.out.println(err);
                frmKonv_B1.append("Gagal terhubung ke server!");
            }
            finally {

                if(pm.equals("2")) {
                   getDisplay().setCurrent(getFrmKonv_B3());
                   frmKonv_B3.deleteAll();
                   frmKonv_B3.append("Anda telah berhasil Login
.\nSilahkan tekan Ok untuk memulai melihat Lapangan dan melakukan
transaksi Pemesanan Lapangan.");

                }

                if(pm.equals("1"))
                {
                   getDisplay().setCurrent(getFrmKonv_B1());
                    frmKonv_B1.deleteAll();
                    frmKonv_B1.append("\nNama atau password anda
salah.Silahkan ketik kembali atau hubungi kustomer service kami.");
                }



            }
        }
    }
    public String KonlihatLapangan () throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url = "http://localhost/futsalOrder/ab.php?a=b";
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();


             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
               conn.close();
          }
        }
    }

   // Thread registrasi untuk melakukan koneksi dengan web server
   // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class LihatLapangan extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String pm = "";
            try {
                pm = KonlihatLapangan ();
                frmLihat_B4.deleteAll();
                frmLihat_B4.append(pm);


              }
            catch(Exception err) {
                System.out.println(err);
                frmLihat_B4.append("Fail to get target file!");
            }
            finally {



            }
        }
    }
      public String KonlihatLapangan1 () throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url = "http://localhost/futsalOrder/abi.php?a=b";
```

```
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();


             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
              conn.close();
          }
        }
    }

    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class LihatLapangan1 extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String pa = "";
            try {
                pa = KonlihatLapangan1 ();
                frmPb.deleteAll();
                frmPb.append(pa);


              }
            catch(Exception err) {
                System.out.println(err);
                frmPb.append("Fail to get target file!");
            }
            finally {



            }
        }
    }
    public String KonlihatLapangan2 () throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url = "http://localhost/futsalOrder/ab2.php?a=b";
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();


             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
              conn.close();
          }
        }
    }

    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class LihatLapangan2 extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String pa = "";
            try {
                pa = KonlihatLapangan2 ();
                frmMk.deleteAll();
                frmMk.append(pa);


              }
            catch(Exception err) {
                System.out.println(err);
                frmMk.append("Fail to get target file!");
            }
            finally {



            }
        }
    }
     public String KonlihatLapangan3 () throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url = "http://localhost/futsalOrder/ab3.php?a=b";
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();


             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
              conn.close();
          }
        }
    }

    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class LihatLapangan3 extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String pa = "";
            try {
                pa = KonlihatLapangan3 ();
                frmSM.deleteAll();
                frmSM.append(pa);


              }
            catch(Exception err) {
                System.out.println(err);
                frmSM.append("Fail to get target file!");
            }
            finally {



            }
        }
    }
    public String KonPesan(String n,String a,String b,String c,String
d) throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url =
"http://localhost/futsalOrder/dau.php?d=n&n="+n+"&a="+a+"&b="+b+"&c="+
c;
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }
```

```
            input.close();

             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
               conn.close();
          }
        }
    }


    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class PesanLapangan extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String rs = "";
            try {
                rs = KonPesan
(nama1,lapangan,lapangan1,lapangan2,lapangan3);
                frmKonv_B6.deleteAll();
                frmKonv_B6.append(rs);
              }
            catch(Exception err) {
                System.out.println(err);
                frmKonv_B6.append("Fail to get target file!");
            }
            finally {

                //bersihkan variabel
                lapangan ="";
                lapangan1 ="";lapangan2 ="";lapangan3 ="";
                }

            }
        }
    public String KonPesan1(String n,String a,String b,String c,String
d) throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url =
"http://localhost/futsalOrder/dau1.php?d=r&n="+n+"&a="+a+"&b="+b+"&c="
+c;
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();

             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
               conn.close();
          }
        }
    }


    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class PesanLapangan1 extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String rs = "";
            try {
                rs = KonPesan1
(nama1,lapangan5,lapangan6,lapangan7,lapangan8);
                frmBSM1.deleteAll();
                frmBSM1.append(rs);
              }
            catch(Exception err) {
                System.out.println(err);
                frmBSM1.append("Fail to get target file!");
            }
            finally {

                //bersihkan variabel
                lapangan ="";
                lapangan1 ="";lapangan2 ="";

                }

            }
        }
    public String KonPesan2(String n,String a,String b,String c,String
d) throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url =
"http://localhost/futsalOrder/dau2.php?d=r&n="+n+"&a="+a+"&b="+b+"&c="
+c;
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();

             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
               conn.close();
          }
        }
    }


    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class PesanLapangan2 extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String rs = "";
            try {
                rs = KonPesan2
(nama1,lapangan10,lapangan11,lapangan12,lapangan13);
                frmBMK1.deleteAll();
                frmBMK1.append(rs);
              }
            catch(Exception err) {
                System.out.println(err);
                frmKonv_B6.append("Fail to get target file!");
            }
            finally {

                //bersihkan variabel
                lapangan ="";
                lapangan1 ="";lapangan2 ="";

                }

            }
        }
public String koneksi2 (String g,String h) throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url =
"http://localhost/futsalOrder/ag1.php?a=c&g="+g+"&h="+h;
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 1;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[size];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[size];
            }

            input.close();

             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;
```

```
        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
                conn.close();
          }
        }
     }

    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
     class login2 extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String pm = "";
            try {
               pm = koneksi2(nama2,pass2);
                frmKonv_Bt3.deleteAll();
                frmKonv_Bt3.append(pm);
              }
            catch(Exception err) {
                System.out.println(err);
                frmKonv_Bt1.append("Gagal terhubung ke server!");
            }
            finally {

                if(pm.equals("2")) {
                   getDisplay().setCurrent(getFrmKonv_Bt3());
                   frmKonv_Bt3.deleteAll();
                   frmKonv_Bt3.append("Anda telah berhasil
Login.\nSilahkan tekan Lanjut untuk memulai melihat Lapangan.");
                }

                if(pm.equals("1"))
                {
                   getDisplay().setCurrent(getFrmKonv_Bt1());
                    frmKonv_Bt1.deleteAll();
                    frmKonv_Bt1.append("\nNama atau password anda
salah.Silahkan ketik kembali atau hubungi kustomer service kami.");
                }




            }
        }
    }
      public String KonListPesan (String g) throws IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            // lakukan koneksi
            String url =
"http://localhost/futsalOrder/daa.php?a=b&g="+g;
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();


             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
                conn.close();
          }
        }
    }

    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class ListPesan extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String rs = "";
            try {
                rs = KonListPesan (nama2);
                frmList_Bt4.deleteAll();
                frmList_Bt4.append(rs);
              }
            catch(Exception err) {
                System.out.println(err);
                frmList_Bt4.append("Fail to get target file!");
            }
            finally {



            }
        }
    }
      public String KonBatal(String g,String k,String r) throws
IOException {
        //1. Buat obyek koneksi HTTP
        HttpConnection conn = null;

        try {
            String url =
"http://localhost/futsalOrder/db.php?d=r&g="+g+"&k="+k+"&r="+r;
            conn = (HttpConnection)Connector.open(url);

            // Mengambil data message dari web server
            int size = 10;
            StringBuffer tmp = new StringBuffer(size);
            InputStreamReader input = new
InputStreamReader(conn.openInputStream());
            char[] buffer = new char[25];

            while ((input.read(buffer,0,buffer.length)) != -1) {
                tmp.append(buffer);
                buffer = new char[25];
            }

            input.close();

             //5. Kembalikan hasilnya
             String result= tmp.toString();
             return result;

        } catch(Exception er) {
            throw new IllegalArgumentException(er.getMessage());
        }
        finally {
          //6. Tutup koneksi HTTP
          if (conn != null) {
                conn.close();
          }
        }
    }


    // Thread registrasi untuk melakukan koneksi dengan web server
    // Koneksi sebaiknya dibuat dengan thread supaya tidak terjadi
deadlock
    class BatalPemesanan extends Thread {
        private CommandListener listener;
        public void setListener (CommandListener cl) {
            listener = cl;
        }
        public void run() {
            String rs = "";
            try {
                rs = KonBatal (nama2,lapangan,lapangan1);
                frmKonv_Bt6.deleteAll();
                frmKonv_Bt6.append(rs);
              }
            catch(Exception err) {
                System.out.println(err);
                frmKonv_Bt6.append("Fail to get target file!");
            }
            finally {

                //bersihkan variabel
                lapangan ="";
                lapangan1 ="";lapangan2 ="";

                }

            }
        }
}
```