RANCANG BANGUN APLIKASI PENDAFTARAN *ONLINE* TURNAMEN *PLAYER UNKOWN BATTLE GROUND* DI *THE PILLARS E-SPORT* BERBASIS *WEB*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh kelulusan
Jenjang Strata Satu (S1)
Pada Program Studi Sistem Informasi

Oleh
KIKI MAULANA MALIK
351743004

SEKOLAH TINGGI MANAJEMEN SISTEM INFORMASI
DAN KOMPUTER INDONESIA MANDIRI

2021

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI PENDAFTARAN *ONLINE* TURNAMEN *PLAYER UNKOWN BATTLE GROUND* DI *THE PILLARS E-SPORT* BERBASIS *WEB*

Oleh
Kiki maulana malik
351743004

Tugas Akhir ini telah diterima dan disahkan untuk
Memenuhi persyaratan mencapai gelar

SARJANA SISTEM INFORMASI

Pada
PROGRAM STUDI SISTEM INFORMASI
SEKOLAH TINGGI MANAJEMEN SISTEM INFORMASI DAN KOMPUTER
INDONESIA MANDIRI

Bandung, januari 2021

Disahkan oleh

Ketua Program Studi,	Dosen Pembimbing,
Moch Ali Ramdhani, S.T, M.Kom	Hendra Gunawan, S.T.,M.Kom
NIDN: 0403097701	NIDN: 0423037202

LEMBAR PERSETUJUAN REVISI

RANCANG BANGUN APLIKASI PENDAFTARAN *ONLINE* TURNAMEN *PLAYER UNKOWN BATTLE GROUND* DI *THE PILLARS E-SPORT* BERBASIS *WEB*

Oleh
Kiki Maulana Malik

351743004

Telah melakukan sidang tugas akhir dan telah melakukan revisi sesuai dengan perubahan dan perbaikan yang diminta pada saat sidang tugas akhir.

Bandung, januari 2021

Menyetujui

No	Nama Dosen	Keterangan	Tanda Tangan
1.	Hendra Gunawan, S.T.,M.Kom	Pembimbing	
2.	Haryoso Wicaksono, S.Si., M.M., M.Kom	Penguji 1	
3.	. Chalifa Chazar, S.T, M.T	Penguji 2	

Mengetahui
Ketua Program Studi Sistem Informasi

Moch Ali Ramdhani, S.T, M.Kom
NIDN: 0403097701

ABSTRAK

RANCANG BANGUN APLIKASI PENDAFTARAN *ONLINE* TURNAMEN *PLAYER UNKOWN BATTLE GROUND* DI *THE PILLARS E-SPORT* BERBASIS *WEB*

Oleh
Kiki Maulana Malik
351743004

The Pillars E-sport adalah organisasi yang aktif dalam mengikuti *event* turnamen *game online* dan membuat *event* turnamen *game online* seperti membuat *event* turnamen *PLAYER UNKOWN BATTLE GROUND MOBILE*. Banyak nya pendaftar turnamen di *The Pillars E-sport* sekarang ini sering kali membuat panitia pendaftaran kewalahan dalam melakukan penyelenggaraan turnamen karena sistem pendaftaran masih secara konvensional . Bahkan petugas panitia pendaftaran di organisasi tersebut sering melakukan pembuatan laporan yang salah karena pengelolaan data masih manual dan kurang akurat. Maka penulis berusaha membangun sistem Perancangan Aplikasi Pendaftaran Turnamen *Online* Berbasis *Web*. Menggunakan *PHP* dan *MYSQL*, untuk memudahkan penggunanya mengetahui rekap data pendaftar. Penelitian ini bertujuan untuk mempermudah bagi calon pendaftar dan menghasilkan suatu sistem pendaftaran turnamen yang cepat dan akurat serta memudahkan dalam hal pendataan pendaftar turnamen, menghasilkan perancangan pendaftaran turnamen *online* berbasis *web*. Selain itu tampilan yang interaktifnya memudahkan *user* untuk melakukan pendataan pendaftar. Dari segi bahasa pemrograman *PHP*, *MYSQL* merupakan bahasa dan *database* yang sinkron saling bersinergi. Perancangan sistem ini menggunakan metode *waterfall*, sehingga dapat disimpulkan bahwa sistem ini layak jika aplikasi pendaftaran turnamen ini di terapkan di *The Pillars E-sport*.
Kata kunci : Perancangan, Pendaftaran, *E-sport*, Berbasis *Web*, *PHP*, *MYSQL*.

ABSTRACT

RANCANG BANGUN APLIKASI PENDAFTARAN *ONLINE* TURNAMEN *PLAYER UNKOWN BATTLE GROUND* DI *THE PILLARS E-SPORT* BERBASIS *WEB*

Oleh
Kiki Maulana Malik
351743004

The Pillars E-sport is an organization that is active in participating in online game tournament events and creating online game tournament events such as holding PLAYER UNKOWN BATTLE GROUND MOBILE tournament events. The large number of tournament registrants at The Pillars E-sport today often overwhelms the registration committee in organizing tournaments because the registration system is still conventional. Even the registration committee officers in these organizations often make incorrect reports because data management is still manual and inaccurate. So the authors try to build a Web-Based Online Tournament Registration Application Design system, Using PHP and MYSQL, to make it easier for users to know the registrant data recap. This study aims to make it easier for prospective registrants and produce a tournament registration system that is fast and accurate and makes it easier for tournament registrants to be registered, producing web-based online tournament registration designs. In addition, the interactive display makes it easy for users to collect registrant data. In terms of the PHP programming language, MYSQL is a synchronous language and database that synergizes with each other. The design of this system uses the waterfall method, so it can be concluded that this system is feasible if the tournament registration application is implemented in The Pillars E-sport.
Keywords: Design, Registration, E-sport, Web Based, PHP, MYSQL.

KATA PENGANTAR

Segala puji syukur atas kehadirat Tuhan Yang Maha Esa yang telah memberikan berkat-Nya kepada penulis berupa kesehatan, sehingga penulis dapat menyelesakan skripsi saya yang berjudul Rancang Bangun Aplikasi Pendaftaran Turnamen *Online* pada *The Pillars E-sport*s Berbasis *Web* tepat pada waktunya. Penulisan skripsi ini diajukan untuk memenuihi salah satu syarat kelulusan program studi Sistem Informasi di STMIK-IM Bandung.

Penghargaan dan terima kasih yang setulus-tulusnya kepada kedua orang tua yang telah mencurahkan segenap cinta dan kasih sayang serta perhatian moril dan materil. Semoga Allah SWT selalu melimpahkan Rahmat, Kesehatan, Karunia dan Keberkahan di dunia dan di akhirat atas budi baik yang telah diberikan kepada penulis.

Penghargaan dan terima kasih penulis berikan kepada Bapak Hendra Gunawan, S.T.,M.Kom selaku pembimbing yang telah membantu penulisan skripsi ini. Serta ucapan terima kasih kepada :

1. Seluruh Jajaran Institusi Kampus STMIK-IM
2. Seluruh Jajaran Managemen di *The Pillars E-sport*
3. Sahabat Kampus IT angkatan 2017 dan 2018

Akhir kata penulis penulis memohon saran dan kritik yang sifatnya membangun demi kesempurnaannya dan semoga bermanfaat bagi kita semua. Amiin.

Bandung, Januari 2021

Penulis

Kiki maulana malik

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi di bidang *game online* kini telah berkembang pesat di kalangan masyarakat bahkan hampir di setiap umur dan golongan. Seiring perkembangan jaman, *game online* telah memasuki tahap dimana *game online* bukan hanya menjadi hobi melainkan menjadi profesi beberapa orang yang disebut *pro player*.

Banyak yang sukses hanya dengan bermain *game*, tidak bisa dianggap remeh, keuntungan yang di dapat sampai puluhan juta setiap bulan contohnya dilansir jalantikus.com di Indonesia terdapat tujuh atlet *E-sport* dengan bayaran termahal Joshua Vega "Soifong" Tanaka (Rp52 juta) pemain *Counter Strike*: *Global Offensive* (CS:GO), Ryan "rynXTRFL" Prakasha (Rp53,2 juta) menggeluti *game Crossfire*, Farand "Koala" Kowara (Rp54 juta) bermain *game Dota 2*, Richard "frgd" Permana (Rp57,4 juta) pemain *Counter Strike*: *Global Offensive* (CS:GO), Muhammad "inYourdreaM" Rizky (Rp60,3 juta) pemain *game Dota 2*, Kevin "xccurate" Susanto (Rp65,3 juta), pemain *Counter Strike*: *Global Offensive* (CS:GO) dan Agung "Sys" Frianto (Rp81,3 juta) yang juga rekan satu tim Kevin "xccurate" Susanto pemain *Counter Strike*: *Global Offensive* (CS:GO).5 Selain dari sisi finansial *game online* juga memiliki sisi positif tentunya dengan bimbingan dan pengarahan yang tepat seperti mengajarkan sportivitas dengan nilai-nilai yang dapat dikembangkan sebagai olahraga dan organisasi.

*The Pillars E-sport*s merupakan komunitas *game online* yang telah di bangun oleh musisi Indonesia yaitu Ariel NOAH sebagai *founder* dari komunitas *The Pillars*. *The Pillars* aktif dalam mengikuti ajang turnamen *game online* skala besar maupun membuat *event* turnamen *game online*. Dalam melakukan kegiatan turnamen, *The Pillars* masih menggunakan media penyampaian konvensional yang masih sederhana sehingga untuk pendaftaran turnamen dan layanannya masih belum maksimal dan bahkan mengakibatkan antrian yang Panjang, kehilangan arsip data, dan *update* skor yang seringkali tidak tepat. Dengan masalah yang dihadapi oleh *The Pillars E-sport*, maka dibutuhkan sebuah Sistem Informasi berbasis *web*, hal ini sangatlah penting karena dengan adanya sebuah sistem ini akan dapat membantu kendala yang ada. Pendaftar akan dapat mengakses informasi yang dibutuhkan kapan saja dan dimana saja. Dengan demikian *The Pillars E-sport* dapat memperluas jangkauan komunitas tanpa harus menghabiskan banyak waktu dan biaya.

Oleh karena itu, pada tugas akhir ini berdasarkan permasalahan tersebut akan dikaji dan dilakukan aplikasi pendaftaran *online* berbasis *web* yang dapat diterapkan pada *The Pillars E-sport* dan dituangkan dalam bentuk laporan tugas akhir yang berjudul "**RANCANG BANGUN APLIKASI PENDAFTARAN *ONLINE* TURNAMEN *PLAYER UNKOWN BATTLE GROUND* DI *THE PILLARS E-SPORT* BERBASIS *WEB*".**

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah diuraikan, maka dapat di identifikasi masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana membangun aplikasi *online* berbasis *web* untuk pendaftaran peserta kompetisi.
2. Bagaimana membuat perhitungan skor dan laporan *update* pertandingan yang masih secara konvensional menjadi semi otomatis dan *update* informasi secara akurat.

1.3 Tujuan Penelitian

Mengimplementasikan pendaftaran *online* menggunakan aplikasi berbasis *web*. Serta menjawab indentifikasi masalah diatas yakni:

1. Merancang aplikasi dengan Bahasa pemograman *web* yaitu *PHP* dan *database MYSQL* untuk penerapan pendaftaran *online* berbasis *web*.
2. Merancang sistem untuk perhitungan semi otomatis serta pelaporan yang *update*.

1.4 Batasan Masalah

Dalam pembuatan Tugas Akhir ini penulis membatasi permasalahan sebagai berikut:

1. Aplikasi *web base* ini hanya membantu dalam proses pendaftaran dan perhitungan skor serta pelaporan peserta kompetisi.

2. kriteria yang digunakan pada kompetisi hanya skor saja.

3. aplikasi pendaftaran *online* turnamen ini berbasis *web*.

1.5 Metode Penelitian

Beberapa metode penelitian yang penulis gunakan seperti dibawah:

1.5.1 Teknik Pengumpulan Data

Dalam penelitian ini metode yang digunakan adalah dengan menggunakan metode action program dengan teknik pengumpulan data sebagai berikut:

1. Observasi

   Observasi yaitu pengumpulan data dan informasi yang dilakukan dengan cara mengamati langsung ke objek yang akan diteliti.

2. Wawancara.

   Wawancara yaitu pengumpulan data dengan cara melakukan tanya jawab dengan pihak-pihak terkait.

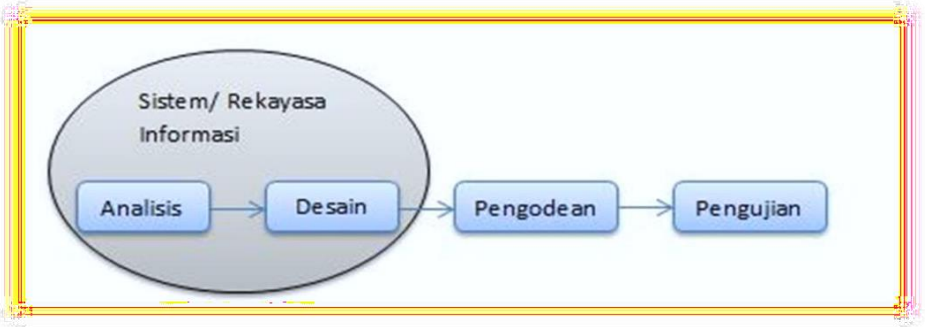
3. Studi perustakaan

   Melakukan pencarian melalui buku di perpustakaan, referensi dari *internet* yang secara langsung terkait dengan permasalahan yang ada.

1.5.2 Metode Pengembangan Perangkat Lunak

Metode untuk merancang sistem yang digunakan penulis dalam penelitian ini yaitu metode *waterfall*. Air terjun (*waterfall*) sering juga disebut model sekuensial linier (sequential linear) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*) (Rosa dan Shalahuddin, 29:2013).

Berikut adalah gambar model air terjun:



Gambar 1.1. Model *Waterfall* (Juniardi Dermawan & Sistem, 2017)

A. Analisis Kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

B. Desain

Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, *representasi* antarmuka, dan prosedur

pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke *representasi* desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

C. Pembuatan Kode Program

Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

D. Pengujian

Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan *(error)* dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

E. Pendukung *(Support)* atau Pemeliharaan *(Maintenance)*

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

1.6 Sistem Penulisan

Sistematika penulisan skripsi ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan.:

BAB I: PENDAHULUAN

Berisi latar belakang, identifikasi dan batasan masalah, manfaat, metodologi penulisan dan sistematika penulisan.

BAB II: LANDASAN TEORI

Pada bab ini memuat landasan teori yang berisi Pustaka yang berhubungan dengan masalah-masalah yang dibahas dalam penelitian.

BAB III: PEMBAHASAN

Pada bab ini menjelaskan mengenai masalah yang di jabarkan, sesuai dengan tujuan penelitian pada BAB 1 dan dengan metode penelitian pada BAB 1 yaitu metode *waterfall.*

BAB IV: IMPLEMENTASI SISTEM

Pada bab ini menjelaskan kesimpulan yang didapatkan setelah melakukan penelitian serta sara-saran.

BAB V: KESIMPULAN DAN SARAN

Pada bab ini menjelaskan kesimpulan yang didapatkan setelah melakukan penelitian serta sara-saran.

BAB II
LANDASAN TEORI

2.1 Rancang Bangun

Menurut Pressman yang dikutip oleh Buchari dkk dalam jurnal E-Journal Teknik Sistem Informasi Vol. 6 No. 1 (2015), rancang merupakan serangkaian prosedur untuk menerjemahkan hasil analisa dari sebuah sistem ke dalam bahasa pemrograman untuk mendeskripsikan dengan detail bagaimana komponen-komponen sistem diimplementasikan.

Menurut Pressman yang dikutip oleh Taufan dalam jurnal E-Journal Teknik Sistem Informasi Vol. 11 No. 1 (2017), "bangun atau pembangunan adalah kegiatan menciptakan sistem baru maupun mengganti atau memperbaiki sistem yang telah ada secara keseluruhan".

2.2 Pengertian Aplikasi *Online*

- **Pengertian Aplikasi**

Aplikasi menurut Dhanta dikutip dari Sanjaya (2015) adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya Microsoft Word, Microsoft Excel. Aplikasi berasal dari kata application yang artinya penerapan lamaran penggunaan.

- **Pengertian *Online***

Menurut Khanna dan Tuti Nurhaeni (2016: 66), *online* adalah jika kita sedang terhubung dengan *internet*, baik itu terhubung dengan akun media sosial kita, email dan berbagai jenis akun lainnya yang kita pakai atau gunakan lewat *internet*. Dengan *internet* kita dapat menerima dan mengakses informasi dalam berbagai format dari seluruh penjuru dunia. Kehadiran *internet* juga dapat memberikan kemudahan dalam dunia pendidikan dan informasi lainnya, hal ini terlihat dengan begitu banyaknya situs *web* yang menyediakan media pembelajaran dan sebuah pelayanan yang semakin mudah.

2.3 Pengertian Pendaftaran

Menurut Depdikbud (2015:1) menyatakan Pendaftaran adalah proses cara perbuatan mendaftar yaitu pencatatan nama, alamat dsb dalam daftar.

Menurut Kamus Besar Bahasa Indonesia (2016:306) menyatakan Pendaftaran adalah proses, cara, pencatatan nama, alamat, dan sebagiannya ke dalam daftar. Dari uraian diatas dapat disimpulkan bahwa pendaftaran adalah proses atau cara yang meliputi, pencatatan nama, alamat dan sebagiannya untuk memenuhi syarat dalam mendaftar.

2.4 Model Pembelajaran *Team Games Tournament* (TGT)

Menurut Saco dalam buku Rusman (2017:22), dalam *Team Games Tournament* (TGT) peserta memainkan permainan dengan anggota-anggota tim lain untuk memperoleh skor bagi tim mereka masing-masing.

Team Games Tournamen (TGT) adalah salah satu tipe pembelajaran kooperatir yang menempatkan peserta dalam kelompok-kelompok yang beranggotakan 5-6 peserta yang memiliki kemampuan, jenis kelamin dan suku kata atau kata yang berbeda. Ini dapat digunakan dalam berbagai macam mata pelajaran, ilmu-ilmu heksaks, ilmu social hingga *professional*.

2.4.1 Langkah-langkah *Team Games Tournament* (TGT)

Secara runut implementasi *Team Games Tournament* (TGT) terdiri dari empat komponen utama:

1. Presentasi Panitia

2. Peserta

3. Turnamen

4. Pengenalan Kelompok

Adapun Langkah-langkahnya sebagai berikut. Pada model (TGT) peserta ditempatkan dalam tim beranggotakan empat orang yang merupakan campuran jenis kelamin, suku dan skill. Panitia menyiapkan arena pertandingan *online* dan kemudian peserta di dalam tim mereka memastikan bahwa seluruh anggota tim telah menguasai aturan pertandingan.

2.4.2 Aturan (Skenario) Permainan

Dalam permainan terdiri dari beberapa grup, misal grup a, grup b, grup c dan Seterus nya sejumlah total jumlah peserta, dan masing masing grup berjumlah 5 orang. Ada 3 tahap turnamen yaitu:

1. Tahap kualifikasi, pada tahap ini masing-masing grup akan melakukan pertandingan satu kali *match* kualifikasi, delapat tim dengan poin tertinggi akan lolos ke tahap semifinal.

2. Tahap semifinal, delapan tim dari masing-masing grup yang lolos ke babak semi final akan di kelompokan Kembali menjadi 2 grup untuk dilakukan pertandingan babak semifinal. Delapan tim dengan skor tertinggi akan masuk ke tahap final.

3. Tahap Final, tim dengan jumlah poin tertinggi menjadi juara dari turnamen.

2.4.3 Sistem Perhitungan Skor Turnamen

Adapun perhitungan skor yang akan di berlakukan pada turnamen ini yaitu:

- Jumlah total tim yang akan ikut dalam 1 pertandingan adalah 16 tim.
- Tim yang mati pertama (#16) sampai dengan ke 13 tidak akan mendapatkan poin placement.
- Tim yang mati pada posisi ke 12 sampai posisi ke 8 akan mendapatkan 1 poin.
- Tim yang mati pada posisi 7 sampai posisi 2 mendapatkan poin placement kelipatan 2 (2, 4, 6, 8, 10, 12) sesuai urutan tim yang mati dengan posisi terendah.
- Tim akan mendapatkan poin tambahan jika melakukan kill pada musuh. 1 kill = 1 poin.

2.4.4 Pendokumentasian Turnamen

A. Tabel List Tahap Kualifikasi

Form informasi tim yang berpartisipasi dan akan bertanding pada babak kualifikasi untuk maju ke semifinal, hasil dari pertandingan, empat tim dengan total jumlah poin tertinggi maju ke babak semifinal (Misal: grup yang lolos adalah dari masing masing grup A, B, C, D adalah Tim no urut 1 sampai Tim no urut 8).

Tabel 2.1 Tahap Kualifiasi

No	Grup A	*Match* 1	*Match* 2	Total Poin
1	Tim 1			
2	Tim 2			
3	Tim 3			
4	Tim 4			

B. Tabel List Tahap Semifinal

Delapan tim yang lolos dari fase kulaifikasi akan di acak dan di bagi menjadi dua grup. Delapan tim dengan poin tertinggi akan masuk ke babak final (Misal: grup yang lolos adalah dari masing masing grup A, B, C, D adalah Tim no urut 1 sampai Tim no urut 8).

Tabel 2.2 Tahap Semifinal

No	Grup A	Poin *Match* 1	Poin *Match* 2	Total Poin
1	Tim 1			
2	Tim 2			
3	Tim 3			
4	Tim 4			

3. Tabel List Tahap Final

Babak penentuan juara. Biasanya pada babak final total *match* yang di mainkan antara 4–6 *match*. Tim dengan poin tertinggi menjadi juaranya.

Tabel 2.3 Tahap Final

No	Grup A	Poin *Match* 1	Poin *Match* 2	Poin *Match* 3	Poin *Match* 4	Total Poin
1	Tim 1					
2	Tim 2					
3	Tim 3					
4	Tim 4					

2.5 *Waterfall* Model

Menurut Pressman (2015:42) *Waterfall Model* merupakan dasar dari aktivitas proses yang terdiri dari spesifikasi, pengembangan, validasi, evolusi dan

semua di*representasi* kan dalam tahaan proses yang terpisah seperti spesifikasi kebutuhan, perancangan perangkat lunak, implementasi, pengujian dan sebagainya.

Model dari *software development proses* ini adalah model yang pertama kali dipublikasikan yang diperoleh dari *system engineering process* yang umum. Karena satu tahap ke tahap lainnya mengalir kebawah. Model ini disebut sebagai *Waterfall Model*. Secara prinsip semua aktivitas pross harus direncanakan dan diproses terlebih dahulu sebelum mulai mengerjakannya.

Tahapan dari *Waterfall Model* merefleksikan pokok-pokok dari aktivitas pengembangan:

1. *Requirements Analysis and Definition*

   Layanan yang diberikan oleh sistem**,** batasan sistem, dan tujuan ditetapkan setelah melakukan konsultasi dengan pengguna sistem. Semua didefinisikan secara rinci dan dibuat sebagai spesifikasi dari sistem.

2. *System and Software Design*

   Proses perancangan sistem menyediakan kebutuhan hardware atau *software* dengan menyediakan ar*site*ktur dari keseluruhan sistem. Perancangan sistem melibatkan pengidentifikasian dan penjelasan dari abstraksi sistem dan hubungan nya.

3. *Implementation and Unit Testing*

Pada tahap ini, perancangan sistem direalisasikan menjadi sebuah program atau unit program. Pengujian untuk melibatkan pengidentifikasian dan penjelasan dari abstraksi sistem dan hubungannya.

4. *Integration and Systemm Testing*

   Setiap unit program dan program-program yang sudah ada diintegrasikn dan diuji sebagai satu keutuhan sistem untuk memastikan apakah kebutuhan sistem sudah terpenuhi. Setelah melakukan pengujian, sistem baru disebarkan ke pengguna.

5. *Operation and Maintenance*

   Dilakukan instalasi terhadap sistem dan digunakan dalam prakteknya. *Maintenance* melibatkan koreksi terhadap *error* yang tidak ditemukan pada tahap sebelumnya, memperbaiki implementasi dari unit sistem dan meningkatkan layanan yang diberikan oleh sistem sebagai kebutuhan baru yang ditemukan.

Waterfall Model digunakan hanya jika semua kebutuhan sudah dimengerti dan tidak berubah secara radikal pada tahap pengembangan

2.6 *Unified Modeling Language* (UML)

UML adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan desain, serta menggambarkan ar*site*ktur dalam pemrograman berorientasi objek (Rosa dan Shalahuddin, 2018:137).

UML merupakan bahasa standar pemodelan untuk perangkat lnak dan pengembangan sistem. Merancang sebuah desain untuk sistem yang besar

merupakan hal yang sulit. Dari aplikasi *desktop* yang sederhana sampai sistem *multi-tier* dapat dibangun dari ratusan, bahkan ribuan, komponen perangkat lunak dan perangkat keras.

Dalam merancang sebuah sistem, mengatur kompleksitas adalah satu alas an utama mengapa harus membuat model. Pemodelan membantu para pengembang untuk dapat focus, dapat mendokumentasikan, menangkap keseluruhan sistem dan mengkomunikasikan aspek-aspek penting dalam sistem yang sedang di rancang.

UML tepat digunakan untuk memodelkan sistem dari mulai memodelkan informasi sistem untuk perusahaan hinggap aplikasi *web*, bahkan untuk sistem yang rumit sekalipun. UML menggunakan *class* dan *operation* dalam konsep dasarnya.

2.7 *Use case Diagram*

Use case Diagram merupakan pemodelan untuk melakukan (*behavior*) Sistem Informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah Sistem Informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

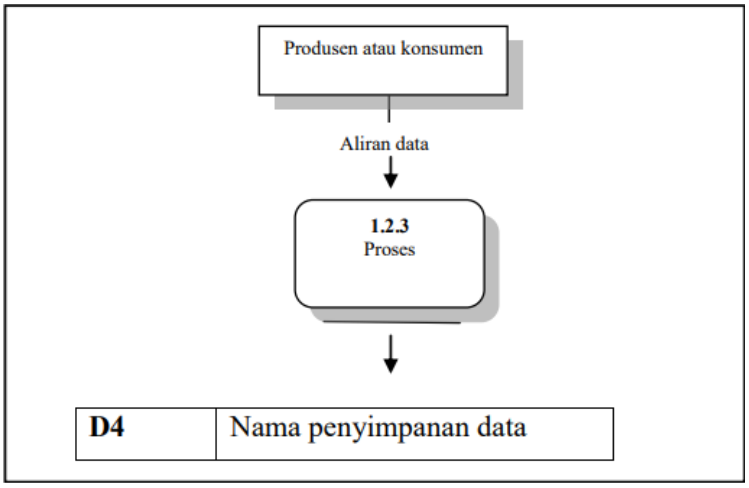
Berikut adalah simbol-simbol yang ada pada diagram *use case* (Rosa dan Shalahuddin, 2015:156)

Tabel 2.4. Simbol *Use case* (Rosa dan Shalahuddin, 2015:156)

Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling tertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal *frase* nama *Use case*
AKtor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan Sistem Informasi yang akan dibuat diluar Sistem Informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal *frase* nama aktor.
Asosiasi / *assosiation* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi / *extend* 	Relasi *use case* tambahan kesebuah *use case* dinamakan *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inherence* pada pemograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

2.8 *Data Flow Diagram*

Sumanto dan Shalahuddin (2015:69) yang dimaksud dengan *Data Flow Diagram* awalnya dikembangkan oleh Chris Gane dan Trish Sarson pada tahun 1979 yang termasuk dalam *Structured System Analysis and Design Methodology* (SSADM) yang ditulis oleh Chris Gane dan Trish Sarson. Sistem yang dikembangkan ini berbasis pada dekomposisi fungsional dari sebuah sistem. Berikut adalah contoh *Data Flow Diagram* yang dikembangkan oleh Chris Gane dan Trish Sarson:



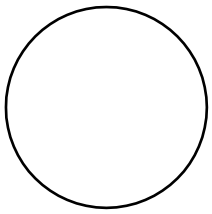
Gambar 2.1. Contoh DFD yang dikembangkan Chris Gane dan Trish Sarson

Edward Yourdon dan Tom DeMarco memperkenalkan metode yang lain pada tahun 1980-an dimana mengubah persegi dengan sudut lengkung (pada DFD Chris dan Tris Sarson) dengan lingkaran untuk menotasikan.

DFD Edward Yourdon dan Tom DeMarco populer digunakan sebagai model analisis sistem perangkat lunak untuk sistem perangkat lunak yang akan diimplementasikan dengan pemograman terstruktur.

Sukamto dan Shalahuddin (2015:71) notasi-notasi pada DFD (Edward Yourdon dan Tom DeMarco) adalah sebagai berikut :

Tabel 2.5. Notasi DFD (Edward Yourdon dan Tom DeMarco)

No	Notasi	Keterangan
1.		Proses atau fungsi atau prosedur pada perangkat lunak yang akan diimplementasikan dengan pemograman terstruktur, maka pemodelan notasi inilah yang harusnya menjadi fungsi atau prosedur didalam kode program. Catatan: Nama yang diberikan pada sebuah proses biasanya berupa kata kerja.
2.		*File* atau *basis data* atau penyimpanan (*storage*), pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemograman terstruktur, maka pemodelan notasi inilah yang harusnya dibuat menjadi tabel-tabel *basis data* yang dibutuhkan, tabel-tabel ini juga harus sesuai dengan perancangan tabel-tabel pada basis data ERD, *Conceptual Data Model* (CDM), *Physical Data Model* (PDF).
3		Entitas luar (*External Entity*) atau masukan

		(*Input*) atau keluaran (*output*) atau orang yang memakai/berinteraksi dengan perangkat lunak yang dimodelkan atau sistem lain yang terkait dengan aliran data dari sistem yang dimodelkan. Catatan: Nama yang digunakan pada masukan (*input*) atau keluaran (*output*) biasanya berupa kata benda.
4.		Aliran data, merupakan data yang dikirim antar proses, dari penyimpanan ke proses, atau dari proses ke masukan (*input*) atau keluaran (*output*). Catatan: Nama yang digunakan pada aliran data biasanya berupa kata benda, dapat diawali dengan kata data missal "data *peserta*".

Berikut ini adalah tahapan-tahapan perancangan dengan menggunakan *Diagram Flow Diagram* (DFD):

6. Membuat *DFD Level 0* atau sering disebut juga *Context Diagram*

DFD Level 0 menggambarkan sistem yang akan dibuat sebagai suatu entitas tunggal yang berinteraksi dengan orang maupun sistem yang lain. *DFD Level 0* digunakan untuk menggambarkan interaksi antara sistem yang akan dikembangkan dengan entitas luar.

7. Membuat *DFD Level 1*

DFD Level 1 digunakan untuk menggambarkan modul-modul yang ada dalam sistem yang akan dikembangkan. *DFD Level 1* merupakan hasil *breakdown DFD Level 0* yang sebeleumnya sudah dibuat.

8. Membuat *DFD Level 2*

Modul-modul pada *DFD Level 1* dapat di *breakdown* menjadi *DFD Level 2*. Modul mana saja yang harus di *breakdown* lebih detail tergantung pada tingkat kedetailan modul tersebut. Apabila modul tersebut sudah cukup detail dan rinci maka modul tersebut sudah tidak perlu di *breakdown* lagi. Untuk sebuah sistem, jumlah *DFD Level 2* sama dengan jumlah modul pada *DFD Level 1* yang di *breakdown.*

9. Membuat *DFD Level 3* dan seterusnya

DFD Level 3,4,5 dan seterusnya merupakan *breakdown* dari modul pada *DFD Level* diatasnya. *Breakdwon* pada level 3,4,5 dan seterusnya aturannya sama percic dengan *DFD Level 1* atau *Level 2*.

Pada satu diagram *DFD* sebaiknya jumlah modul tidak boleh lebih dari 20 buah. Jika lebih dari 20 buah modul, diagram akan terlihat rumit dan susah untuk dibaca sehingga menyebabkan sistem yang dikembangkan juga menjadi rumit.

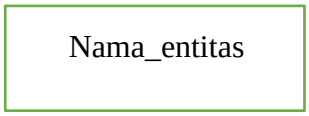
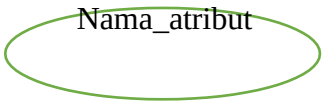
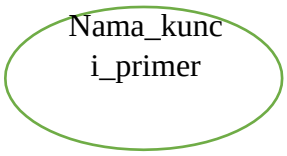
2.9 *Entity Relationship Diagram* (ERD)

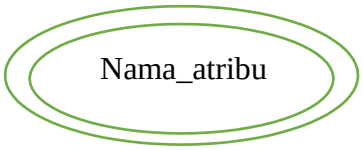
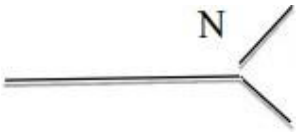
Menurut Sukamto dan Shalahuddin (2015:289) "*Entity Relationship Diagram (ERD)* adalah pemodelan awal basis data yang akan dikembangkan

berdasarkan teori himpunan dalam bidang matematika untuk pemodelan basis data relasional".

ERD memiliki beberapa aliran ntasi seperti notasi Chen (dikembangkan oleh Peter Chen). Barket (dikembangkan oleh Richard Barket, Ian Palmen, Harry Ellis), notasi Crow's Foot, dan beberapa notasi lain. Namun yang banyak digunakan adalah notasi dari Chen. Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi Chen:

Tabel 2.6. Simbol ERD (Sukamto dan shalahuddin,2015)

Simbol	Deskripsi
Entitas / *Entity* 	Entitas merupakan data inti yang akan disimpan, bakal tabel pada basis data benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer, penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel.
Atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
Atribut Kunci Primer 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan, biasanya berupa *id*, kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).

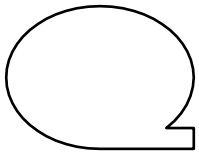
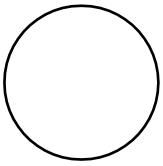
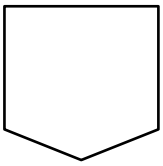
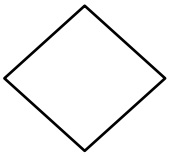
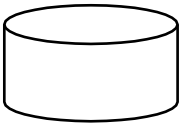
Atribut mulinilai/multivalue 	*Field* atau kolom data yang butuh disimpan dalam entitas yang dapat memiliki nilai dari satu.
Relasi 	Relasi yang menghubungkan antar entitas, biasanya diawali dengan kata kerja.
Asosiasi 	Penghubung antar relasi dan entitas dimana dikedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan atara entitas satu dengan entitas yang lain disebut dengan kardinalitas, misalkan ada kardinalitas 1 ke N atau sering disebut dengan *one to many* menghubungkan entitas A dan entitas B.

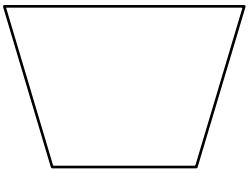
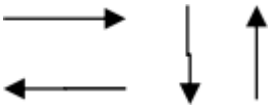
2.10 *Flow Chart*

Menurut Indrajani (2015:36) yang dimaksud *Flow Chart* adalah penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program. Simbol-simbol dalam *Flow Chart* adalah sebagai berikut:

Tabel 2.7. Simbol *Flow Chart* (Indrajani, 2015:36)

No	Simbol	Keterangan
1.		Simbol *Start* atau *End* yang mengidentifikasikan awal atau akhir dari sebuah *flowchart.*

2.		Simbil pemrosesan yang terjadi pada sebuah alur kerja.
3.		Simbol yang menyatakan bagian dari program (sub program).
4.		Simbol masukan atau keluaran dari atau kesebuah pita *magnetic*.
5.		Simbol *Input/Output* yang mendefinisikan masukan dan keluaran proses.
6.		Simbol konektor untuk menyambung proses pada lembar kerja yang sama.
7.		Simbol konektor untuk menyambung proses pada lembar kerja yang berbeda.
8.		Simbol masukan atau keluaran dari atau ke sebuah dokumen.
9.		Simbol untuk memutuskan proses lanjutan dari kondisi tertentu.
10.		Simbol *database* atau basis data.

11		Simbol yang menyatakan piranti keluaran, seperti layar monitor, *printer*, dll.
12		Simbol yang mendefinisikan proses yang dilakukan secara manual.
13		Simbol untuk menghubungkan antar proses atau antar simbol.

2.11 *Website*

Menurut Bekti (2015:35) menyimpulkan bahwa: *Website* merupakan kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing masing dihubungkan dengan jaringan-jaringan halaman.

Sebuah situs *web* biasanya ditempatkan setidaknya pada sebuah *server web* yang dapat diakses melalui jaringan seperti *internet*, ataupun jaringan lokal (*LAN*) melalui alamat *internet* yang dikenali sebagai *URL*. Gabungan atas semua

situs yang dapat diakses publik di *internet* disebut pula sebagai *Waring Wera Wanua* atau lebih dikenal dengan singkatan *WWW*.

Meskipun setidaknya halaman beranda situs *internet* umumnya dapat diakses publik secara bebas, pada prakteknya tidak semua situs memberikan kebebasan bagi public untuk mengaksesnya, beberapa situs *web* mewajibkan pengunjung untuk melakukan pendaftaran sebagai anggota, atau bahkan meminta pembayaran untuk dapat menjadi anggota untuk dapat mengakses isi yang terdapat dalam situs *web* tersebut. Misalnya situs-situs yang menampilkan berita, layanan *email* dll. Pembatasan-pembatasan ini umumnya dilakukan karena alasan keamanan, menghormati privasi atau karena tujuan komersial tertentu.

2.12 *Web Server*

Menurut Supono dan Putratama (2016:6) "*Web- Server* adalah perangkat lunak *server* yang berfungsi untuk menerima permintaan dalam bentuk situs *web* melalui *HTTP* atau *HTTPS* dari klien itu, yang dikenal sebagai browser *web* dan mengirimkan kembali (reaksi) hasil dalam bentuk situs yang biasanya merupakan dokumen *HTML*.".

Fungsi utama *web server* adalah untuk melakukan atau mentransfer berkas permintaan pengguna melalui protokol komunikasi yang telah ditentukan sedemikian rupa. Halaman *web* yang diminta terdiri dari berkas teks, video, gambar, file dan banyak lagi. Pemanfaatan *web server* berfungsi untuk mentrasfer

seluru aspek pemberkasan dalam sebuah halaman *web* termasuk yang di dalam berupa teks, video, gambar dan banyak lagi.

Salah satu contoh dari *web server* adalah *Apache* (*Apache web server*) merupakan *web server* yang paling banyak dipergunakan di *internet*. Program ini pertama kali didesain untuk sistem operasi lingkungan UNIX. *Apache* mempunyai progam pendukung yang cukup banyak. Hal ini memberikan layanan yang cukup lengkap bagi penggunanya.

Saat ini umumnya *server web* telah dilengkapi pula dengan mesin penerjemah bahasa skrip yang memungkinkan *server web* menyediakan layanan situs *web* dinamis dengan memanfaatkan pustaka tambahan seperti *PHP, ASP*

2.13 *HTML*

Menurut Sidik dan Husni (2017:10) "*HTML* kependekan dari Hyper*link* Text Markup Language. Dokumen *HTML* adalah file teks murni yang dapat dibuat dengan editor teks sembarang. Dokumen ini dikenal sebagai *web* page. Dokumen *HTML* merupakan dokumen yang disajikan dalam browser *web* surfer. Dokumen ini umumnya berisi informasi atau interface aplikasi di dalam *internet*".

2.14 *PHP*

Menurut Hikmah, dkk (2015:1)"*PHP* merupakan kependekan dari Hypertext Preprocessor. *PHP* tergolong sebagai perangkat lunak *open source* yang diatur dalam aturan *general purpose licences* (GPL). Bahasa pemograman *PHP* sangat cocok dikembangkan dalam lingkungan *web*, karena *PHP* bisa diletakkan pada script *HTML* atau sebaliknya. *PHP* dikhususkan untuk pengembangan *web* dinamis".

Kelebihan *PHP* dari bahasa pemograman lain yaitu:

10. Bahasa pemograman *PHP* adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaanya.
11. Dalam sisi pengembangan lebih mudah, karena banyaknya *developer* yang siap membatu dalam pengembangannya.
12. Dalam sisi pemahamannya. *PHP* adalah bahasa *scripting* yang paling mudah karena memiliki referensi yang cukup banyak.
13. *PHP* adalah banasa *open source* yang dapat digunakan di berbagai mesin (*Linux, Unix, Macintosh, Windows*).

2.14.1 *PHP MyAdmin*

PHPMyAdmin adalah perangkat lunak bebas yang ditulis dalam basa pemograman *PHP* yang digunakan untuk menangani *Administrator*istrasi *MYSQL* melalui jaringan *WWW*. *PHPMyAdmin* mendkungn berbagai operasi *MYSQL*,

diantaranya mengelola basis data, tabel-tabel, bidang (*fields*), relasi(*relation*), *indeks*, pengguna (*users*), perijinan(*permissions*) dan lain-lain.

Pada dasarnya, mengelola basis data dengan *MYSQL* harus dilakukan dengan cara mengetikkan baris-baris perintah yang sesuai (*command line*) untuk setiap maksud tertentu. Jika seseorang ingin membuat basis data (*database*), ketikkan baris perintah yang sesuai untuk membuat basis data. jika seseorang menghapus tabel, ketikkan baris perintah yang sesuai untuk menghapus tabel. Hal tersebut tentu saja sangat menyulitkan karena seseorang harus hafal dan mengetikkan perintahnya satu persatu.

Saat ini banyak sekali perangkat lunak yang dapat dimanfaatkan untuk mengelola basis data dalam *MYSQL*, salah satunya adalah *PHPMyAdmin*. Dengan *PHPMyAdmin* seseorang dapat membuat *database,* membuat tabel, mengisi data, dan lain-lain dengan mudah tanpa harus menghafal baris perintahnya.

Adapun fasilitas dari *PHPMyAdmin* yaitu:

- Membuat dan menghapus *database*
- Dapat membuat *FDF* grafik, dan mampu mencari data dalam *database*
- Membuat, menyalin, menghapus, dapat menambah *field*
- Manajemen pengguna dan *privilege* (hak akses) pada *MYSQL*

2.15 *MYSQL*

MYSQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi *GPL* (*General PublicLicense*). Setiap pengguna dapat secara bebas menggunakan *MYSQL*, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. *MYSQL* sebebarnya merupakan turunan salah satu konsep utama basis data yang telah ada sebelumnya yakni SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoprasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoprasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem *basis data* (*DBMS*) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai “peladen” basis data. *MYSQL* mendukung operasi *basis data* transaksional maupun operasi *basis data* non-transaksional. Pada modus operasi non-transaksional, *MYSQL* dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak “peladen” basis data kompetitor lainnya. Namun demikian pada modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi *blogging* berbasis *web* (*wordpress*), *CMS,* dan sejenisnya. Untuk kebutuhan sistem yang ditunjukan untuk bisnis sangat disarankan untuk menggunakan modus basis data transaksional, hanya saja sebagia konsekuensinya unjuk kerja *MYSQL* pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

2.15.1 *Apache*

Server HTPP *Apache* atau *Server Web Apache* adalah *server web* yang dapat dijalankan di banyak sistem operasi (Unix, BSD, Linux, Microsoft Windows serta platform lainnya) yang berguna untuk melayani dan memfungsikan situs *web*. Protokol yang digunakan untuk melayani fasilitas *web/www* ini menggunakan *HTTP*.

Apache memiliki fitur-fitur canggih seperti pesan kesalahan yang dapat dikonfigurasi, autentikasi berbasis *basis data* dan lain-lain. *Apache* juga didukung oleh sejumlah antarmuka pengguna berbasis grafik (*GUI*) yang memungkinkan penanganan *server* menjadi mudah.

Apache merupakan perangkat lunak *open source* dikembangkan oleh komunitas terbuka yang terdiri dari pengembang-pengembang dibawah naungan *Apache Software Foundation.*

2.16 *OOP (Object Oriented Programming)*

Menurut Rosa dan Shalahudin dalam (Mandiri & Octasia, 2016) “metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya”.

Dalam OOP, *class* merupakan sekumpulan objek yang memiliki atribut-atribut dan *method. Class* merupakan deskripsi dari satu atau lebih objek yang memiliki kesamaan atribut, layanan, metode, hubungan dan semantik, termasuk deskripsi cara membuat objek baru dalam *class.* Ada juga yang disebut dengan *super class*, sebuah *class* induk yang nantinya mempunyai *class-class* yang terdiri dari *class* dan *subclass.*

Objek dalam OOP adalah sebuah benda atau unit atau sifat kerja yang memiliki atribut-atribut. Objek adalah sebuah abstraksi dari sesuatu pada domain masalah, menggambarkan kemampuan untuk menyimpan informasi mengenai hal tersebut, berinteraksi dengan hal tersebut atau keduanya.

Abstraksi *procedural* dalam OOP disebut dengan operasi yang menspelsifikasi tipe dari perilaku dan terdiri dari fungsi-fungsi.

Istilah lain terdapat *encapsulation/*pengkapsulan, yang merupakan pembatasan ruang lingkul program terhadap data yang diproses supaya data terlindungi oleh prosedur atau objek lain, kecuali prosedur yang berada di objek itu sendiri.

Polymorphism adalah konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku yang berbeda, bahwa operasi yang sama mungkin memiliki perbedaan dalam *class* yang berbeda. Pada OOP terdapat juga yang disebut dengan *inheritance* (Perwarisan), yaitu kepemilikan yang bersifat *implicit* dari fitur *subclass* yang didefinisikan dalan *superclass* fitur tersebut mencakup *variables* dan *method.*

Konsep dasar dari Pemograman Berorientasi Objek menekankan konsep sebagai berikut:

a. Kelas: kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemgrograman berorientasi objek. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan *domain* permasalahan yang ada, dan kode yang terdapat dalam sebuah *class*, sebaiknya (relative) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.
b. Objek: membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer, objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
c. Abstraksi: kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya. Yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani segala model dari

"perilaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

d. Enkapsulasi: memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak, hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada *representasi* dalam objek tersebut.

e. *Polimorfisme* melalui pengiriman pesan, tidak tergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan, metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu dimana pesan tersebut dikirim, contohnya, bia sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut

polimorfisme karena sebuah variable tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai *polimorfisme* melalui pengguna fungsi kelas pertama.

f. dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki *manager*, sekretaris, petugas Administratoristrasi data dan lainnya. Missal *manager* tersebut ingin memperoleh data dari bagian Administratoristrasi maka *manager* tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian Administratoristrasi untuk mengambilnya. Pada kasus tersebut seorang *manager* tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi *manager* bisa mendapatkan data tersebut melalui objek petugas Administratoristrasi. Jadi untuk menyelesaikan suatu masalah dengan kolabrasi antar objek- objek yang ada karena setiap objek memiliki desktripsi tugasnya sendiri.

.

BAB III
PEMBAHASAN

3.1 Analisis Sistem

Mengenal masalah merupakan langkah pertama yang dilakukan dalam tahap analisa sistem. Masalah (*problem*) dapat didefinisikan sebagai suatu pertanyaan yang harus dipecahkan. Oleh karena itu pada tahap analisa sistem, langkah pertama yang dilakukan adalah mengidentifikasi masalah yang terjadi.

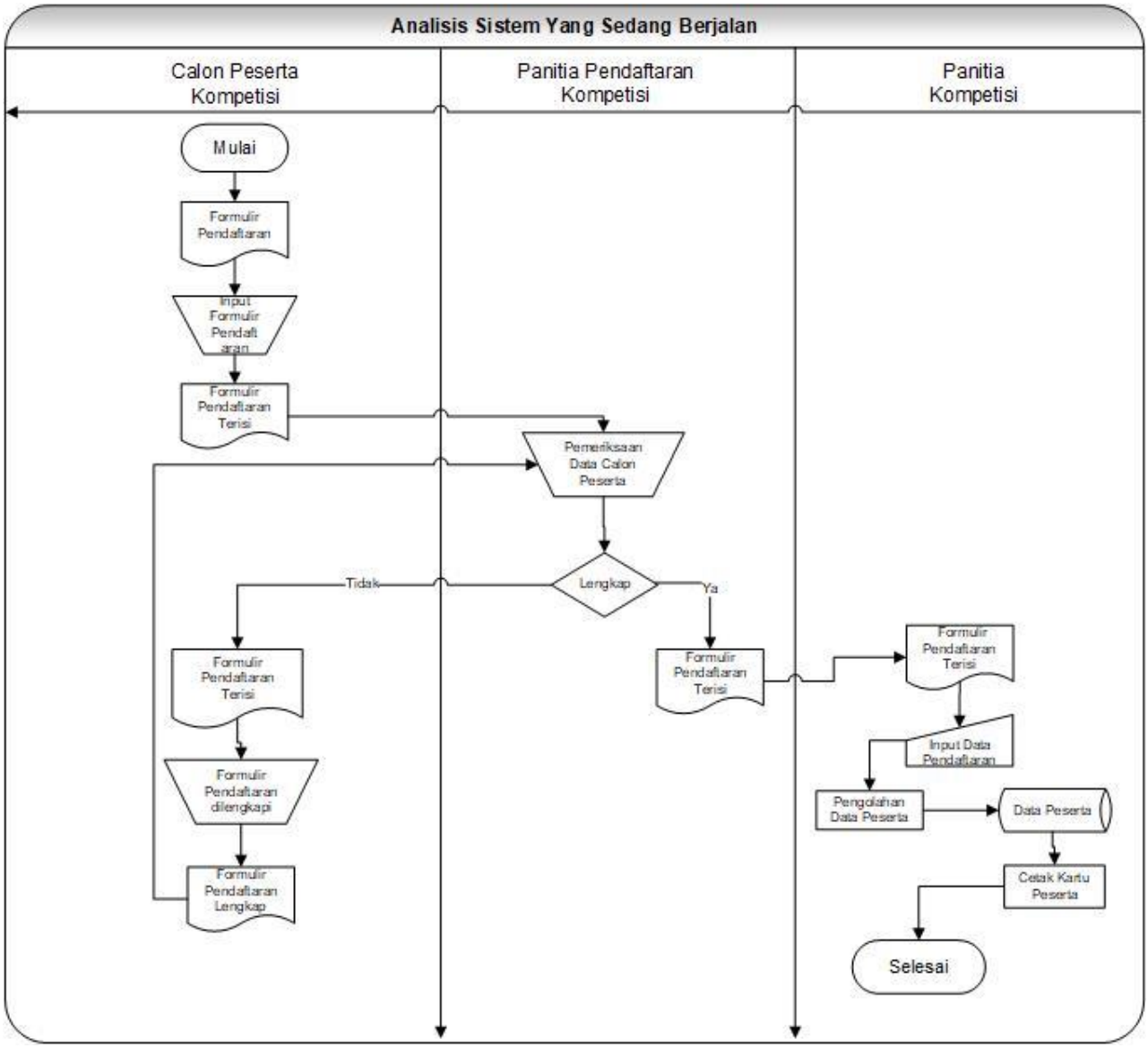
Mengidentifikasi masalah dimulai dengan mengkaji subyek permasalahan yang ada. Adapun analisis yang dilakukan yakni:

3.1.1 Analisis sistem yang sedang berjalan

Berdasarkan hasil analisis yang dilakukan dengan metode studi literature dan observasi saat ini masih sederhana yang hanya ditulis tangan dan hasilnya diarsipkan kedalam tempat penyimpanan arsip. Beberapa masalah tersebut dapat diuraikan sebagai berikut:

1. Pengumuman Turnamen dilakukan Via *Whatsapp* grup / *Facebook Fanspage.*
2. Pendaftaran kompetisi dilakukan secara manual atau *offline* dengan datang ke kantor.
3. Para calon peserta kompetisi mendaftar dengan mengisi form pendaftaran kompetisi.

4. Setelah formulir pendaftaran kompetisi di isi maka di kembalikan kepada panitia pendaftaran kompetisi untuk selanjutnya diserahkan ke panitia kompetisi.
5. Untuk hasil yang lolos verifikasi dari tim Panitia kompetisi Turnamen di umumkan lewat Grup Khusus *Whatsapp* 1 minggu setelah pendaftaran kompetisi sesuai kesepakatan.
6. Tim yang lolos pendaftaran kompetisi kemudian di bagi menjadi beberapa Grup oleh Panitia Turnamen.
7. Tim yang lolos juga mendapatkan
8. Pembagian tim dan skoring masih manual sehingga tidak tersampaikan dengan real time. Dan pada akhir nya data tersebut di arsipkan manual kedalam rak rak penyimpanan data yang telah disediakan, faktor ini berakibat data tercecer atau memakan waktu lama dalam mencari file yang di butuhkan.



Gambar 3.1 *Flow Map* sistem yang sedang berjalan

3.1.2 Analisis Prosedur Perhitungan Skor

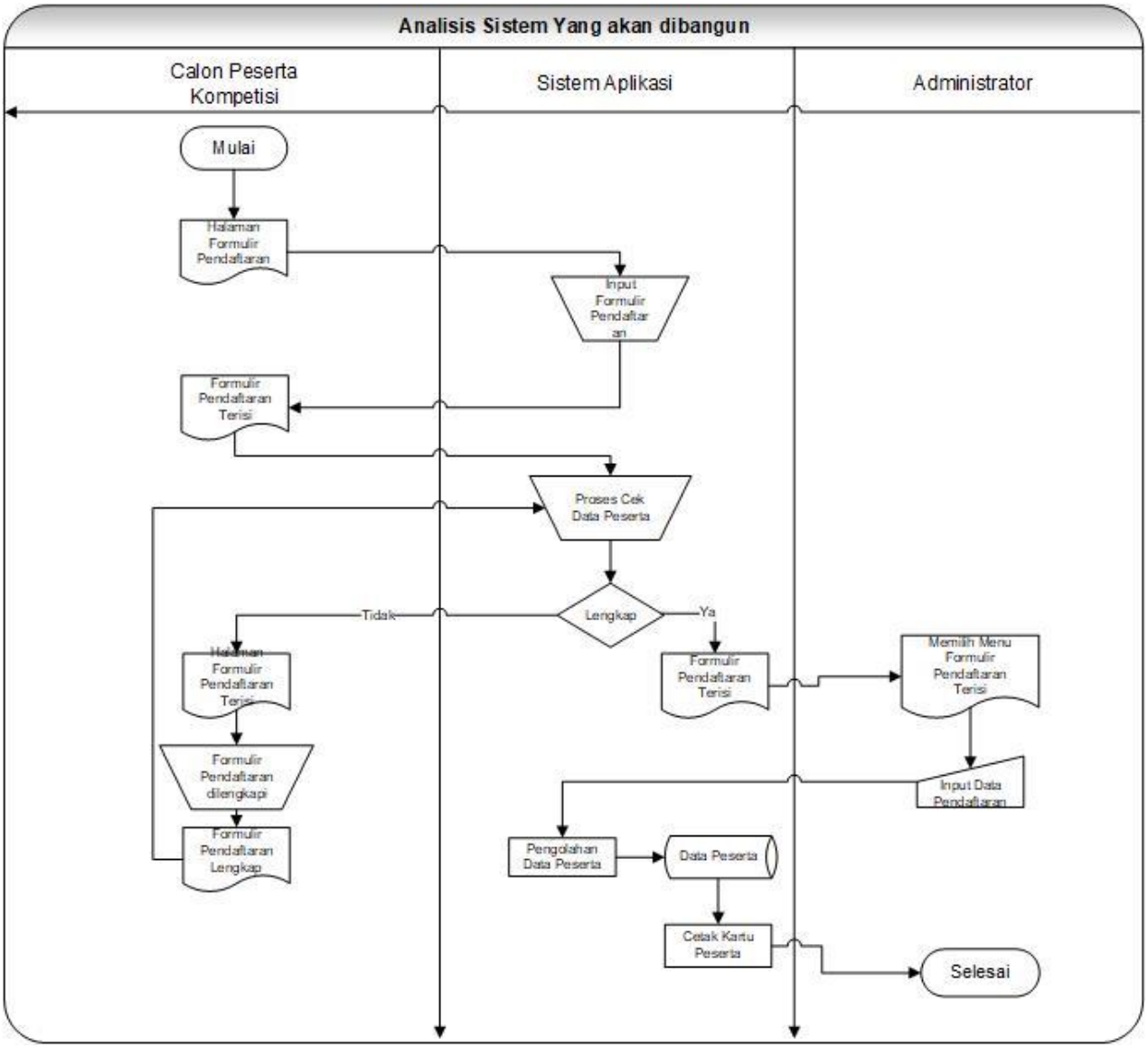
Prosedur perhitungan skor pada kompetisi dilakukan masih manual terdapat beberapa fase yakni fase penyisihan grup, fase semifinal dan fase final adapun analisis di lapangan yakni sebagai berikut:

1. Pada fase penyisihan grup Peserta kompetisi dari beberapa tim di bagi menjadi beberapa grup dan di catat oleh panitia pelaksana dan dicatat di formulir grup.
2. Panitia mencatat skor sementara hasil pertandingan seluruh tim yang bermain pada dokumen skor sementara.
3. Pemenang setiap tim pada fase penyisihan grup akan di catat pada form pemenang penyisihan masing-masing grup.
4. Peserta kompetisi yang lolos ke babak selanjut nya akan masuk kedalam fase semifinal dan untuk skor yang di tetapkan yaitu pemenang dengan skor tertinggi.
5. Peserta kompetisi yang lolos ke babak final akan di catat pada form final turnamen dan di jadwalkan untuk pertandingan final.
6. Pertandingan final dilakukan sesuai data yang tercantum di data panitia panyelenggara dan pemenang di tentukan dengan skor tertinggi. Kemudian hasil dari pertandingan di simpan di form pemenang turnamen dan di dokumentasikan.

3.1.3 Analisis kebutuhan fungsional

Pada subbab ini akan menerangkan tentang kebutuhan fungsional diantaranya analisis fungsi masukan, menyusun dan menilai peserta kompetisi sesuai standar nilai hasil kesepakatan panitia.

3.2 Analisis Sistem yang Akan di Bangun



Gambar 3.2 Sistem yang Akan di Bangun

Dari *Diagram* 3.2 Sistem yang akan di bangun dapat di deskripsikan sebagai berikut:

1. *Administrator* dan Peserta yang telah terdaftar pada *database* yang dapat *Login* kedalam sistem aplikasi.
2. *Administrator* dan Peserta dapat mengakses aplikasi pada banyak perangkat contoh: *Smartphone* dan *Web browser online*.
3. Aplikasi yang telah dibuat telah di publish *online* sehingga dapat diakses dimanapun dan *uptodate*.
4. Aplikasi Turnamen disimpan dalam sebuah *Server online* baik itu *hosting* dan domain yang telah di sewa.
5. Data yang telah terproses seluruhnya disimpan kedalam *database* turnamen.
6. Peserta *History*

Peserta	Kebutuhan Sistem
Administrator	Mengelola data seperti data peserta kompetisi, data kompetisi, Peserta, metode pembayaran, galeri dan store pada aplikasi *website* berbasis *online* untuk turmamen ini
Peserta	Registrasi pendaftaran kompetisi calon peserta kompetisi turnamen, mengelola profile, daftar turnamen dan melakukan pembayaran pada aplikasi *website* berbasis *online* untuk turnamen ini.

Tabel.3.1 Peserta *History*

3.2.1 Perancangan Fungsi Masukan

Acuan dalam membangun Aplikasi pendaftaran kompetisi *online* ini berdasarkan formulir peserta kompetisi *The Pillars E-sport* dan standarisasi nilai yang sedang berjalan. Maka setiap kriteria di formulir pendaftaran kompetisi peserta kompetisi nanti nya akan diberikan suatu bobot dan dihitung yang nantinya akan dimasukan kedalaman aplikasi untuk kompetisi. Sedangkan standarisasi nilai digunakan untuk menghasilkan juara pada kompetisi tersebut.

3.2.2 Analisis Pendaftaran kompetisi Peserta kompetisi

Hal-hal yang harus diperhatikan dalam menyusun pedoman pendaftaran kompetisi peserta kompetisi yakni mencakup pada data yang valid untuk peserta kompetisi serta mencantumkan beberapa kolom isian dimana nanti akan dijadikan data peserta kompetisi.

Berdasarkan wawancara yang dilakukan di lapangan hasilnya adalah sebagai berikut:

1. Calon peserta kompetisi wajib 5 orang 1 orang *manager* Tim
2. Calon peserta kompetisi wajib mempunyai KTP / di atas 17 tahun
3. Sehat Jasmani dan Rohani
4. Mampu memainkan Game dan bekerja sama

3.2.3 Analisis Metode Pembayaran

Pada poin ini akan di jelas kan mengenai metode pembayaran untuk biaya registrasi calon peserta kompetisi turnamen, pembayaran transaksi non-tunai menjadi pilihan metode pembayaran lebih mudah dan aman akan tetapi perlu regulasi yang jelas agar dapat memudahkan calon peserta kompetisi dalam melakukan pembayaran registrasi ini yakni:

1. Calon peserta kompetisi yang sudah membuat akun dan mendaftar pada aplikasi *website* turnamen akan di arahkan pada menu *Payment*.
2. Menu *Payment* ini merupakan menu yang akan mengarahkan calon peserta kompetisi untuk melakukan pembayaran *online* ke rekening panitia turnamen yang telah di tentukan.
3. Calon peserta kompetisi yang sudah mengetahui informasi rekening panitia turnamen wajib melakukan pembayaran maksimal 1x24 jam setelah mengkonfirmasi pembayaran.
4. Bukti pembayaran yakni bukti transfer baik manual ataupun *m-banking* dan mengkonfirmasi pada no *whatsapp* panitia turnamen.
5. Setelah terkonfirmasi maka akun calon peserta kompetisi akan di aktifkan untuk dapat mengikuti turnamen yang akan dilaksanakan.

3.2.4 Analisis Perhitungan Skor

Untuk menyelesaikan permasalahan penilaian skor kompetisi yang ada di *The Pillars E-sport* penulis melakukan survey dengan melakukan wawancara kepada *manager* komunitas. Hasil wawancara dengan *manager* yaitu menyangkut penilaian skor peserta kompetisi yang mempunyai dasar penilaian skor setiap fase. Misalnya untuk kompetisi ini hanya di berlakukan penilaian berdasarkan skor tertinggi.

3.2.5 Analisis Kebutuhan Non Fungsional

Pada analisa *The Pillars E-sport* diantaranya perangkat lunak, serta perangkat keras sebagai bahan analisa kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diterapkan.

3.2.6 Analisis Kebutuhan Perangkat Lunak

Berikut ini adalah perancangan pada Perangkat lunak (*Software*) agar dapat berjalan dengan baik, Adapun perangkat lunak yang diimplementasikan nanti yakni:

1. Sistem Operasi *Windows 10 Ultimate 64 Bit*
2. *Xampp*
3. *Mozzila Firefox/chrome*

3.2.7 Analisis Kebutuhan Perangkat Keras

Perangkat keras (*Hardware*) ini dapat berjalan baik, standar minimal serta spesifikasi (*Hardware*) yang dipergunakan yakni:

1. Processor Intel Core i3
2. Minimal RAM 4 GB
3. Minimal HDD 250 GB
4. Monitor
5. Keyboard
6. Mouse
7. Printer (*Optional*)

Adapun perangkat keras yang dipergunakan dalam membuat sistem pendukung keputusan penentu *peserta* terbaik menggunakan metode *Analytic Hierarki Process* berbasis *web* adalah sebagai berikut:

1. Laptop Lenovo Z3-451 (AMD A10-5757M APU With Radeon™ HD Graphic (4CPUs), ~2,5 GHz)
2. Minimal RAM 4 GB
3. Minimal HDD 250 Gb

3.3 Perancangan Sistem

Perancangan merupakan penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam suatu kesatuan yang utuh dan berfungsi.

3.3.1 Tujuan Perancangan Sistem

Tujuan perancangan sistem ini untuk menghasilkan sistem pendaftaran kompetisi *online* turnamen, sehingga memudahkan pengguna untuk mengolah

data calon peserta kompetisi dan pada akhirnya menghasilkan suatu informasi yang akurat.

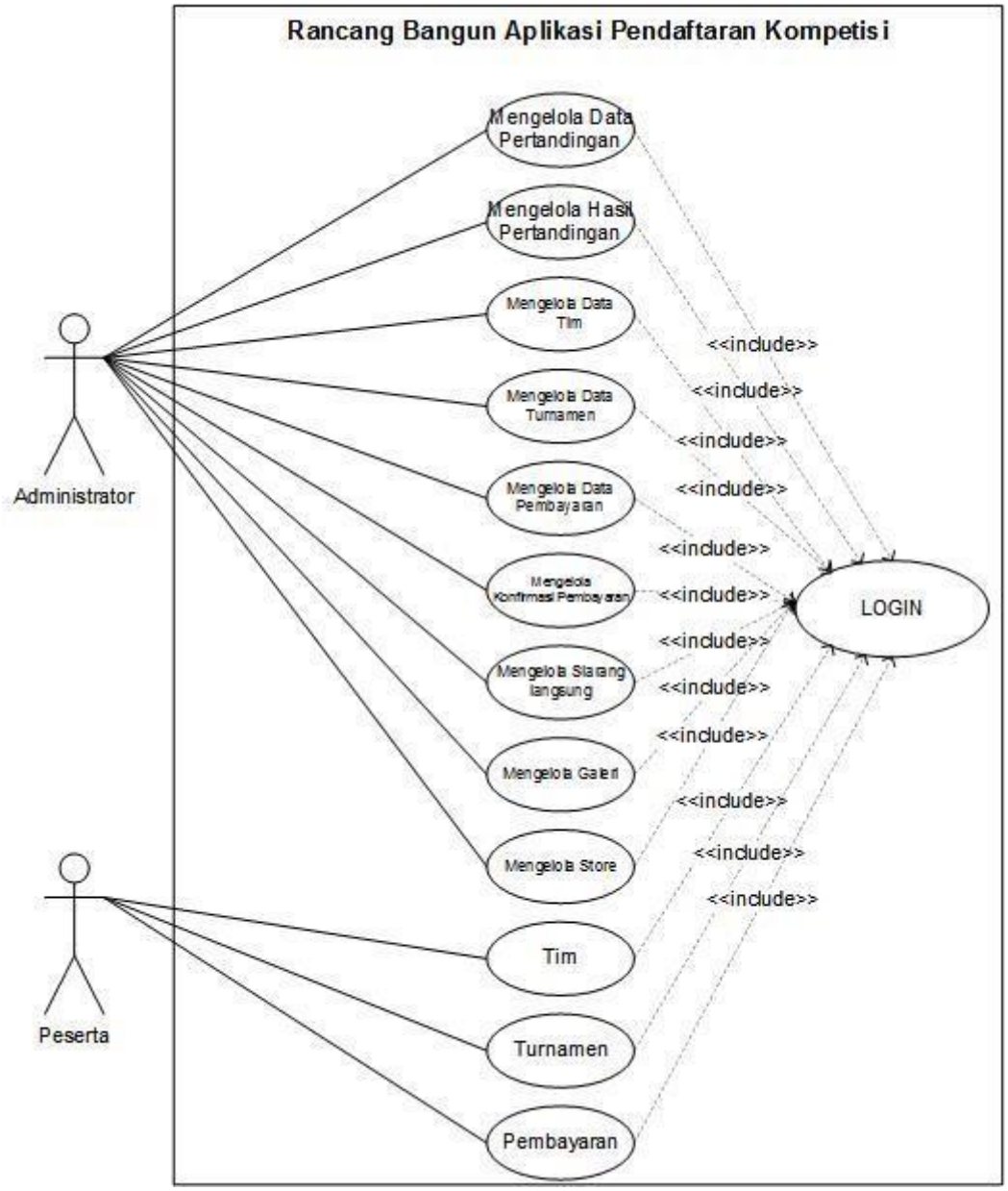
Adapun komponen-komponen sistem pendukung keputusan ini meliputi:

1. Perancangan proses (UML)
2. Perancangan Basis data (Tabel Relasi, Struktur Tabel)
3. Perancangan Program Aplikasi (Perancangan Antarmuka)

3.3.2 UML (*Unified Modeling Language*)

3.3.2.1 *Use case Diagram*

Use case digunakan untuk menjelaskan apa yang terjadi pada sistem. Seperti yang digambarkan pada gambar 3.3, sistem berjalan dengan melibatkan aktor yakni *Administrator* dan Peserta (*Manager*). *Administrator* (Panitia turnamen dan Panitian pendaftaran turnamen) adalah aktor yang bertugas mengisikan informasi turnamen yang ada pada aplikasi pendaftaran kompetisi turnamen *The Pillars E-sport*.



Gambar 3.3 *Use case Diagram* Sistem yang akan dibuat

3.3.2.2 Skenario *Use case Diagram* Sistem

Pada bagian *Use case Diagram* sistem di buat 2 aktor berbeda seperti *Administrator* dan Peserta, hal ini di buat sesuai kebutuhan pada pengaplikasian sistem nantinya dimana *Administrator* bertugas untuk mengelelola keseluruhan

data baik aplikasi maupun data dari Peserta tersebut, kemudian Peserta merupakan *actor* yang akan berperan untuk mempergunakan aplikasi tersebut dengan batasan yang telah di tentukan. Pada *use case* ini juga dapat di lihat perintah pada setiap modul yang diberikan serta respon yang di kirim oleh sistem kepada kedua *actor* tersebut.

Tabel 3.2 Definisi *Administrator*

Aktor	Deskripsi
Administrator	Melakukan seluruh pengolahan data pada sistem aplikasi.

Tabel 3.1 Definisi Peserta

Aktor	Deskripsi
Peserta	Merupakan aktor yang akan mendapatkan laporan alternatif / peserta, laporan hasil perhitungan.

3.3.2.3 Skenario *Use case*

Use case Login merupakan langkah kerja yang di lakukan oleh seluruh *actor* sebelum memulai mangakses aplikasi *web base. Use case Login* ini dibuat untuk menjelaskan apa dan siapa saja yang dapat mengakses apalikasi,

jika *Username* dan *Password* yang diisi benar maka sistem akan menampilkan halaman menu sesuai *role*. Untuk lebih jelas dapat dilihat pada tabel 3.2

1. Nama *Use case* : *Login*
   Aktor : *Administrator*
   Kondisi Awal : Form Halaman Awal terbuka
   Kondisi Akhir : Form Halaman *Administrator* terbuka

Tabel 3.2 Skenario *Use case Login*

Aksi Aktor	Reaksi Sistem
2. Administrator **memasukkan *Username, password*** ***3.*** **Peserta *klik Login***	1. Sistem Menampilkan halaman *Login* 4. Sistem memvalidasi dan menampilkan halaman *Home*

2. Nama *Use case* : Mengelola Pertandingan
   Aktor : *Administrator*
   Kondisi Awal : Halaman *Administrator* terbuka
   Kondisi Akhir : Pertandingan dibuat

Tabel 3.3 Skenario *Use case* Mengelola Pertandingan

Aksi Aktor	Reaksi Sistem
2.*Administrator* memilih menu pertandingan pada *dashboard*	1.Sistem menampilkan halaman *Dashboard Administrator*

4.*Administrator* mengisi form tambah pertandingan	3.sistem menampilkan form pertandingan 5.sistem menyimpan data ke *basis data*

3. Nama *Use case* : Mengelola Tim
   Aktor : *Administrator*
   Kondisi Awal : Form Hasil terbuka
   Kondisi Akhir : *Administrator* masuk form tim

Tabel 3.4 Skenario *Use case* Mengelola Tim

Aksi Aktor	Reaksi Sistem
2. *Administrator* menginputkan data tim	1. Sistem menampilkan halaman tim 3. Sistem menyimpan ke dalam basis data dan menampilkan halaman *dashboard*

4. Nama *Use case* : Mengelola Turnamen
   Aktor : *Administrator*
   Kondisi Awal : Form turnamen terbuka
   Kondisi Akhir : Form hasil turnamen terbuka

Tabel 3.5 Skenario *Use case* Turnamen

Aksi Aktor	Reaksi Sistem
2. *Administrator* memilih menu turnamen pada *dashboard* **3.*Administrator* menginputkan data turnamen**	1. Sistem menampilkan halaman *dashboard*. 4. Sistem menyimpan ke *database* peserta

5. Nama *Use case* : Mengelola pembayaran Aktor : *Administrator*
   Kondisi Awal : Menampilkan halaman pembayaran Kondisi Akhir : Menampilkan hasil pembayaran

Tabel 3.6 Skenario *Use case* Pembayaran

Aksi Aktor	Reaksi Sistem
2. *Administrator* memilih menu pembayaran **3. *Administrator* menginputkan data pembayaran**	1.Sistem menampilkan halaman *dashboard* 4. Sistem menyimpan ke *database*

6. Nama *Use case* : Mengelola Konfirmasi Pembayaran
   Aktor : *Administrator*
   Kondisi Awal : Form konfirmasi pembayaran terbuka
   Kondisi Akhir : Form hasil konfirmasi terbuka

Tabel 3.7 Skenario *Use case* Mengelola Konfirmasi Pembayaran

Aksi Aktor	Reaksi Sistem
2.*Administrator* memilih menu konfirmasi pembayaran **4.*Administrator* mengkonfirmasi pembayaran**	1.Sistem menampilkan halaman *dashboard* 3.Sistem menampilkan pembayaran 5.Sistem menyimpan ke dalam basis data

7. Nama *Use case* : Mengelola *Live Streaming*

Aktor : *Administrator*

Kondisi Awal : Menampilkan Halaman *Streaming*

Kondisi Akhir : Halaman hasil *Streaming* berhasil di buka

Tabel 3.8 Skenario *Use case Live Streaming*

Aksi Aktor	Reaksi Sistem
2.*Administrator* memilih menu setting *Streaming* ***3. Administrator* menginputkan *link Streaming***	1.Sistem menampilkan halaman *dashboard* 3.Sistem menyimpan ke dalam *database* *4.* Sistem menyimpan ke *database*

8. Nama *Use case* : *Logout*

Aktor : *Administrator*

Kondisi Awal : Menampilkan Halaman *Dashboard*

Kondisi Akhir : Menampilkan Halaman Awal

Tabel 3.9 Skenario *Use case Logout*

Aksi Aktor	Reaksi Sistem
2.*Administrator* klik tombol *Logout*	1.Sistem menampilkan Halaman Beranda 3.Sistem menampilkan Halaman Awal

9. Nama *Use case* : Tim

Aktor : Peserta

Kondisi Awal : Halaman *Dashboard* Peserta tampil

Kondisi Akhir : Halaman menu tim tampil

Tabel 3.10 Skenario *Use case* Tim Peserta

Aksi Aktor	Reaksi Sistem
2. Peserta memilih menu Tim **3. Peserta menginput data tim**	1. Sistem menampilkan Halaman *Dashboard* Peserta 4.Sistem menyimpan data ke *database*

10. Nama *Use case* : Turnamen

Aktor : Peserta

Kondisi Awal : Halaman *Dashboard* Peserta tampil

Kondisi Akhir : Halaman menu turnamen tampil

Tabel 3.11 Skenario *Use case* Turnamen Peserta

Aksi Aktor	Reaksi Sistem
2. Peserta memilih menu Tim **3. Peserta menginput data tim**	1.Sistem menampilkan Halaman *Dashboard* Peserta 4.Sistem menyimpan data ke *database*

11. Nama *Use case* : Pembayaran

Aktor : Peserta

Kondisi Awal : Halaman *Dashboard* Peserta tampil

Kondisi Akhir : Halaman menu pembayaran tampil

Tabel 3.12 Skenario *Use case* Pembayaran Peserta

Aksi Aktor	Reaksi Sistem
2. Peserta memilih menu Pembayaran **3. Peserta memilih metode pembayaran**	1.Sistem menampilkan Halaman *Dashboard* Peserta 4.Sistem menyimpan data ke *database*

3.3.2.4 *Activity Diagram*

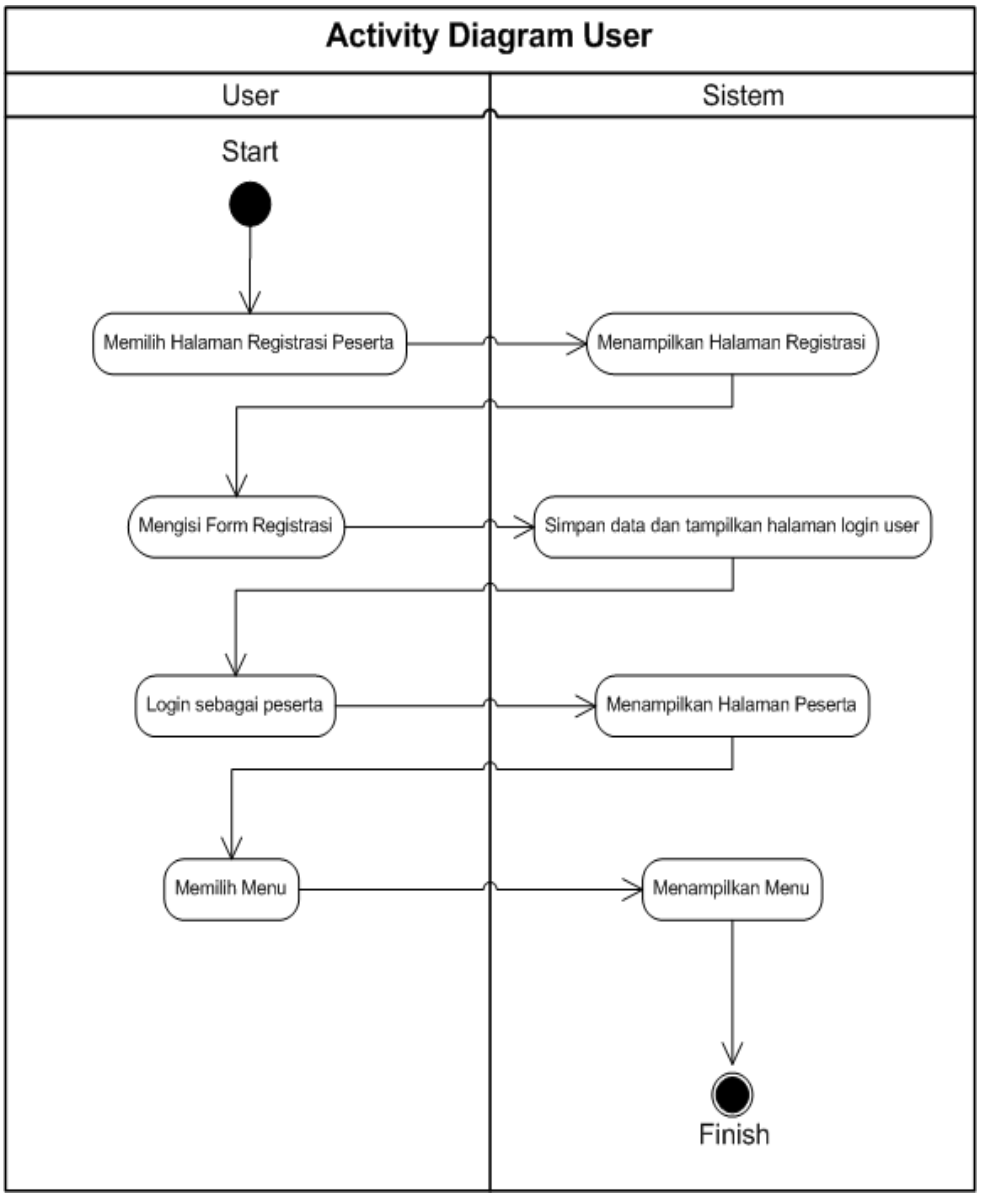
Activity Diagram digunakan untuk menggambarkan aliran dari suatu aktivitas ke aktivitas lainnya dalam sebuah sistem. Sistem menggunakan *Activity*

Diagram untuk menggambarkan aliran aktivitas sebuah interaksi antara *Administrator* dan Peserta terhadap sistem.

Berikut adalah *activity Diagram* dari setiap proses yang ada pada rancang bangun aplikasi pendaftaran kompetisi *online* turnamen *player unknown battle ground* di *The Pillars E-sport* berbasis *web*:

- ***Activity Diagram Peserta***

  Pada *Diagram activity*, Peserta mengakses *website* aplikasi. Setelah itu melakukan *Login* menggunakan *Username* dan *password* yang telah di daftarkan oleh *Administrator*. Kemudian Peserta dapat mengakses halaman utama Peserta dimana terdapat berbagai halaman seperti halaman alternatif, kriteria, perhitungan dan *Logout*. Berikut Gambar 3.4 untuk *Diagram activity* Peserta.

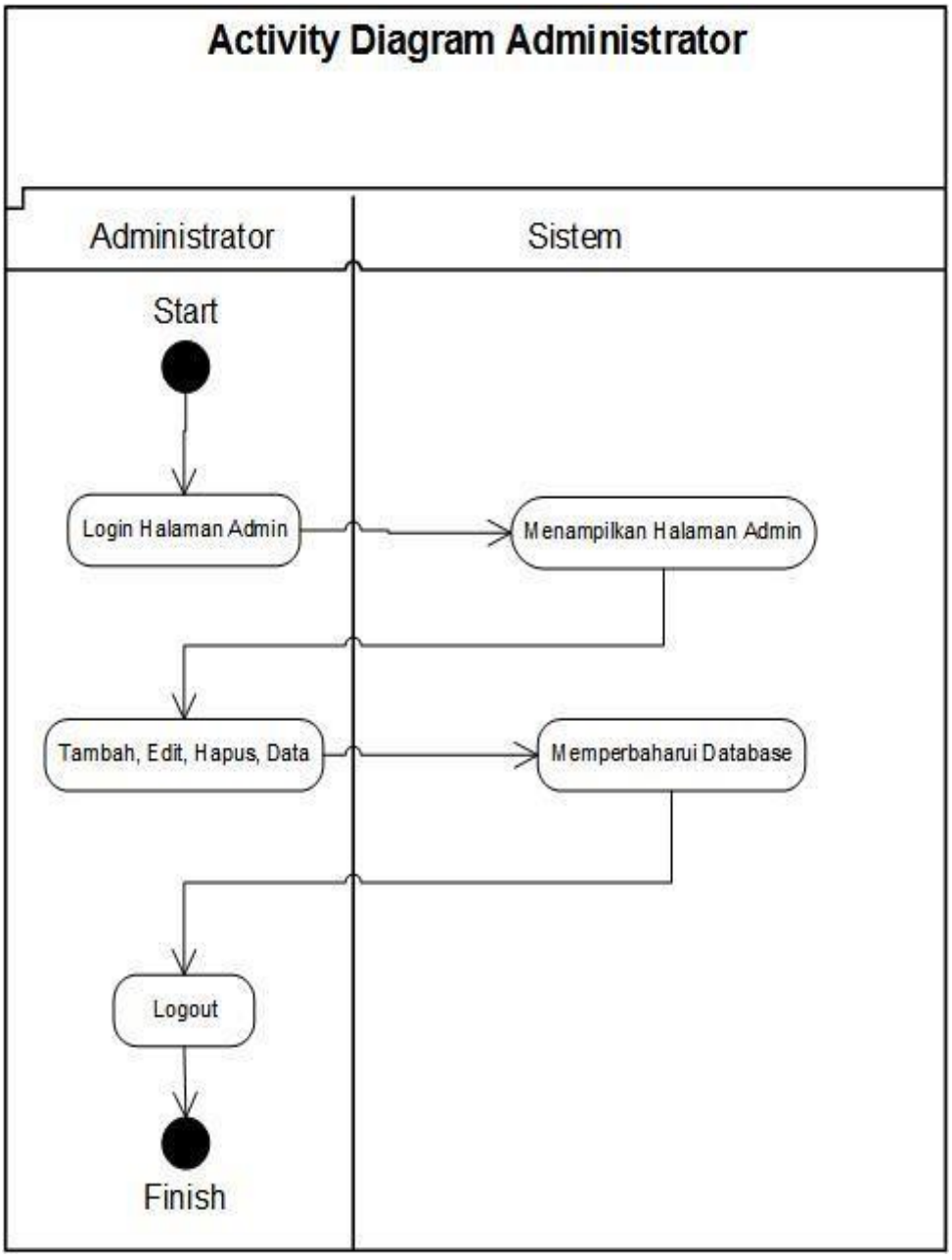


Gambar 3.4 *Activity Diagram* Peserta

❖ ***Activity Diagarm Administrator***

Pada Gambar 3.5 dijelaskan bahwa *Administrator* jika berhasil *Login* akan diarahkan ke halaman utama aplikasi dan jika salah maka akan kembali pada menu *Login*, pada halaman utama ini *Administrator* dapat mengelola aplikasi seperti menambahkan, mengedit bahkan menghapus data, lalu *database* akan

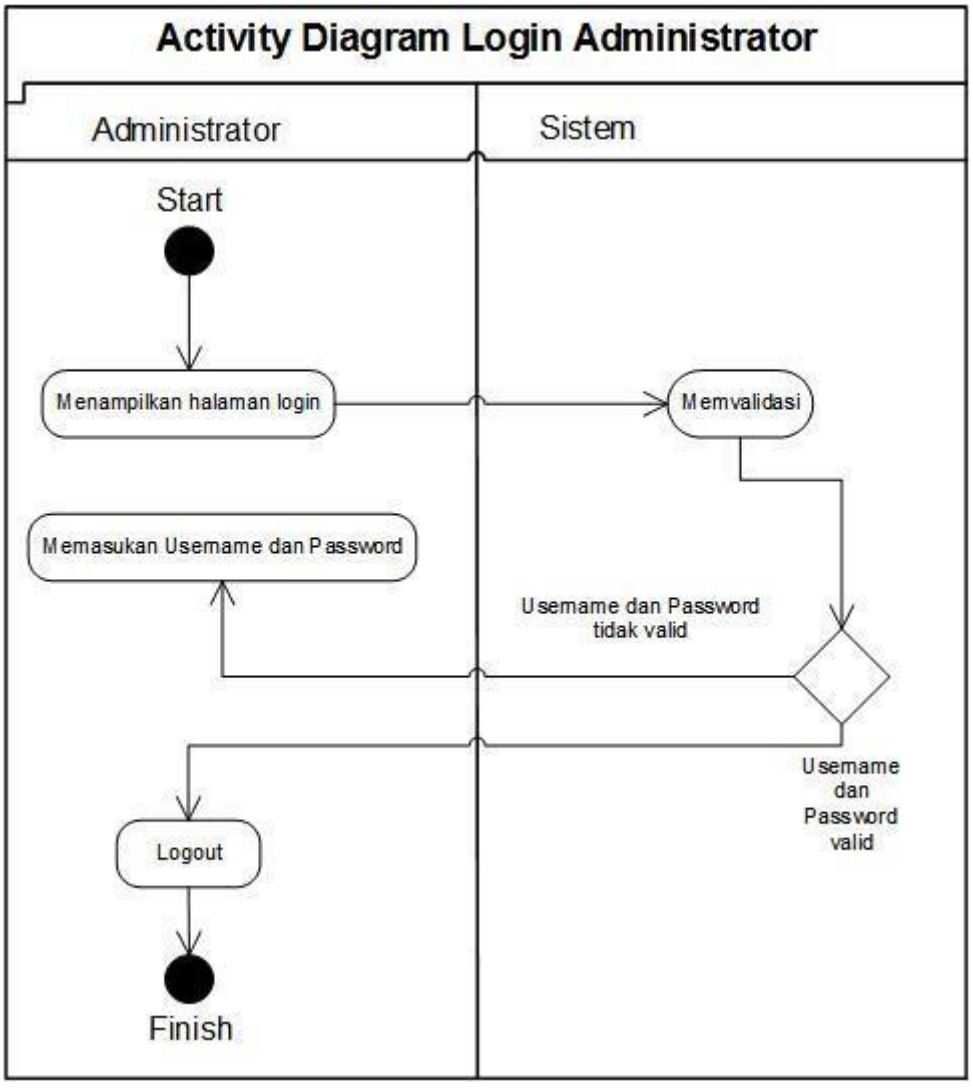
menyimpan dan memperbarui data. setelah itu *Administrator* dapat melakukan *Logout*.



Gambar 3.5 *Activity Diagram Administrator*

❖ ***Activity Diagram Login Administrator***

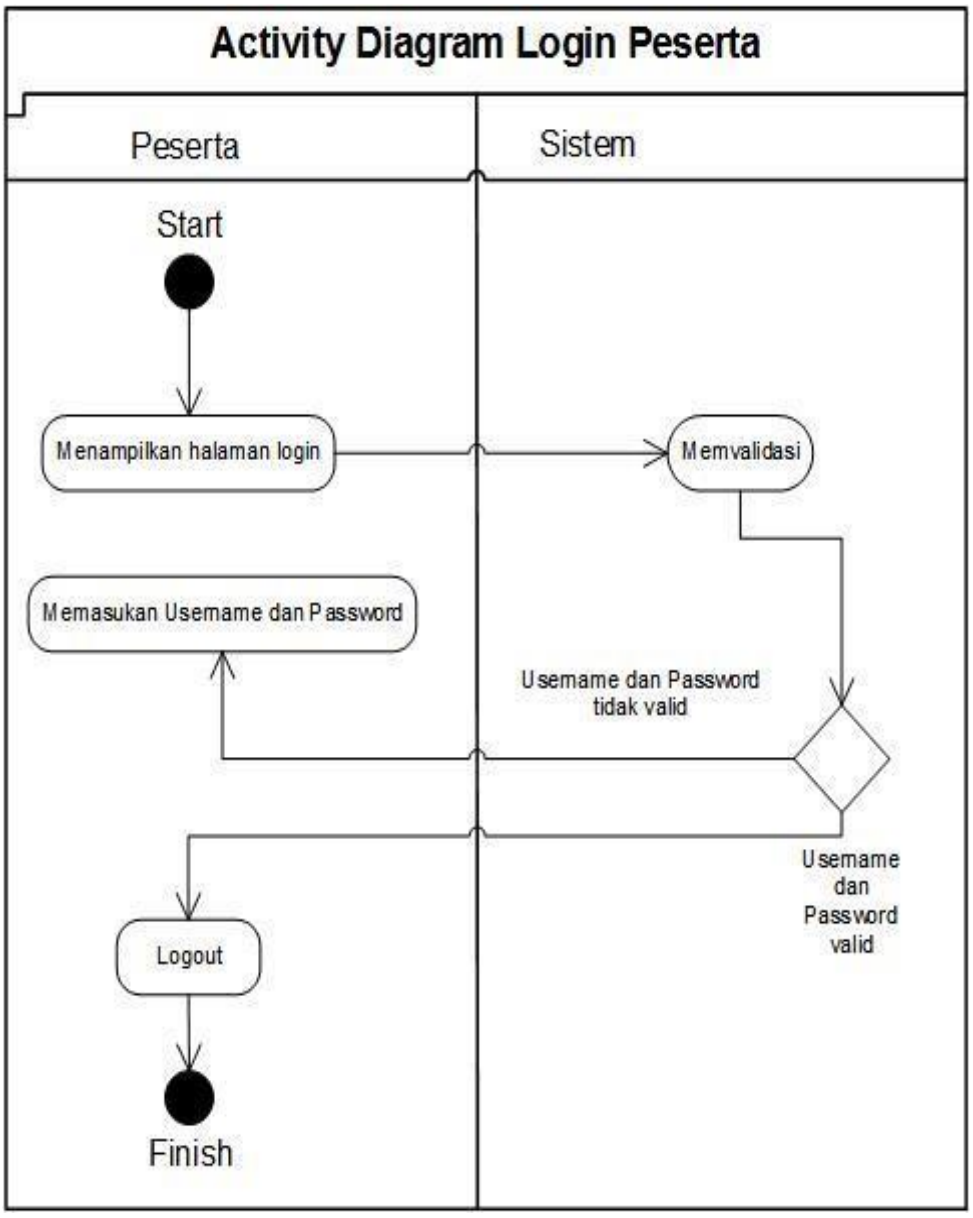
Pada *Diagram activity Login, Administrator* menginput *Username* dan *password.* Setelah itu sistem akan melakukan validasi apakah *Username* dan *password* telah diisi dengan benar dan datanya di cocokan dengan *database* yang ada. Jika sesuai maka akan diarahkan pada halaman utama dan jika tidak sesuai maka muncul pesan kesalahan dan diarahkan untuk *Login* kembali. Berikut gambar 3.6 untuk *Diagram activity Login Administrator.*



Gambar 3.6 *Activity Diagram Login Administrator*

❖ ***Activity Diagram Login Peserta***

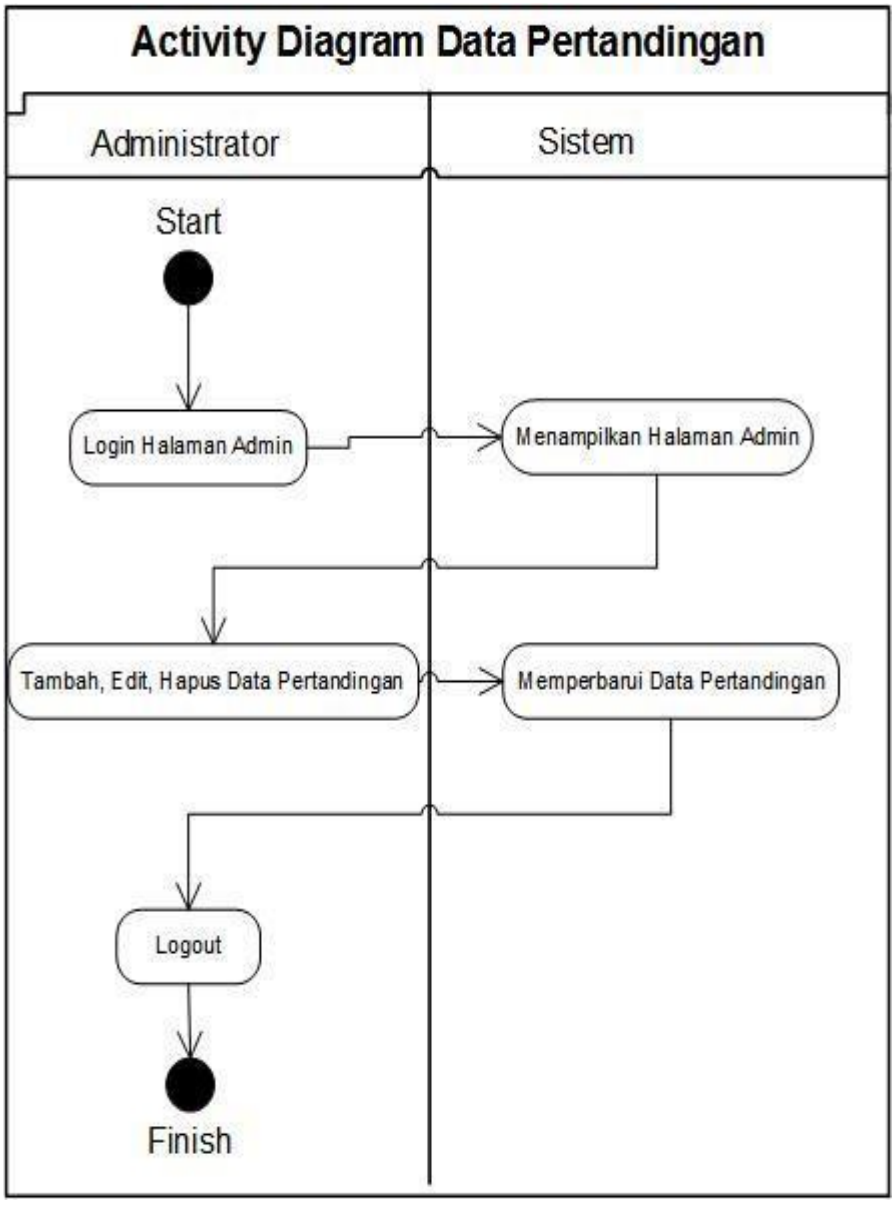
Pada *Diagram activity Login*, Peserta mengakses aplikasi *web* dan melakukan *Login* dengan *Username* dan *password* yang telah di daftarkan oleh *Administrator*, jika benar maka akan masuk ke halaman utama Peserta, jika salah maka akan kembali ke halaman *Login* awal.



Gambar 3.7 *Activity Diagram Login* Peserta

❖ ***Activity Diagram*** **Pertandingan**

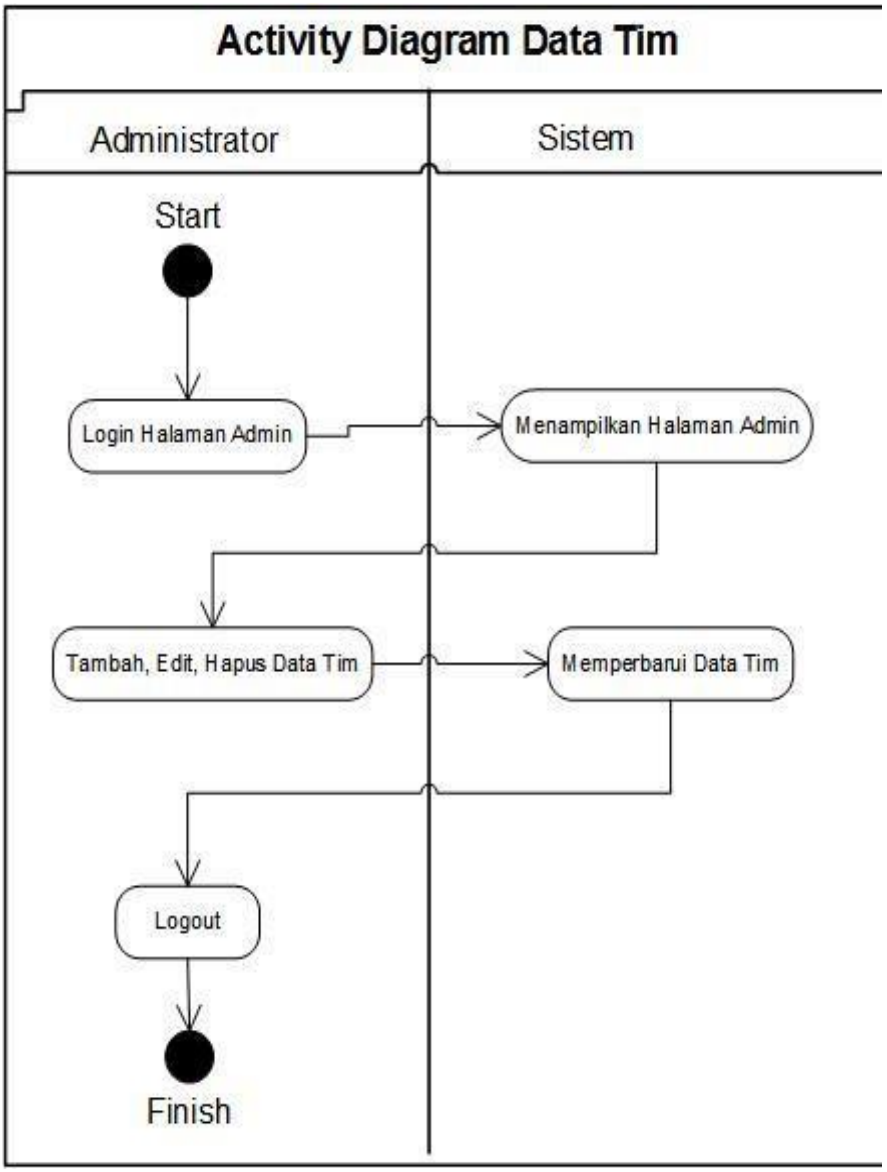
Pada *Diagram activity* Pertandingan, *Administrator* dapat melakukan input data pertandingan, kemudian sistem akan menyimpan data kedalam *database*, *Administrator* juga dapat melakukan edit dan hapus data sesuai kebutuhan.



Gambar 3.8 *Activity Diagram Administrator* Mengelola Pertandingan

❖ ***Activity Diagram* Tim**

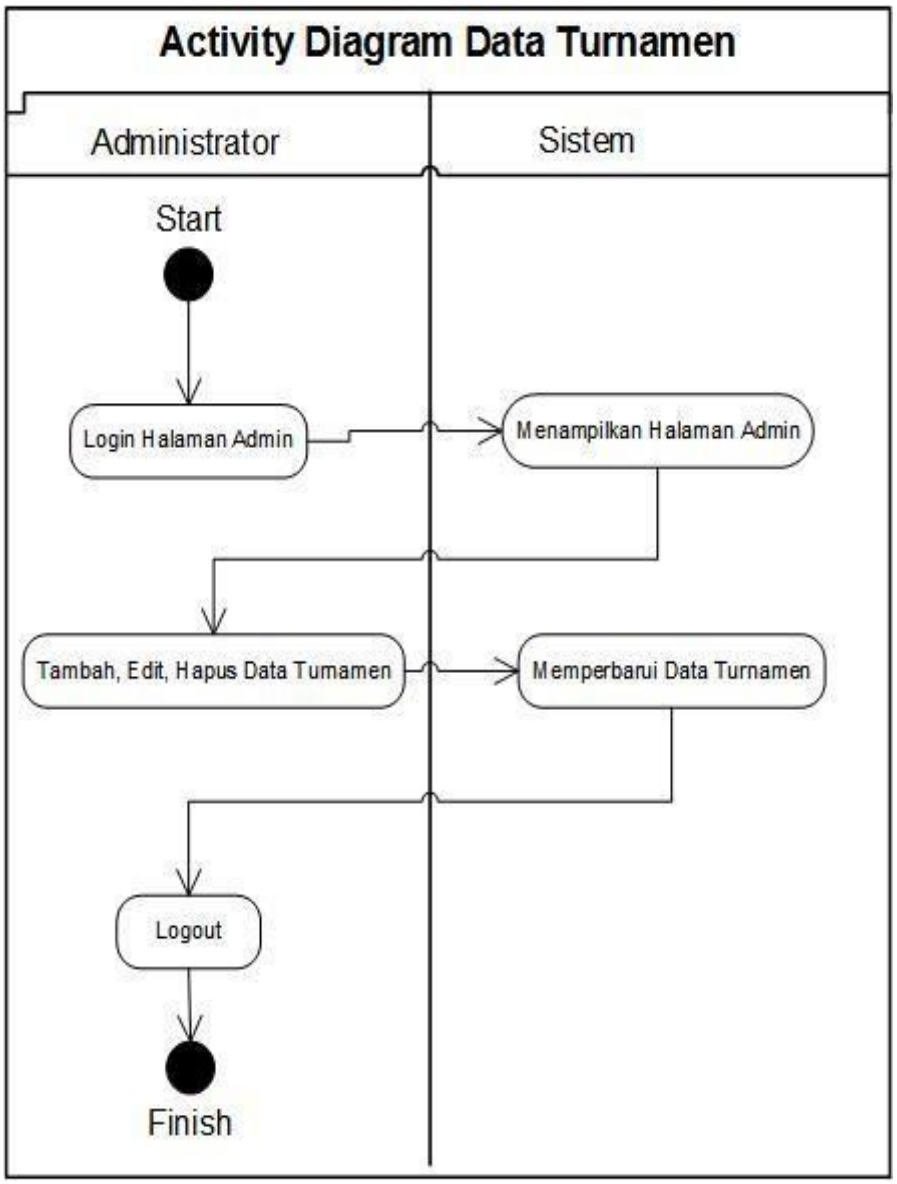
Pada *Diagram activity* tim, *Administrator* dapat melakukan tambah tim sesuai kebutuhan, dan juga dapat mengedit data yang sudah ada serta menghapus data tim bila diperlukan. Kemudian data akan di proses oleh sistem dan disimpan kedalam *database*.



Gambar 3.9 *Activity Diagram Administrator* Mengelola Data Tim

❖ ***Activity Diagram*** **Turnamen**

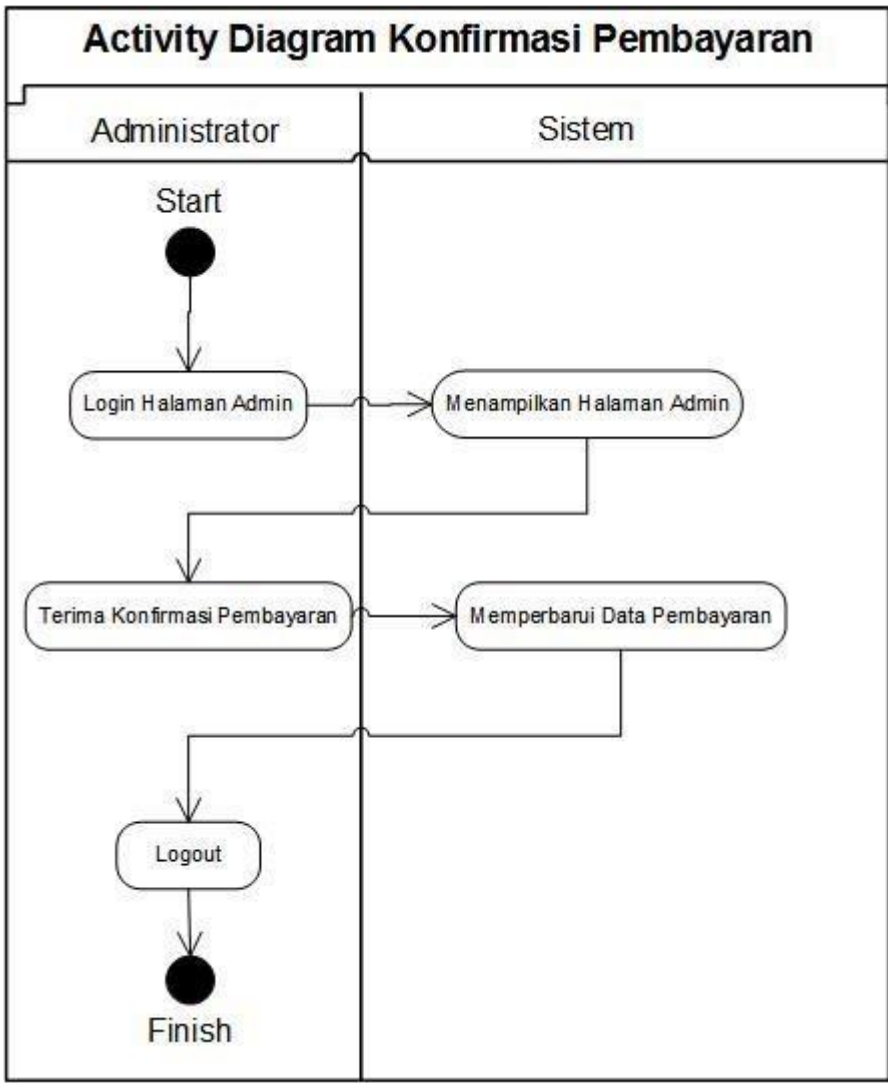
Pada Gambar 3.10 dijelaskan *Diagram activity* data Turnamen. *Administrator* dapat mengisi data turnamen sesuai kebutuhan dan juga melakukan edit dan hapus data turnamen jika di perlukan.



Gambar 3.10 *Activity Diagram Administrator* Mengelola Data Turnamen

❖ ***Activity Diagram* Data dan Konfirmasi Pembayaran**

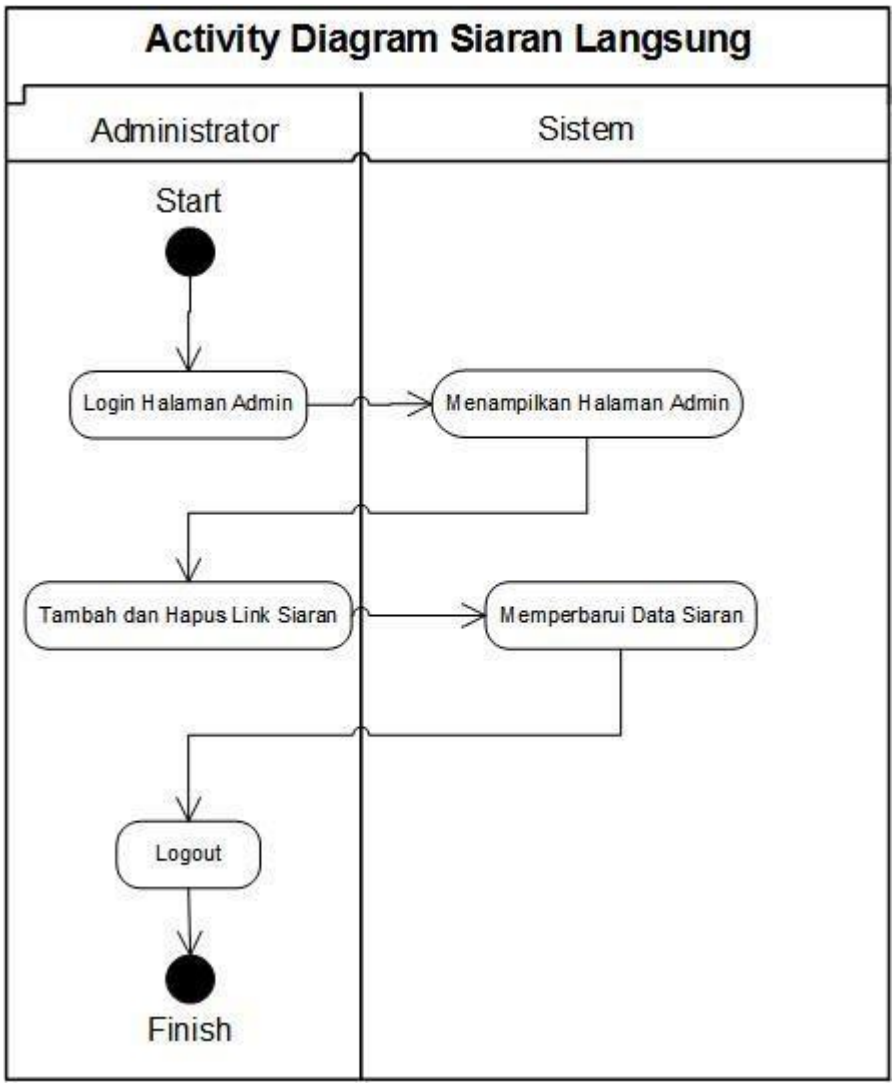
Pada Gambar 3.10 dijelaskan *Diagram activity* data konfirmasi pembayaran. *Administrator* dapat melakukan input data pembayaran dan melakukan konfirmasi pembayaran kemudian sistem akan menyimpan dalam *database* turnamen. Data yang sudah disimpan dapat di di edit atau di hapus jika di perlukan.



Gambar 3.11 *Activity Diagram Administrator* Mengelola Data Konfirmasi Pembayaran

❖ ***Activity Diagram* Data Siaran Langsung**

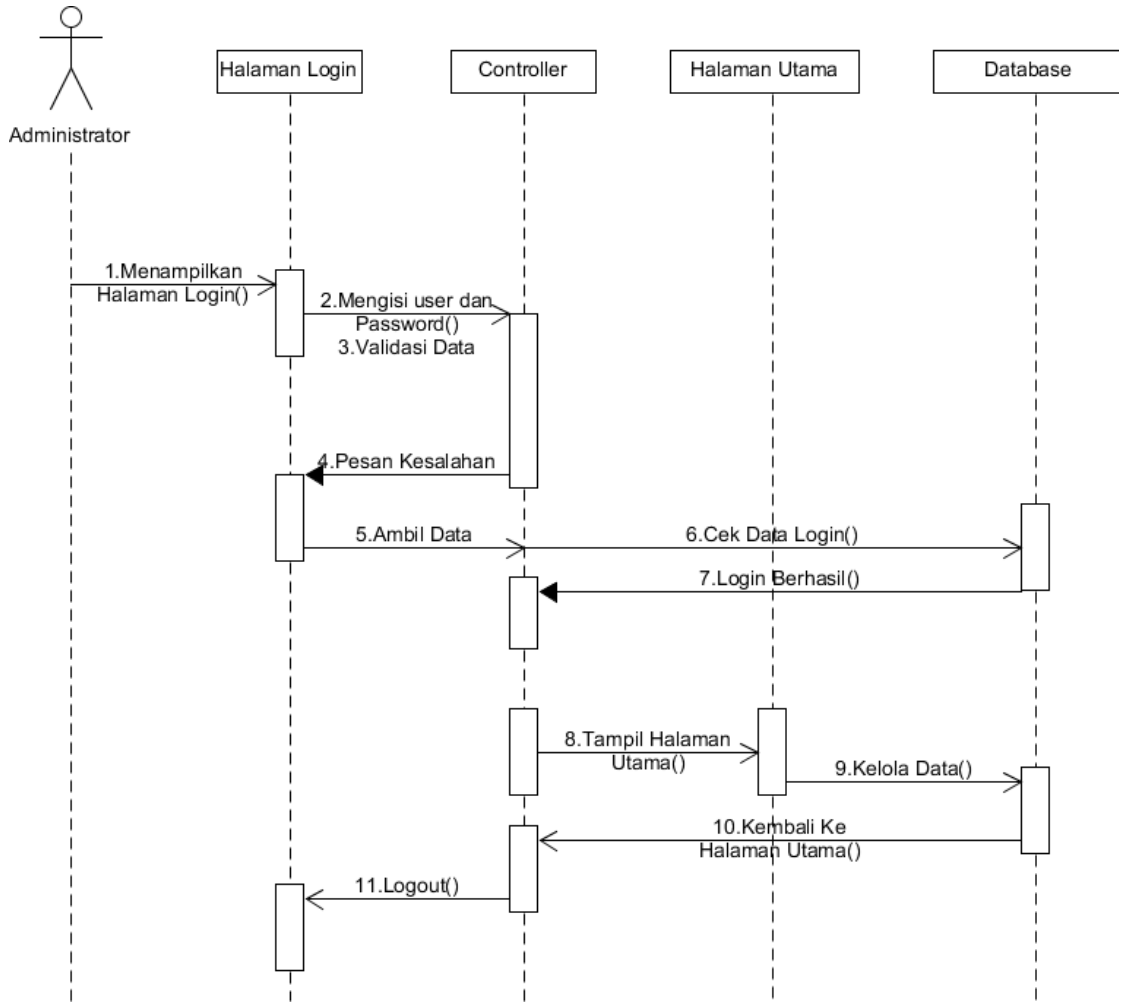
Pada Gambar 3.12 dijelaskan *Diagram activity* data siaran langsung. *Administrator* dapat melakukan input *link* dari saluran siaran langsung kemudian sistem akan menyimpan dalam *database* turnamen. Data yang sudah disimpan dapat di di edit atau di hapus jika di perlukan.



Gambar 3.12 *Activity Diagram Administrator* Mengelola Siaran Langsung

3.3.2.5 Sequence Diagram

❖ ***Diagram Sequence Administrator* Keseluruhan**

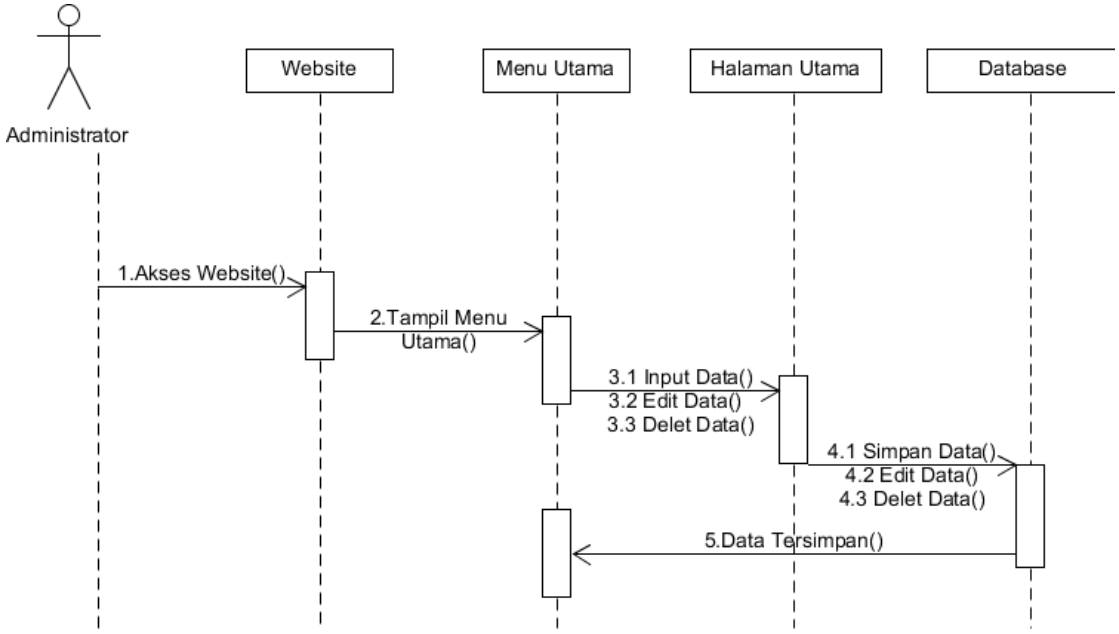


Gambar 3.13 *Diagram Sequence Administrator* Keseluruhan

Pada *Diagram sequence Administrator* diatas, pertama *Administrator* mengisi *Username* dan *password*. Setelah itu sistem akan mengecek validasi kedalam *database* apakah *Username* dan *password* yang terisi telah benar, jika benar maka sistem akan mengarahkan ke halaman utama dalam sistem, jika salah maka akan muncul pesan kesalahan dan diarahkan kehalaman *Login* kembali,

pada halaman utama terdapat menu yang dapat *Administrator* lakukan yakni input, edit, delet data sesuai kebutuhan dan *Logout* apabila telah selesai.

❖ ***Diagram Sequence Administrator* Input, Edit dan Delet Data**



Gambar 3.14 *Diagram Sequence Administrator* Input, Edit dan Delet Data

Pada *sequence Diagram* diatas *Administrator* mengakses *website,* kemudian masuk kedalam menu utama. Pada halaman menu utama terdapat aksi untuk input, edit dan delet data jika di perlukan dan *database* akan melakukan *update* data dan menyimpannya.

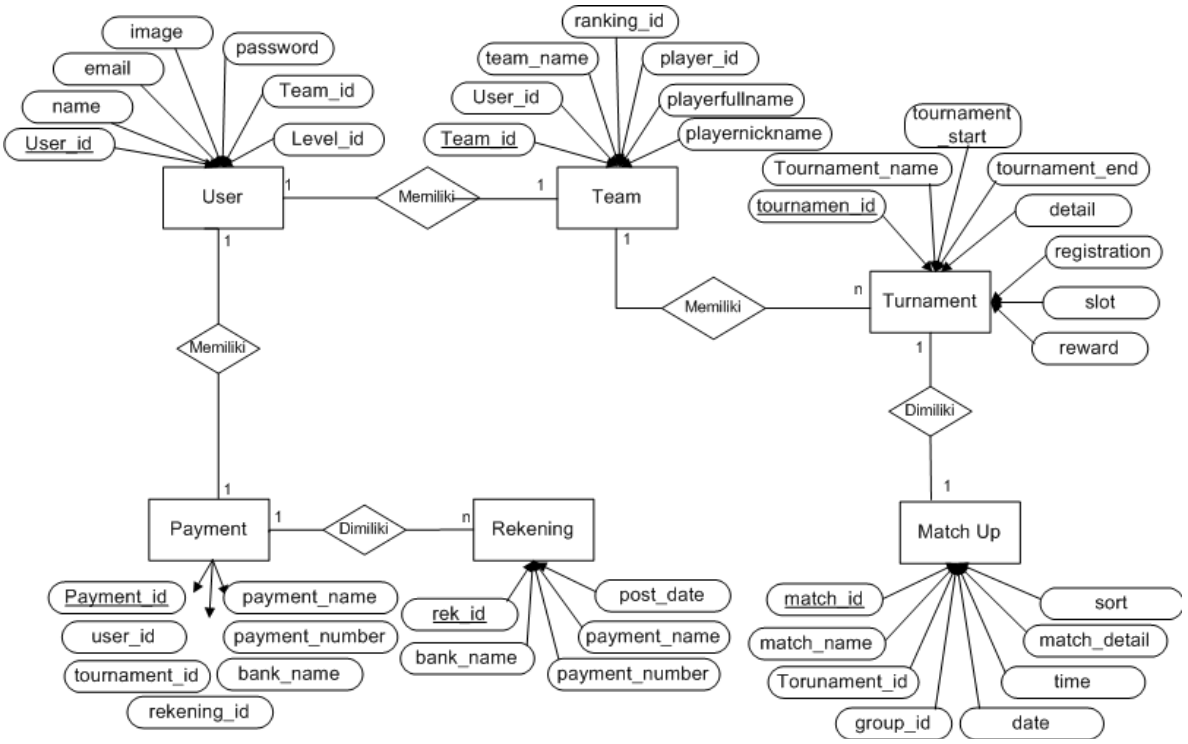
3.4 *Class Diagram*

Class Diagram menggambarkan struktur dan deskripsi *class* serta hubungannya antar *class. Class Diagram* mirip *Entitiry Diagram (ERD)* pada

perancangan *database* bedanya pada *Entity Diargram (ERD)* tidak terdapat operasi/metode tetapi hanya atribut. *Class* terdiri dari nama kelas, atribut dan operasi/mode.

3.4.1 *Class Diagram* Rancang Bangun Aplikasi Turnamen berbasis *web*

Berikut ini adalah *class Diagram* rancang bangun aplikasi turnamen *The Pillars E-sport* berbasis *web*:



Gambar 3.15 *Class Diagram*

3.4.2 Perancangan *Database*

Perancangan *Database* ini termasuk bagian yang sangat penting dalam pembuatan aplikasi rancang bangun pendaftaran kompetisi turnamen *player unknown battle ground The Pillars E-sport* berbasis *web* ini. *Database* ini dirancang sebagai tempat penyimpanan data-data yang terhubung ke dalam

aplikasi. *Database* yang akan digunakan dalam aplikasi ini adalah *database MYSQL Server*.

3.4.3 Struktur Tabel

Untuk mendeskripsikan kebutan data dalam perancangan basis data, maka di buatlah struktur data yang di butuhkan melalui table data berikut berdasarkan *class Diagram* yang telah di rancang sebelumnya.

Dari *class Diagram* yang telah di rancang, maka dapat terbentuk sebuah *database* di dalam *database* tersebut terdapat beberapa tabel yakni:

Tabel 3.13 Peserta

Field	Tipe Data	Size	Keterangan
Peserta_id	Int	11	Primary_key
Name	Varchar	25	
Email	Varchar	25	
Image	Varchar	30	
Password	Varchar	25	
Team_id	Int	11	
Level	Varchar	25	
Is_active	Int	1	
Date_created	Int	1	

Tabel 3.14 Tabel *Team*

Field	Tipe Data	Size	Keterangan

Team_id	Int	11	Primary_key
Peserta_id	Varchar	25	
Team_name	Int	11	
Ranking_id	Int	11	
Playerfullname	Varchar	25	
Playernickname	Varchar	25	
Contact	Int	25	
foto	Varchar	25	

Tabel 3.15 *Player*

Field		**Tipe Data**	**Size**	**Keterangan**
Player_id		Int	11	Primary_key
Player_name		Varchar	25	
Player_nickname		Varchar	25	
Peserta_id		Int	11	
Team_id		Int	11	
Player_detail		Int	11	
image		Varchar	30	

Tabel 3.16 *Match up*

Field	**Tipe Data**	**Size**	**Keterangan**
Match _id	Int	11	Primary_key
Match _name	Varchar	25	
*Tournament*_id	Int	11	
Grour_id	Int	11	
Date	Int	11	
Time	Int	11	

Match _detail	Varchar	25	
Sort	Int	2	
Timemodified	Timestamp		
Team_id	Varchar	25	
poin	Varchar	25	

Tabel 3.17 *Ranking*

Field	Tipe Data	Size	Keterangan
Ranking_id	Int	11	Primary_key
Ranking_*Result*	Varchar	25	

Tabel 3.18 *Tournament*

Field	Tipe Data	Size	Keterangan
*Tournament*_id	Int	11	Primary_key
*Tournament*_name	Int	11	
*Tournament*_start	Date	Now	
Time_start	Date	Now	
Time_end	Varchar	25	
*Tournament*_detail	Varchar	25	
Registration_fee	Int	11	
Slot	Int	11	
reward	Int	25	

Tabel 3.19 *Result*

Field	Tipe Data	Size	Keterangan
Poin_id	Int	11	Primary_key
Poin1	Int	11	

Poin2	Int	11	
Total	Int	11	
Team_id	Int	11	
Match _id	Int	11	

Tabel 3.17 *Payment*

Field	Tipe Data	Size	Keterangan
*Payment*_id	Int	11	Primary_key
Peserta_id	Int	11	
*Tournament*_id	Int	11	
Fee	Int	11	
Rekening_id	Int	11	
Bank_name	Varchar	25	
*Payment*_number	Int	20	
*Payment*_name	Varchar	25	
*Payment*_date	Date		
Proof	Varchar	25	
Status	Varchar	25	
date*update*	Timestamp		

3.5 Perancangan Antar Muka Pengguna (Peserta *Interface*)

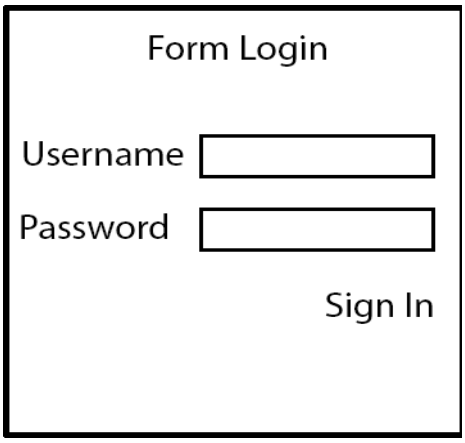
Perancangan antar muka dari rancang bangun aplikasi pendaftaran kompetisi *online* turnamen *player unknown battle ground The Pillars E-sport* berbasis *web*.

3.5.1 Perancangan Tampilan Aplikasi

Perancangan tampilan pada aplikasi terdiri dari bebearapa halaman menu, yaitu halaman *Login, Dashboard* Peserta dan *Administrator*, turnamen, tim, hasil, *merchandise*, pembayaran, skor.

3.5.1.1 Halaman *Login*

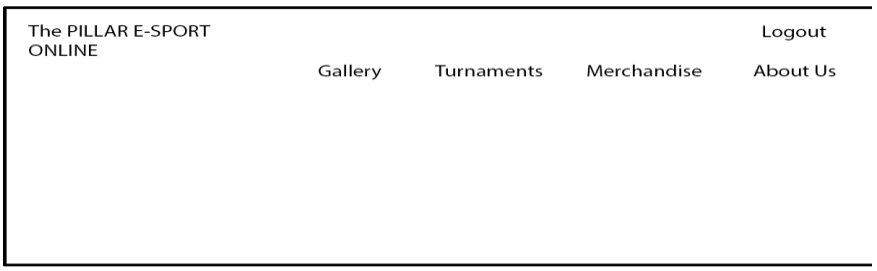
Rancangan halaman *Login* pada tampilan aplikasi dapat di lihat pada gambar berikut ini:



Gambar 3.17 Tampilan *Login*

3.5.1.2 Halaman Menu Utama

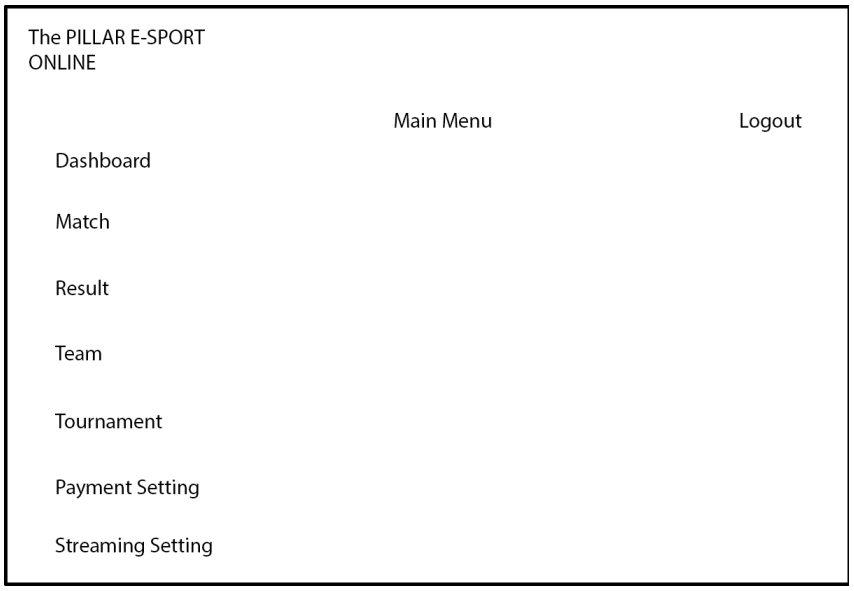
Halaman Menu Utama ketika *Login* berhasil dilakukan.



Gambar 3.18 Tampilan Menu Utama

3.5.1.3 Halaman Menu *Dashboard*

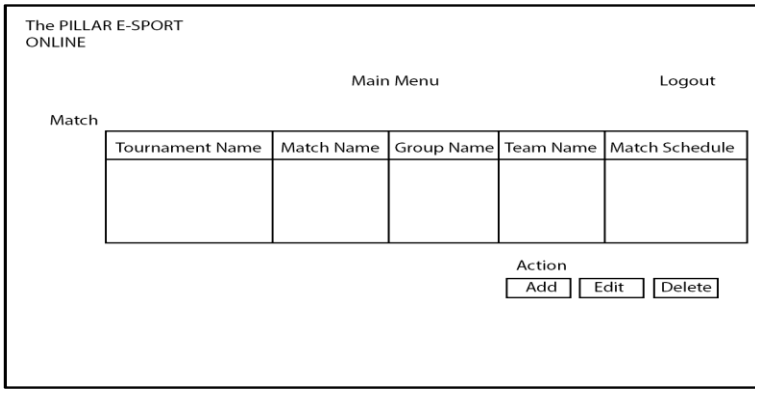
Halaman Menu *Dashboard* pada aplikasi.



Gambar 3.19 Tampilan Halaman *Dashboard*

3.5.1.4 Halaman Menu *Match*

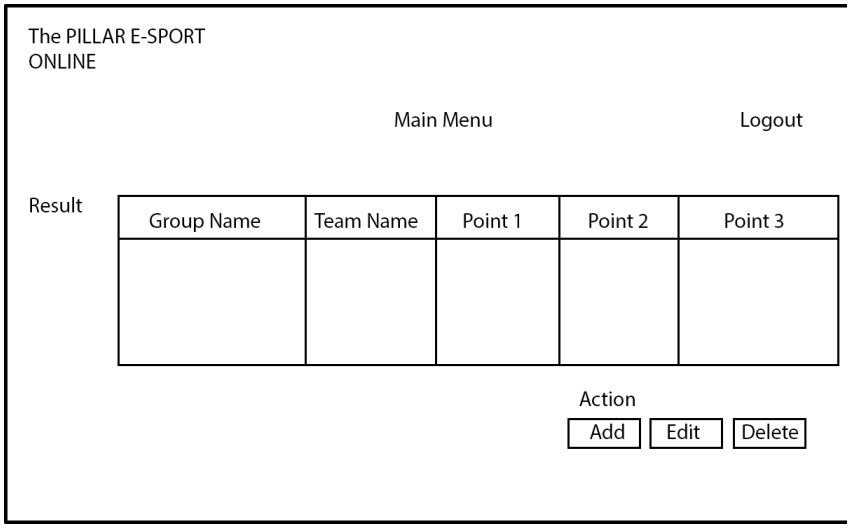
Halaman Menu *Match* pada aplikasi.



Gambar 3.20 Tampilan Halaman *Match*

3.5.1.5 Halaman Menu *Result*

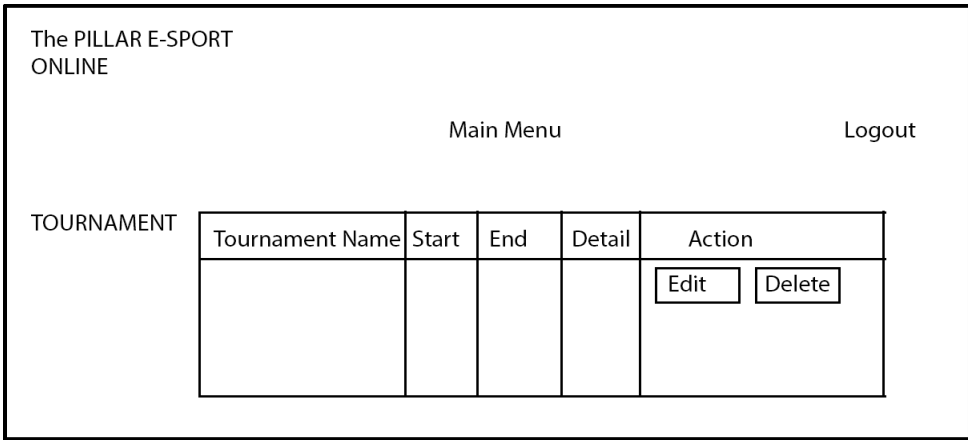
Halaman Menu *Result* pada aplikasi.



Gambar 3.21 Tampilan Halaman *Result*

3.5.1.6 Halaman Menu *Tournament*

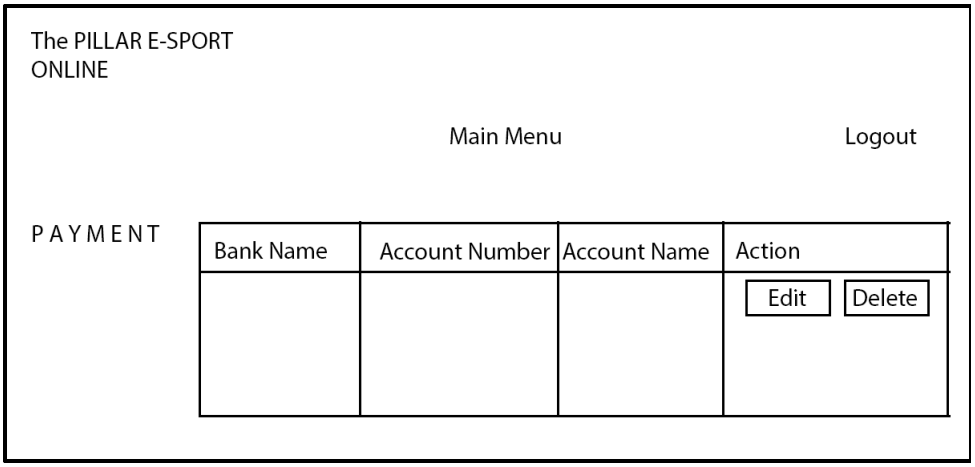
Halaman Menu *Tournament* pada aplikasi.



Gambar 3.22 Tampilan Halaman *Tournament*

3.5.1.6 Halaman Menu *Payment*

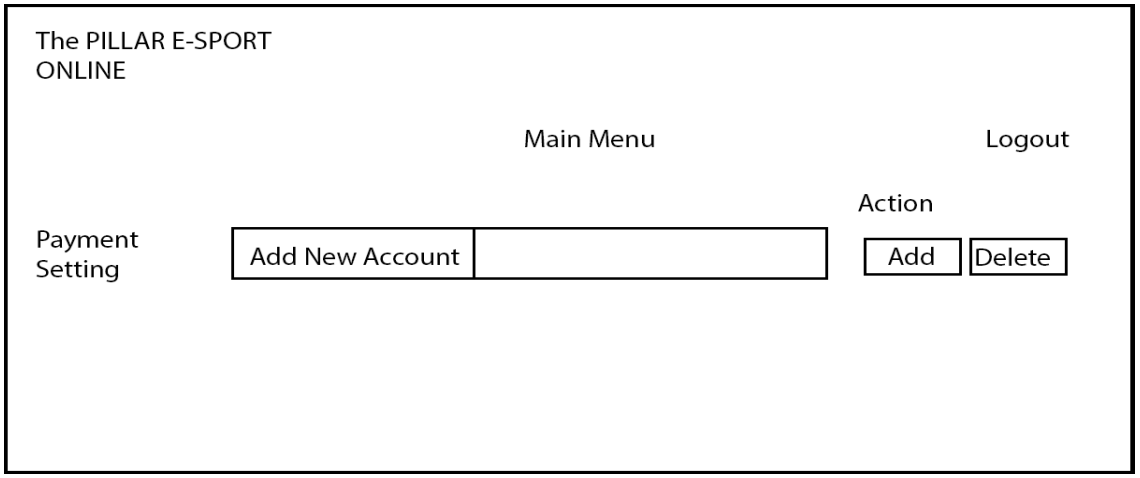
Halaman Menu *Payment* pada aplikasi.



Gambar 3.23 Tampilan Halaman *Payment*

3.5.1.7 Halaman Menu *Payment setting*

Halaman *Payment setting* pada aplikasi.



Gambar 3.24 Tampilan Halaman *Payment setting*

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Kebutuhan Implementasi Perangkat

Pada tahap ini adalah tahap perancangan penulis akan menampilkan hasil dari perancangan aplikasi berbasis *website*. Pada proses ini dilakukan dalam beberapa tahap diantaranya berupa persiapan, perangkat, pendukung, dan implemantasi sistem sebagai hasil dari implemantasi yang telah dibuat dan sebagai tahap akhir adalah pengujian sistem.

Langkah awal dari implementasi adalah mempersiapkan perangkat pendukung yang diperlukan diantaranya adalah perangkat keras yaitu komputer dan perangkat lunak yaitu program aplikasi yang digunakan dalam membuat *website*. Perangkat lunak yang digunakan dalam membangun aplikasi ini seperti di jelaskan pada sub bab di bawah ini.

4.1.1 Kebutuhan Implementasi Hardware

Implementasi aplikasi ini dapat dijalankan dengan baik maka diperlukan spesifikasi standar minimum dari suatu perangkat keras atau *hardware* agar dapat menjalankan aplikasi.

Spesifikasi perangkat yang diperlukan untuk membangun sist

em adalah sebagai berikut:

1. *PC* dengan *Processor Core Xeon(R) CPU E5-2680,*
2. *Random Access Memory* (RAM) 6 Gb,
3. *Harddisk* 250 Gb,
4. *LAN Card 10/100Mbps,*

4.1.2 Kebutuhan Implementasi *Software*

Dalam pembutan aplikasi, membutuhkan beberapa perangkat lunak (*software*) perangkat lunak komputer yang digunakan yaitu:

1. Sistem Operasi pada PC/Laptop

   Pada tahap implementasi dan pengujian ini menggunakan *Windows 10 Profesional,*

2. Mozzila Firefox

   IIS digunakan sebagai *web server,*

3. Microsoft SQL *Server* 2014

   Microsoft SQL *Server* 2014 digunakan sebagai *database server,*

4. XAMPP

   Xampp sebagai *server local* pada *computer*.

4.2 Implementasi Sistem

Berikut ini tampilan yang sudah jadi merupakan hasil dari perancangan yang sebelumnya sudah dilakukan pada tahap perancangan.

4.2.1 Implementasi Database

Pada tahap ini dilakukan implementasi dibuat perancangan *database* sesuai dengan tahap sebelumnya. Berikut tampilan implementasi *database* rancang bangun aplikasi pendaftaran *online* turnamen *PLAYER UNKOWN BATTLE GROUND* di *The Pillars E-sport* berbasis *web*.

Dibawa ini merupakan tampilan implementasi tabel *user* berisi kolom yang nantinya berisikan data yang inputkan.

	#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
☐	1	user_id	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
☐	2	name	varchar(255)	latin1_swedish_ci		Tidak	*Tidak ada*		
☐	3	email	varchar(255)	latin1_swedish_ci		Tidak	*Tidak ada*		
☐	4	contact	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
☐	5	alamat	text	latin1_swedish_ci		Ya	*NULL*		
☐	6	image	varchar(255)	latin1_swedish_ci		Tidak	*Tidak ada*		
☐	7	password	varchar(255)	latin1_swedish_ci		Tidak	*Tidak ada*		
☐	8	level_id	int(11)			Tidak	*Tidak ada*		
☐	9	is_active	int(1)			Tidak	*Tidak ada*		
☐	10	date_created	int(11)			Tidak	*Tidak ada*		

Gambar 4.1 Tabel *User*

Dibawa ini merupakan tampilan implementasi tabel *Team* berisi kolom yang nantinya berisikan data yang inputkan.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
1	team_id	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
2	user_id	int(11)			Tidak	*Tidak ada*		
3	team_name	varchar(32)	latin1_swedish_ci		Tidak	*Tidak ada*		
4	ranking_id	varchar(32)	latin1_swedish_ci		Tidak	*Tidak ada*		
5	player_id	int(11)			Ya	*NULL*		
6	playerfullname	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
7	playernickname	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
8	contact	int(255)			Ya	*NULL*		
9	foto	varchar(255)	latin1_swedish_ci		Ya	*NULL*		

Gambar 4.2 Tabel *Team*

Dibawa ini merupakan tampilan implementasi tabel *Players* berisi kolom yang nantinya berisikan data yang inputkan.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
1	player_id	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
2	player_name	varchar(32)	latin1_swedish_ci		Tidak	*Tidak ada*		
3	player_nickname	varchar(32)	latin1_swedish_ci		Tidak	*Tidak ada*		
4	team_id	int(11)			Tidak	*Tidak ada*		
5	player_detail	text	latin1_swedish_ci		Ya	*NULL*		
6	image	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
7	id_game	varchar(255)	latin1_swedish_ci		Ya	*NULL*		

Gambar 4.3 Tabel *Players*

Dibawa ini merupakan tampilan implementasi tabel *Match up* berisi kolom yang nantinya berisikan data yang inputkan.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
1	match_id	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
2	match_name	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
3	tournament_id	int(11)			Tidak	*Tidak ada*		
4	group_id	int(11)			Ya	*NULL*		
5	date	date			Ya	*NULL*		
6	time	time			Ya	*NULL*		
7	match_detail	varchar(32)	latin1_swedish_ci		Tidak	*Tidak ada*		
8	sort	int(2)			Ya	0		
9	timemodified	timestamp			Ya	current_timestamp()		
10	team_id	int(11)			Ya	*NULL*		

Gambar 4.4 Tabel *Match up*

Dibawa ini merupakan tampilan implementasi tabel *Ranking* berisi kolom yang nantinya berisikan data yang inputkan.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
1	tournament_name	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
2	match_name	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
3	group_name	int(11)			Ya	*NULL*		
4	date	date			Ya	*NULL*		
5	time	time			Ya	*NULL*		
6	team_name	varchar(32)	latin1_swedish_ci		Tidak	*Tidak ada*		
7	tournament_id	int(11)			Tidak	0		
8	match_id	int(11)			Tidak	0		
9	team_id	int(11)			Tidak	0		

Gambar 4.5 Tabel *Ranking*

Dibawa ini merupakan tampilan implementasi tabel *Tournament* berisi kolom yang nantinya berisikan data yang inputkan.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
1	tournament_id	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
2	tournament_name	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
3	tournament_start	date			Ya	*NULL*		
4	time_start	time			Ya	*NULL*		
5	tournament_end	date			Ya	*NULL*		
6	time_end	time			Ya	*NULL*		
7	tournament_detail	text	latin1_swedish_ci		Ya	*NULL*		
8	registration_fee	int(11)			Ya	*NULL*		
9	slot	int(11)			Ya	*NULL*		
10	reward	int(255)			Ya	*NULL*		

Gambar 4.6 Tabel *Tournament*

Dibawa ini merupakan tampilan implementasi tabel *Result* berisi kolom yang nantinya berisikan data yang inputkan.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
1	point_id	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
2	point1	int(11)			Tidak	*Tidak ada*		
3	point2	int(11)			Tidak	*Tidak ada*		
4	total	int(11)			Tidak	*Tidak ada*		
5	team_id	int(11)			Tidak	*Tidak ada*		
6	match_id	int(11)			Tidak	*Tidak ada*		

Gambar 4.7 Tabel *Result*

Dibawa ini merupakan tampilan implementasi tabel *Payment* berisi kolom yang nantinya berisikan data yang inputkan.

	#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra
	1	payment_id	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
	2	user_id	int(11)			Tidak	*Tidak ada*		
	3	tournament_id	int(11)			Tidak	*Tidak ada*		
	4	fee	int(11)			Tidak	*Tidak ada*		
	5	expired_date	date			Ya	*NULL*		
	6	rekening_id	int(11)			Ya	*NULL*		
	7	bank_name	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
	8	payment_number	int(20)			Ya	*NULL*		
	9	payment_name	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
	10	payment_date	date			Ya	*NULL*		
	11	proof	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
	12	status	varchar(255)	latin1_swedish_ci		Ya	*NULL*		
	13	dateupdate	timestamp			Ya	*NULL*		

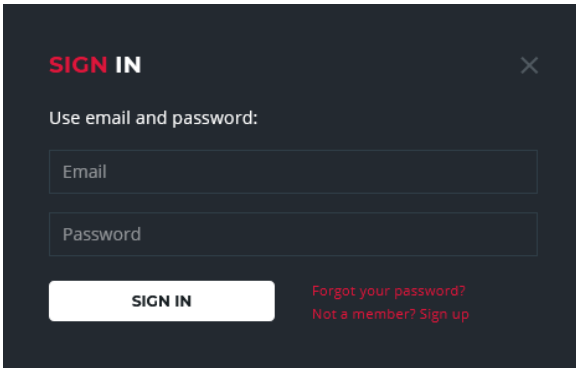
Gambar 4.8 Tabel *Payment*

4.2.2 Implementasi Sistem *User Interface Website*

Implementasi sistem pada bagian ini, akan memperlihatkan bagaimana tampilan yang muncul pada saat *user* mengakses aplikasi *presensi marketing*, Implementasi ini meliputi:

1. Tampilan Halaman Login

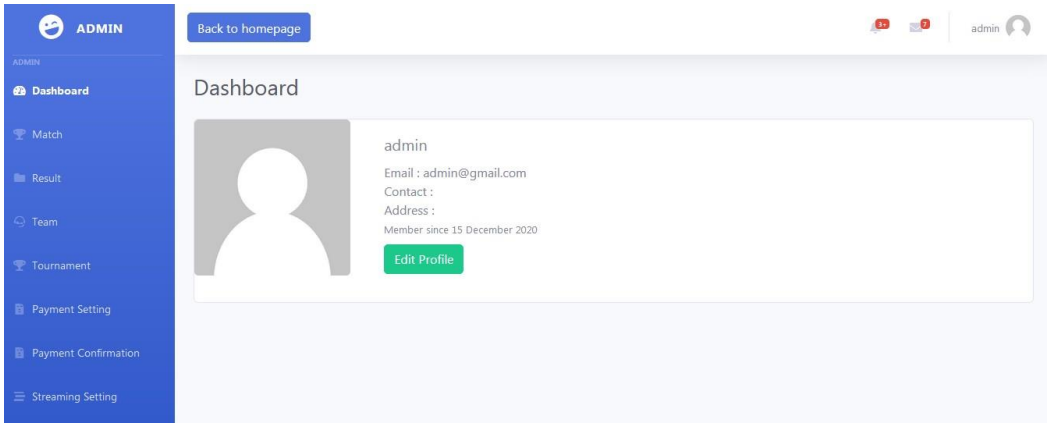
   Berikut ini adalah tampilan halaman login pada aplikasi *website*



Gambar 4.1. *Login Android*

2. Tampilan Menu Utama

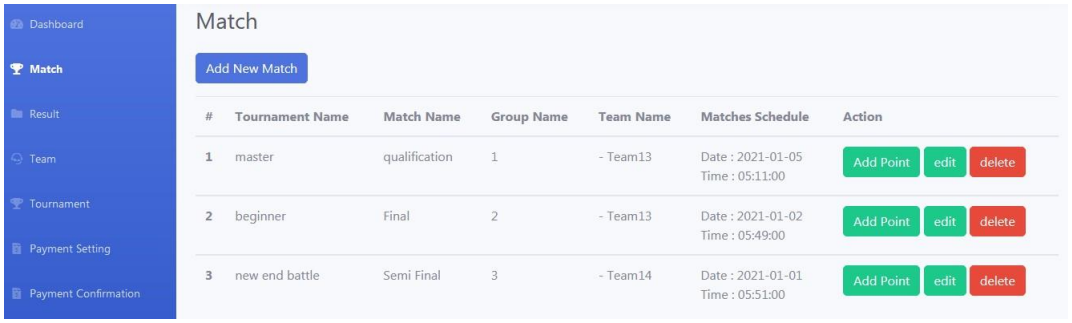
Halaman ini adalah utama untuk mengakses beberapa menu dan juga terdapat inputan untuk mengelola seluruh data yang ada pada aplikasi.



Gambar 4.2. Menu Utama *Administrator*

3. Tampilan Halaman *Match*

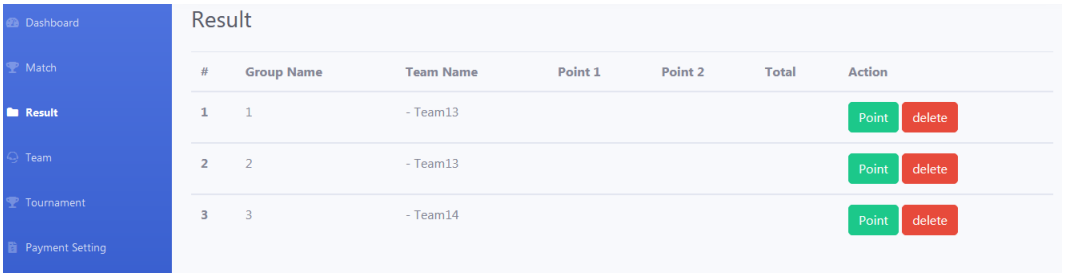
Halaman *Match* merupakan halaman yang dipersiapkan untuk menampung seluruh data pertandingan pada saat tournament, *Administrator* akan menginputkan data turnamen seperti *tournament name, match name, team name, match es schedule.*



Gambar 4.3 Tampilan Halaman *Match*

4. Tampilan Halaman *Result*

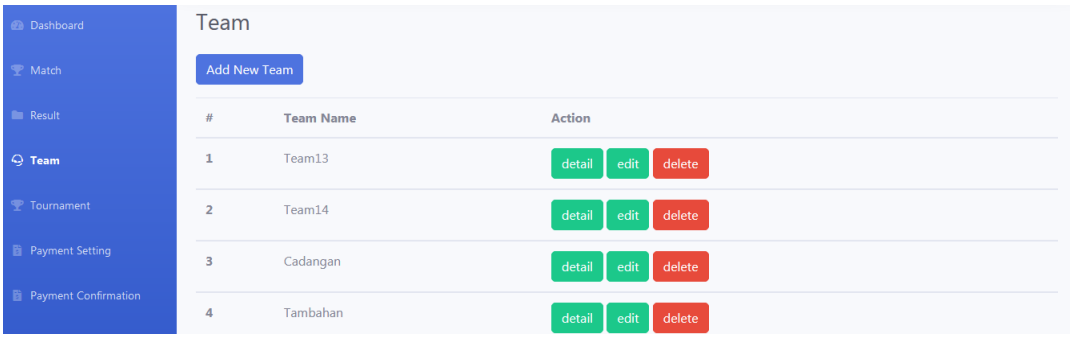
Halaman *Result* dibuat untuk menampilkan hasil akhir disetiap pertandingan yang ada di turnamen *The Pillars E-sport*, adapun kolom yang di tampilkan yaitu *group name, team name*, poin *1*, poin *2*, poin *3, total.*



Gambar 4.3 Tampilan Halaman *Result*

5. Tampilan Halaman *Team*

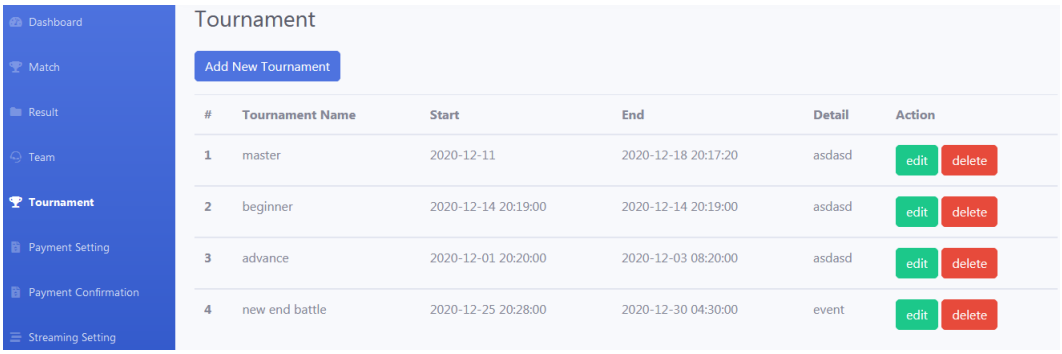
Halaman *Team* ini dibuat untuk memasukan data *Team* yang akan ikut serta dalam pertandingan turnamen yang diselenggarakan.



Gambar 4.4 Tampilan Halaman *Team*

6. Tampilan Halaman *Tournament*

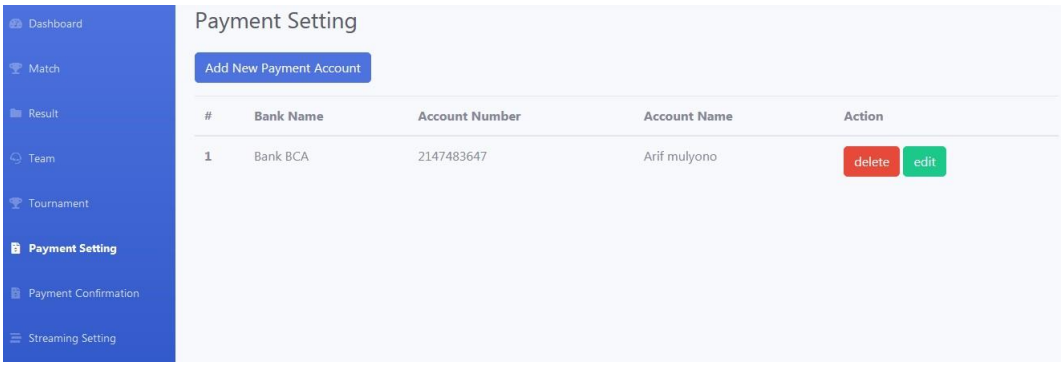
Halaman ini dibuat untuk mengelola data *tournament* pada aplikasi, adapun kolom isian dan tampilan meliputi: *tournament name, start, end* dan *detail.*



Gambar 4.5 Tampilan Halaman *Tournament*

7. Tampilan Halaman *Payment Setting*

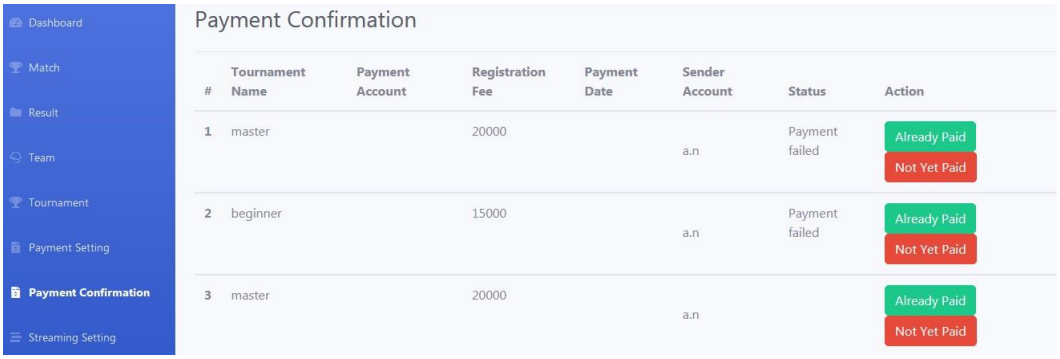
Halaman ini dibuat untuk mengelola *payment setting* atau pembayaran pada aplikasi, sehingga *Administrator* dapat menentukan detail pembayaran untuk calon peserta tournament.



Gambar 4.6 Tampilan Halaman *Payment Setting*

8. Tampilah Halaman *Payment Confirmation*

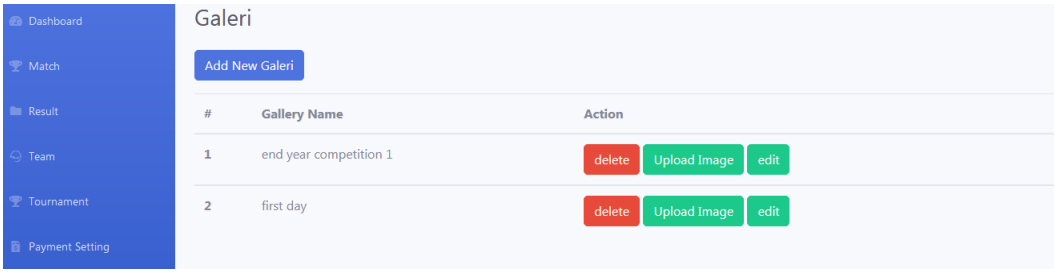
Dibawah ini merupakan halaman *payment confirmation* dimana *Administrator* akan mengkonfirmasi setiap pembayaran para peserta turnamen sebelum mengikuti pertandingan.



Gambar 4.7 Tampilan Halaman *Payment Confirmation*

9. Tampilan Halaman *Galeri*

Halaman *Galeri* merupakan halaman dimana *Administrator* dapat mengelola data galeri untuk menyimpan informasi yang akan di publis di aplikasi *website.*



Gambar 4.8 Tampilan Halaman *Galeri*

4.3 Pengujian Sistem

Hasil pengujian dilakukan dengan menggunakan metode Black Box Testing. Metode black box memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian ini berusaha menemukan kesalahan dalam kategori fungsi-fungsi yang tidak benar atau hilang, kesalahan interface, kesalahan dalam struktur data atau akses basis data eksternal, kesalahan kinerja, dan inisialisasi dan kesalahan terminal. Dengan menggunakan metode pengujian black box, perekayasa sistem dapat menemukan kesalahan dalam kategori sebagai berikut:

1. Fungsi tidak benar atau hilang
2. Kesalahan antar muka
3. Kesalahan pada struktur data (pengaksesan *database*)
4. Kesalahan inisialisasi dan akhir program
5. Kesalahan kinerja

Berikut pengujian yang dilakukan terhadap sistem dengan metode blackbox testing:

4.3.1 Pengujian untuk aplikasi berbasis *web*

Pengujian ini di lakukan untuk aplikasi berbasis *web* yang di gunakan oleh *The Pillars E-sport* berikut beberapa pengujianya:

1. Pengujian *Login*

Tabel 4.1. Hasil Pengujian Sistem *Login*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	*Input user* dan *password* benar	Masuk Ke Sistem Dan Dapat Mengakses Menu Utama	Masuk Menu Utama
2	*Input user* dan *password* salah	Tidak Dapat Masuk Ke Sistem Dan Kembali Kehalaman *Login*	*Login* Gagal

2. Pengujian *Match*

Tabel 4.2. Hasil Pengujian *Match*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	*Select Option Tournament Name*	Dapat memilih *Tournament Name* pada halaman *Add New Match*	Berhasil Memunculkan *Tournament Name*
2	*Input Match Name*	Dapat menginput *Match name* pada kolom isian	Berhasil memasukan *Match name*
3	*Select Option Group Name*	Dapat memilih *Group Name* pada halaman *Add New Match*	Berhasil menampilkan *Group Name*
4	*Select Option Team*	Dapat memilih *Team* pada halaman *Add New Match*	Berhasil menampilkan *Tem Select*
5	*Select Date Matches Schedule*	Dapat memilih tanggal *match* pada halaman *Add New Match*	Berhasil menampilkan *Match Schedule*

3. Pengujian *Result*

Tabel 4.3. Hasil Pengujian *Result*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	Tampil Halaman *Result*	Dapat menampilkan Halaman Utama *Result*	Berhasil menampilkan halaman *Result*
2	*Aksi Tombol* Poin	Dapat menampilkan hasil poin setiap tim	Berhasil menampilkan Poin setiap tim
3	*Aksi Tombol Delete*	Dapat menghapus poin tim yang di delet	Berhasil melakukan delete poin tim yang di delet

4. Pengujian *Team*

Tabel 4.4. Hasil Pengujian *Team*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	Tampil Halaman *Team*	Dapat menampilkan Halaman Utama *Team*	Berhasil menampilkan halaman *Team*
2	*Aksi Tombol Add Team*	Dapat menginput dan menampilkan halaman *input team*	Berhasil menampilkan halaman *Add new team*
3	*Aksi Tombol Detail*	Dapat menampilkan Detail *Team* yang berisikan data peserta	Berhasil menampilkan Detail *Team*
4	*Aksi Tombol Edit*	Dapat mengedit Data Nama *Team*	Berhasil mengedit nama *Team*
5	*Aksi Tombol Delete*	Dapat menghapus data *team* yang akan di hapus.	Berhasil menghapus data *team* yang di pilih

5. Pengujian *Tournament*

Tabel 4.5. Hasil Pengujian *Tournament*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	Tampil Halaman *Tournament*	Dapat menampilkan Halaman Utama *Tournament*	Berhasil menampilkan halaman *Tournament*
2	*Aksi Tombol Add Tournament*	Dapat menginput dan menampilkan halaman *input tournament*	Berhasil menampilkan halaman *Add new tournament*
3	*Aksi Tombol Edit*	Dapat mengedit Data *Tournament* yang di pilih	Berhasil mengedit detail *tournament*
4	*Aksi Tombol Delete*	Dapat menghapus data *tournament* yang akan di pilih.	Berhasil menghapus data *team* yang di pilih

6. Pengujian *Payment Setting*

Tabel 4.6. Hasil Pengujian *Payment Setting*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	Tampil Halaman *Payment Setting*	Dapat menampilkan Halaman Utama *Payment Setting*	Berhasil menampilkan halaman *Payment Setting*
2	*Aksi Tombol Add New Payment Account*	Dapat menginput Kolom Isian *Add New Account Payment*	Berhasil menginput data *New Account Payment*
3	*Aksi Tombol Edit*	Dapat mengedit Data *Payment Account* yang di pilih	Berhasil mengedit detail *Payment Account*
4	*Aksi Tombol Delete*	Dapat menghapus data *Payment Account* yang akan di pilih.	Berhasil menghapus data *Payment Account* yang di pilih

7. Pengujian *Payment Confirmation*

Tabel 4.7. Hasil Pengujian *Payment Confirmation*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	Tampil Halaman *Payment Confirmation*	Dapat menampilkan Halaman Utama *Payment Confirmation*	Berhasil menampilkan halaman *Payment Confirmation*
2	*Aksi Tombol Already Payment*	Dapat mengkonfirmasi pembayaran dengan menekan tombol *Already Payment*	Berhasil Mengkonfirmasi Pembayaran
3	*Aksi Tombol Not Yet Paid*	Dapat mengcancel konfirmasi pembayaran dengan menekan tombol *Not Yet Paid*	Berhasil Cancel Konfirmasi

8. Pengujian *Gallery*

Tabel 4.8. Hasil Pengujian *Gallery*

No	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian
1	Tampil Halaman *Gallery*	Dapat menampilkan Halaman Utama *Gallery*	Berhasil menampilkan halaman *Gallery*
2	*Aksi Tombol Add New Gallery*	Dapat menambahkan *Gallery Name Baru*	Berhasil menambahkan *Gallery Name* terbaru
3	*Aksi Tombol Edit*	Dapat mengedit nama *Gallery* yang dipilih	Berhasil Edit detail *Gallery*
4	*Aksi Tombol Upload Image*	Dapat mengupload Gambar	Berhasil mengupload gambar
5	*Aksi Tombol Delete*	Dapat menghapus Detail *Gallery* yang di pilih	Berhasil menghapus data *Gallery*

4.3.2 Kesimpulan Pengujian Menggunakan *Black Box*

Berdasarkan hasil pengujian sistem dengan metode pengujian *black box* dapat disimpulkan bahwa sistem telah berjalan sesuai kebutuhan dan rancangan awal juga fitur didalamnya telah berjalan sesuai dengan fungsi yang diharapkan.

BAB V
PENUTUP

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari pembuatan aplikasi pendaftaran *online* pada pada *The Pillars E-sport* sebagai berikut:

a. Aplikasi *website* ini mempermudah panitia kompetisi untuk mengarsip data pendaftaran peserta dan pembayaran pendaftaran.

b. Aplikasi *website* ini dapat mempermudah calon peserta dapat mengetahui informasi mengenai proses turnamen yang sedang berlangsung maupun yang akan berlangsung.

c. Dengan menggunakan aplikasi *website* maka calon peserta yang mendaftar tidak terbatas oleh tempat dan waktu.

5.2 Saran

Dalam pembuatan aplikasi pendaftaran *online* pada *The Pillars E-sport*

1. Dalam pengembangan aplikasi selanjutnya dapat ditingkatkan hingga menjadi sistem turnamen yang lebih baik.
2. Aplikasi *website* ini juga dapat dikembangkan sebagai media berita mengenai *E-sport*s dan *The Pillars* itu sendiri.
3. Pengembangan desain template pada aplikasi pendaftaran turrnamen *online* pada *The Pillars E-sport* agar bisa lebih menggambarkan *The Pillars* itu sendiri.
4. Menyediakan sistem turnamen *online* untuk divisi *game* dengan genre yang berbeda seperti moba, mmorpg, rpg dan lain lain.

DAFTAR PUSTAKA

Adams & Rollings, 2010, *Fundamentals Of Game Design, Barkeley*

Arbani, Marizka. 2011, Pengembangan Sistem Informasi Sekolah Berbasis *Web* (studi kasus: MI An-Nizhomiyah Depok) Fakultas Sains dan Teknologi, Universitas Islam Negeri Syarif Hidayatullah, Jakarta

A. S, Rosa., dan Shalahuddin, 2013, Rekayasa Perangkat Lunak Terstruktur dan Berorientiasi Objek, Andi, Bandung

Bunafit Nugroho., 2013, Dasar Pemrograman *Web PHP – MYSQL* dengan *Dreamweaver*. Yogyakarta: Gava Media.

Elisabeth Yunaeti Anggraeni dan Rita Irviani., 2017. Pengantar Sistem Informasi. Yogyakarta: ANDI.

Fauzi, D. M. (2019). Pengaruh Layanan Bimbingan Kelompok Teknik Permainan Simulasi dalam Mengambangkan Ketangguhan Diri (Resiliensi) Atlet Esport. *Kesejahteraan Sosial Pengembangan Masyarakat Islam Psikologi*

Bimbingan Dan Konseling Islam, *2*(1), 8. https://doi.org/10.1007/s10681-019-2458-6%0Ahttps://www.researchgate.net/publication/273059476%0Ahttps://learni ng.hccs.edu/faculty/joy.marshall/biol-2320-microbiology-lecture-notes/chapter-3-lecture-notes%0Awww.gscience.net%0Ahttp://www.jim.unsyiah.ac.id/

Indrajani., 2011, Perancangan Basis Data dalam All in 1. Jakarta: PT. Elex Media Komputindo.

Juniardi Dermawan, S. H., & Sistem. (2017). IMPLEMENTASI MODEL WATERFALL PADA PENGEMBANGAN SISTEM INFORMASI PERHITUNGAN NILAI MATA PELAJARAN BERBASIS WEB PADA SEKOLAH DASAR AL-AZHAR SYIFA BUDI JATIBENING. *Notes and Queries*, *19–2*(159), 143. https://doi.org/10.1093/nq/s5-VII.159.37-a

Luthfiani, N 2017, Pengembangan Officeial *Site* eLearning Plus Menggunakan Strategi Digital Marketing Untuk Meningkatkan Promosi Pada Perguruan Tinggi, Tanggerang

Madcoms. 2011. Aplikasi *Web Database* dengan Dreamweaver dan *PHPMYSQL*. Yogyakarta: Andi Offset.

Theresia, L 2010, Hubungan Kecanduan *Game Online* dengan Prestasi Akademik Mahasiswa di Fakultas Teknik Universitas Indonesia. Depok

DAFTAR LAMPIRAN

DATA NARASUMBER

NO	NAMA	JABATAN
1	Raihan Muhammad	Chief Executive Officer
2	Kindi Dwiwenda	Chief Growth Officer
3	Bio	Publisher & Tournament Manager

DAFTAR PERTANYAAN WAWANCARA

Daftar pertanyaan wawancara ini berfungsi untuk menjawab rumusan masalah pada penelitian yang berjudul **"RANCANG BANGUN APLIKASI PENDAFTARAN *ONLINE* TURNAMEN *PLAYER UNKOWN BATTLE GROUND* DI *THE PILLARS E-SPORT* BERBASIS *WEB*"**. Berikut daftar pertanyaan wawancara untuk menjawab rumusan masalah bagaimana implementasi rancang bangun aplikasi pendaftaran *online turnamen player unknown battle ground* di *The Pillars*s *E-sport*.

Daftar pertanyaan:

1. Bagaimana sejarah *The Pillars E-sport*?
2. Apa tujuan *The Pillars E-sport*?
3. Apa alasan *The Pillars* yang tadinya hanya komunitas *game* berubah menjadi organisasi *E-sport*?
4. Divisi *game* apa saja yang ada pada *The Pillars E-sport*?
5. Bagaimana sistem turnamen yang saat ini sedang berjalan pada *The Pillars E-sport*?
6. Bagaimana prosedur pendaftaran turnamen di *The Pillars E-sport*?
7. Apa saja persyaratan yang harus di penuhi bagi yang ingin mengikuti turnamen di *The Pillars E-sport*?
8. Laporan apa saja yang di hasilkan dalam proses turnamen di *The Pillars E-sport*?
9. Apa saja masalah yang pernah terjadi ketika proses pendaftaran turnamen dan pelaksanaan turnamennya itu berlangsung?

10. Apa yang harus dilakukan tim peserta untuk menjadi tim pemenang peserta turnamen?
11. Bagaimana proses pemberian hadiah pemenang turnamen?

Laporan Penelitian

(Hasil Interview)

Tanggal : 12 Desember 2020

Waktu : 09.00 –10.00 Wib

Narasumber : Raihan Muhammad

Jabatan : *Chief Executive Officer*

1. Bagaimana sejarah *The Pillars E-sport*?
2. Tujuan *The Pillars*s *E-sport*?
3. Apa alasan *The Pillars*s yang tadinya hanya komunitas *game* memutuskan untuk merubah menjadi organisasi *E-sport*?

Jawaban:

1. *The Pillars* terbentuk sejak 2003 pada saat trend *game online booming* di Indonesia. Dimulai dari *game* Ragnarok *Online* muncul kami membuat komunitas yang cukup berprestasi dan disegani, puncak dari prestasi komunitas ini yaitu pada saat mencapai *Best Guild Record* 32 Periode.
2. 16 Tahun berkomunitas hingga sekarang kami masih berkumpul dengan passion yang sama, sampai akhirnya kami membentuk tim *Esports* yang bertujuan memberikan manfaat yang lebih untuk Generasi yang akan datang.

3. Ketika *The Pillars* sudah mempunyai banyak anggota, kami melihat bahwa ada bakat dan potensi dari member *The Pillars* untuk bisa bergabung di level oraganisasi *game* berikutnya yaitu *E-sport*. Pada waktu itu para member yang semakin banyak pun turut mendukung *The Pillars* untuk bisa melakukan kegiatan *E-sport* dan ikut bersaing dalam *Competitive Scene E-sport*.

Laporan Penelitian

(Hasil Interview)

Tanggal : 13 Desember 2020

Waktu : 09.00 –10.00 Wib

Narasumber : Kindi Dwiwendha

Jabatan : *Chief Growth Officer*

1. Divisi *game* apa saja yang ada pada *The Pillars E-sport*?
2. Bagaimana sistem turnamen yang saat ini sedang berjalan pada *The Pillars E-sport*?
3. Bagaimana prosedur pendaftaran turnamen di *The Pillars E-sport*?
4. Apa saja persyaratan yang harus di penuhi bagi yang ingin mengikuti turnamen di *The Pillars E-sport*?

Jawaban:

1. Divisi *game* yang ada pada *The Pillars E-sport* antara lain: *Player Unknown Battle Ground, Call Of Duty Mobile, Mobile Legend, Free Fire.* Dan untuk saat ini yang paling aktiv dalam melakukan kegiatan *E-sport* yaitu divisi *Player Unknown Battle Ground.*
2. Sistem yang kami gunakan adalah berdasarkan sistem yang di gunakan pada turnamen *game* dari *official game* nya itu sendiri. Setiap season dari

masing masing *game* juga rata rata selalu ada perubahan sistem, baik dari perhitungan *point* maupun *rule gameplay* nya.

3. Calon peserta turnamen melakukan pendaftaran melalui *Game House* dan *no whatsapp* yang sudah di sediakan oleh panitia pendaftaran turnamen yang kemudian calon peserta turnamen mengisi form biodata dari masing masing member dari tim calon peserta turnamen, serta menyediakan dokumen pendukung. Panitia pendaftaran turnamen memberikan data calon peserta turnamen yang sudah memenuhi syarat kepada panitia turnamen untuk kemudian di proses ke tahap pelaksanaan turnamen.
4. Persyaratannya adalah semua member dari setiap tim memiliki 5 player dengan satu member sebagai kapten tim di dalamnya, masing masing pendaftar harus di pastikan sehat jasmani dan rohani, dan usia peserta minimal 17 tahun.

Laporan Penelitian

(Hasil Interview)

Tanggal : 12 Desember 2020

Waktu : 10.00 –11.00 Wib

Narasumber : Bio

Jabatan : *Publisher & Tournament Manager*

1. Laporan apa saja yang di hasilkan dalam proses turnamen di *The Pillars E-sport*?
2. Apa saja masalah yang pernah terjadi ketika proses pendaftaran turnamen dan pelaksanaan turnamennya itu berlangsung?
3. Apa yang harus dilakukan tim peserta untuk menjadi tim pemenang peserta turnamen?
4. Bagaimana proses pemberian hadiah pemenang turnamen?

Jawaban:

1. Laporan dihasilkan adalah berupa *point point* yang di dapatkan oleh masing masing tim dalam match yang sudah di pertandingan yang dimana *point* tersebut menjadi acuan bagi masing masing tim peserta turnamen untuk bisa menjuarai turnamen tersebut.
2. Masalah yang sering terjadi ketika dalam proses pendataran adalah terjadinya antrian Panjang terutama ketika proses pendaftaran dilakukan di

Game House The Pillars E-sport, sehingga mengakibatkan banyak waktu yang terbuang percuma. Sedangkan masalah ketika proses turnamen berlangsung berupa terjadinya kesalahan pencatatan *point*, hilangnya arsip data, tempat yang tidak bisa menampung semua peserta turnamen, sehaingga masalah tersebut sangat mengganggu proses berjalannya turnamen.

3. Para tim peserta harus berlomba lomba mendapatkan *point* untuk bisa menjuarai tunamen tersebut. *Point* bisa di dapatkan dengan cara memenangkan setiap match yang ada, dan juga *point* bisa di dapatkan dengan cara melakukan *secure kill* kepada tim peserta musuh. Untuk aturan *point* tersebut kita adopsi dari aturan yang berlaku pada *Official* turnamen *Game* itu sendiri.
4. Proses pemberian hadiah turnamen akan di lakukan ketika pemenang sudah di tentukan. Maksimal hadiah akan di berikan 1 minggu setelah pemenang turnamen di tentukan. Hadiahnya bisa berupa uang tunai, piala dan sertifikat turnamen.

Lampiran *Script* Program

```
/*

SQLyog Ultimate v11.11 (64 bit)

MySQL - 5.5.5-10.1.26-MariaDB : Database - esport

*********************************************************************

***

*/



/*!40101 SET NAMES utf8 */;



/*!40101 SET SQL_MODE=''*/;



/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,

UNIQUE_CHECKS=0 */;
```

```
/*!40014 SET
@OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

CREATE DATABASE /*!32312 IF NOT EXISTS*/`esport` /*!40100 DEFAULT
CHARACTER SET latin1 */;

USE `esport`;

/*Table structure for table `galeri` */

DROP TABLE IF EXISTS `galeri`;

CREATE TABLE `galeri` (
  `galeri_id` int(11) NOT NULL AUTO_INCREMENT,
  `image` varchar(255) DEFAULT NULL,
  `title_id` int(11) DEFAULT NULL,
  `detail` text,
  PRIMARY KEY (`galeri_id`),
  KEY `title_id` (`title_id`),
```

```
  CONSTRAINT `galeri_ibfk_1` FOREIGN KEY (`title_id`) REFERENCES
`galeri_title` (`title_id`)

) ENGINE=InnoDB AUTO_INCREMENT=27 DEFAULT CHARSET=latin1;

/*Data for the table `galeri` */

insert  into `galeri`(`galeri_id`,`image`,`title_id`,`detail`) values
(20,'Eq3992CUUAEjzOL8.jpg',5,'2'),(21,'Eq3992CUUAEjzOL9.jpg',5,'2'),(22,'Eq
3992CUUAEjzOL10.jpg',5,'2'),(24,'Eq3992CUUAEjzOL12.jpg',2,'asdasd'),(26,'E
q3992CUUAEjzOL14.jpg',2,'6');

/*Table structure for table `galeri_title` */

DROP TABLE IF EXISTS `galeri_title`;

CREATE TABLE `galeri_title` (

  `title_id` int(11) NOT NULL AUTO_INCREMENT,

  `title_name` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`title_id`)

) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

/*Data for the table `galeri_title` */
```

```
insert  into `galeri_title`(`title_id`,`title_name`) values (2,'end year competition 1'),(5,'first day');

/*Table structure for table `group` */

DROP TABLE IF EXISTS `group`;

CREATE TABLE `group` (
  `group_id` int(11) NOT NULL AUTO_INCREMENT,
  `group_name` varchar(125) DEFAULT NULL,
  `tournament_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`group_id`),
  KEY `tournament_id` (`tournament_id`),
  CONSTRAINT `group_ibfk_1` FOREIGN KEY (`tournament_id`) REFERENCES `tournament` (`tournament_id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;

/*Data for the table `group` */

insert  into `group`(`group_id`,`group_name`,`tournament_id`) values (1,'A',21),(2,'B',21),(3,'C',21),(4,'D',21);
```

```
/*Table structure for table `matchup` */

DROP TABLE IF EXISTS `matchup`;

CREATE TABLE `matchup` (
  `match_id` int(11) NOT NULL AUTO_INCREMENT,
  `match_name` varchar(255) DEFAULT NULL,
  `tournament_id` int(11) NOT NULL,
  `group_id` int(11) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `time` time DEFAULT NULL,
  `match_detail` varchar(32) NOT NULL,
  `sort` int(2) DEFAULT '0',
  `timemodified` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `team_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`match_id`),
  KEY `group_id` (`group_id`),
  KEY `team_id` (`team_id`),
```

```
  CONSTRAINT `matchup_ibfk_1` FOREIGN KEY (`group_id`) REFERENCES
`group` (`group_id`),

  CONSTRAINT `matchup_ibfk_2` FOREIGN KEY (`team_id`) REFERENCES
`team` (`team_id`)

) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1;

/*Data for the table `matchup` */

insert  into
`matchup`(`match_id`,`match_name`,`tournament_id`,`group_id`,`date`,`time`,`m
atch_detail`,`sort`,`timemodified`,`team_id`) values (15,'qualification',11,1,'2021-
01-05','05:11:00','',0,'2021-01-02 05:11:45',13),(18,'Final',18,2,'2021-01-
02','05:49:00','',0,'2021-01-02 05:49:53',13),(19,'Semi Final',21,3,'2021-01-
01','05:51:00','',0,'2021-01-02 05:51:34',14);

/*Table structure for table `payment` */

DROP TABLE IF EXISTS `payment`;

CREATE TABLE `payment` (

  `payment_id` int(11) NOT NULL AUTO_INCREMENT,

  `user_id` int(11) NOT NULL,

  `tournament_id` int(11) NOT NULL,
```

```
  `fee` int(11) NOT NULL,
  `expired_date` date DEFAULT NULL,
  `rekening_id` int(11) DEFAULT NULL,
  `bank_name` varchar(255) DEFAULT NULL,
  `payment_number` int(20) DEFAULT NULL,
  `payment_name` varchar(255) DEFAULT NULL,
  `payment_date` date DEFAULT NULL,
  `proof` varchar(255) DEFAULT NULL,
  `status` varchar(255) DEFAULT NULL,
  `dateupdate` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`payment_id`),
  KEY `user_id` (`user_id`),
  KEY `rekening_id` (`rekening_id`),
  CONSTRAINT `payment_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES
`user` (`user_id`),
  CONSTRAINT `payment_ibfk_2` FOREIGN KEY (`rekening_id`)
REFERENCES `rekening` (`rekening_id`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=latin1;

/*Data for the table `payment` */

insert  into
`payment`(`payment_id`,`user_id`,`tournament_id`,`fee`,`expired_date`,`rekening
_id`,`bank_name`,`payment_number`,`payment_name`,`payment_date`,`proof`,`st
atus`,`dateupdate`) values (14,14,11,20000,'2021-01-
13',NULL,NULL,NULL,NULL,NULL,NULL,'Payment
failed',NULL),(15,14,18,15000,'2021-01-
14',NULL,NULL,NULL,NULL,NULL,NULL,'Payment
failed',NULL),(16,14,11,20000,'2021-01-
16',NULL,NULL,NULL,NULL,NULL,NULL,'',NULL),(17,14,11,20000,'2021-
01-16',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);

/*Table structure for table `players` */

DROP TABLE IF EXISTS `players`;

CREATE TABLE `players` (
  `player_id` int(11) NOT NULL AUTO_INCREMENT,
  `player_name` varchar(32) NOT NULL,
  `player_nickname` varchar(32) NOT NULL,
  `team_id` int(11) NOT NULL,
```

```
  `player_detail` text,
  `image` varchar(255) DEFAULT NULL,
  `id_game` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`player_id`),
  KEY `team_id` (`team_id`),
  CONSTRAINT `players_ibfk_1` FOREIGN KEY (`team_id`) REFERENCES
`team` (`team_id`)
) ENGINE=InnoDB AUTO_INCREMENT=38 DEFAULT CHARSET=latin1;

/*Data for the table `players` */

insert  into
`players`(`player_id`,`player_name`,`player_nickname`,`team_id`,`player_detail`,`
image`,`id_game`) values
(1,'John','nevada',13,'',NULL,NULL),(2,'Cage','V',13,'',NULL,NULL),(3,'Done','as
d',13,'',NULL,NULL),(4,'Hendra','King',13,'',NULL,NULL),(5,'Abdul','Vin',13,'',
NULL,NULL),(37,'asdasd','tes1',14,NULL,'nama15183701691.jpg','ttt');

/*Table structure for table `rekening` */

DROP TABLE IF EXISTS `rekening`;
```

```
CREATE TABLE `rekening` (
  `rekening_id` int(11) NOT NULL AUTO_INCREMENT,
  `bank_name` varchar(32) DEFAULT NULL,
  `payment_number` int(11) DEFAULT NULL,
  `payment_name` varchar(32) DEFAULT NULL,
  `post_date` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`rekening_id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;

/*Data for the table `rekening` */

insert  into
`rekening`(`rekening_id`,`bank_name`,`payment_number`,`payment_name`,`post
_date`) values (2,'Bank BCA',2147483647,'Arif mulyono',NULL);

/*Table structure for table `result` */

DROP TABLE IF EXISTS `result`;

CREATE TABLE `result` (
  `point_id` int(11) NOT NULL AUTO_INCREMENT,
  `point1` int(11) NOT NULL,
```

```
  `point2` int(11) NOT NULL,
  `total` int(11) NOT NULL,
  `team_id` int(11) NOT NULL,
  `match_id` int(11) NOT NULL,
  PRIMARY KEY (`point_id`),
  KEY `team_id` (`team_id`),
  KEY `match_id` (`match_id`),
  CONSTRAINT `result_ibfk_1` FOREIGN KEY (`team_id`) REFERENCES
`team` (`team_id`),
  CONSTRAINT `result_ibfk_2` FOREIGN KEY (`match_id`) REFERENCES
`matchup` (`match_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*Data for the table `result` */
/*Table structure for table `store` */
DROP TABLE IF EXISTS `store`;
CREATE TABLE `store` (
  `store_id` int(11) NOT NULL AUTO_INCREMENT,
```

```
  `category` varchar(255) DEFAULT NULL,
  `product_name` varchar(255) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `detail` text,
  `price` int(255) DEFAULT NULL,
  `link_product` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`store_id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

/*Data for the table `store` */

insert  into
`store`(`store_id`,`category`,`product_name`,`image`,`detail`,`price`,`link_product
`) values (1,'1','1','default.jpg','
        1111',11,'11'),(5,'tes','tes','default.jpg','23234234',12,'https://www.tokopedi
a.com/bjrcomputindo/keyboard-asus-x453-x453m-x455l-x453ma-x453s-x451-
x451c-x451-
f401e?utm_campaign=Product%20Share&utm_source=Desktop&utm_medium=S
hare&_branch_match_id=868507866037547024');
```

```
/*Table structure for table `stream` */

DROP TABLE IF EXISTS `stream`;

CREATE TABLE `stream` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `link` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

/*Data for the table `stream` */

insert  into `stream`(`id`,`link`) values
(1,'https://www.youtube.com/embed/pfW6zBhhgvM');

/*Table structure for table `team` */

DROP TABLE IF EXISTS `team`;

CREATE TABLE `team` (
  `team_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `team_name` varchar(32) NOT NULL,
  `ranking_id` varchar(32) NOT NULL,
```

```
  `player_id` int(11) DEFAULT NULL,
  `playerfullname` varchar(255) DEFAULT NULL,
  `playernickname` varchar(255) DEFAULT NULL,
  `contact` int(255) DEFAULT NULL,
  `foto` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`team_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `team_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES
`user` (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=24 DEFAULT CHARSET=latin1;
/*Data for the table `team` */
insert  into
`team`(`team_id`,`user_id`,`team_name`,`ranking_id`,`player_id`,`playerfullname
`,`playernickname`,`contact`,`foto`) values
(13,13,'Team13','',NULL,NULL,NULL,NULL,NULL),(14,14,'Team14','',NULL,
NULL,NULL,NULL,NULL),(20,14,'Cadangan','',NULL,NULL,NULL,NULL,N
ULL),(22,14,'Tambahan','',NULL,NULL,NULL,NULL,NULL),(23,14,'Tambahan
2','',NULL,NULL,NULL,NULL,NULL);
```

```
/*Table structure for table `tournament` */

DROP TABLE IF EXISTS `tournament`;

CREATE TABLE `tournament` (
  `tournament_id` int(11) NOT NULL AUTO_INCREMENT,
  `tournament_name` varchar(255) DEFAULT NULL,
  `tournament_start` date DEFAULT NULL,
  `time_start` time DEFAULT NULL,
  `tournament_end` date DEFAULT NULL,
  `time_end` time DEFAULT NULL,
  `tournament_detail` text,
  `registration_fee` int(11) DEFAULT NULL,
  `slot` int(11) DEFAULT NULL,
  `reward` int(255) DEFAULT NULL,
  PRIMARY KEY (`tournament_id`)
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=latin1;
```

```
/*Data for the table `tournament` */

insert  into
`tournament`(`tournament_id`,`tournament_name`,`tournament_start`,`time_start`,
`tournament_end`,`time_end`,`tournament_detail`,`registration_fee`,`slot`,`reward
`) values (11,'master','2020-12-11',NULL,'2020-12-
18','20:17:20','asdasd',20000,NULL,NULL),(18,'beginner','2020-12-
14','20:19:00','2020-12-
14','20:19:00','asdasd',15000,NULL,NULL),(19,'advance','2020-12-
01','20:20:00','2020-12-03','08:20:00','asdasd',0,NULL,NULL),(21,'new end
battle','2020-12-25','20:28:00','2020-12-30','04:30:00','event
',120000,NULL,NULL);

/*Table structure for table `tournament_register` */

DROP TABLE IF EXISTS `tournament_register`;

CREATE TABLE `tournament_register` (
  `register_id` int(11) NOT NULL AUTO_INCREMENT,
  `tournament_id` int(11) DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  `team_id` int(11) DEFAULT NULL,
```

```
  `payment_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`register_id`),
  KEY `tournament_id` (`tournament_id`),
  KEY `user_id` (`user_id`),
  KEY `team_id` (`team_id`),
  KEY `payment_id` (`payment_id`),
  CONSTRAINT `tournament_register_ibfk_1` FOREIGN KEY
(`tournament_id`) REFERENCES `tournament` (`tournament_id`),
  CONSTRAINT `tournament_register_ibfk_2` FOREIGN KEY (`user_id`)
REFERENCES `user` (`user_id`),
  CONSTRAINT `tournament_register_ibfk_3` FOREIGN KEY (`team_id`)
REFERENCES `team` (`team_id`),
  CONSTRAINT `tournament_register_ibfk_4` FOREIGN KEY (`payment_id`)
REFERENCES `payment` (`payment_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*Data for the table `tournament_register` */
/*Table structure for table `user` */
```

```
DROP TABLE IF EXISTS `user`;

CREATE TABLE `user` (

  `user_id` int(11) NOT NULL AUTO_INCREMENT,

  `name` varchar(255) NOT NULL,

  `email` varchar(255) NOT NULL,

  `contact` varchar(255) DEFAULT NULL,

  `alamat` text,

  `image` varchar(255) NOT NULL,

  `password` varchar(255) NOT NULL,

  `level_id` int(11) NOT NULL,

  `is_active` int(1) NOT NULL,

  `date_created` int(11) NOT NULL,

  PRIMARY KEY (`user_id`),

  UNIQUE KEY `UNIQUE` (`email`),

  KEY `level_id` (`level_id`),

  CONSTRAINT `user_ibfk_1` FOREIGN KEY (`level_id`) REFERENCES

`user_level` (`level_id`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=latin1;

/*Data for the table `user` */

insert  into
`user`(`user_id`,`name`,`email`,`contact`,`alamat`,`image`,`password`,`level_id`,`i
s_active`,`date_created`) values
(13,'admin','admin@gmail.com',NULL,NULL,'default.jpg','$2y$10$2lAqxrcDLR
nf.TgqLCCkBed6YuCrWY69rEidcaVH7gSWGHqNql8tm',1,1,1608027080),(14,'
user','user@gmail.com','00088822','asdasd','nama1518370169.jpg','$2y$10$RSu8i
0AqzDueQjQgObtwROZb.AiQNsIpoNgTYG6dCJd.p6rcaTxvy',2,1,1608027136)
;

/*Table structure for table `user_access_menu` */

DROP TABLE IF EXISTS `user_access_menu`;

CREATE TABLE `user_access_menu` (
  `id_access_menu` int(11) NOT NULL AUTO_INCREMENT,
  `level_id` int(11) DEFAULT NULL,
  `menu_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_access_menu`),
  KEY `level_id` (`level_id`),
```

```
  KEY `menu_id` (`menu_id`),
  CONSTRAINT `user_access_menu_ibfk_1` FOREIGN KEY (`level_id`)
REFERENCES `user_level` (`level_id`),
  CONSTRAINT `user_access_menu_ibfk_2` FOREIGN KEY (`menu_id`)
REFERENCES `user_menu` (`menu_id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;

/*Data for the table `user_access_menu` */

insert  into `user_access_menu`(`id_access_menu`,`level_id`,`menu_id`) values
(1,1,1),(3,2,2),(4,1,3);

/*Table structure for table `user_level` */

DROP TABLE IF EXISTS `user_level`

CREATE TABLE `user_level` (
  `level_id` int(11) NOT NULL AUTO_INCREMENT,
  `level_name` varchar(128) DEFAULT NULL,
  PRIMARY KEY (`level_id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
```

```
/*Data for the table `user_level` */

insert  into `user_level`(`level_id`,`level_name`) values
(1,'Administrator'),(2,'Member');

/*Table structure for table `user_menu` */

DROP TABLE IF EXISTS `user_menu`;

CREATE TABLE `user_menu` (
  `menu_id` int(11) NOT NULL AUTO_INCREMENT,
  `menu` varbinary(255) DEFAULT NULL,
  PRIMARY KEY (`menu_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

/*Data for the table `user_menu` */

insert  into `user_menu`(`menu_id`,`menu`) values
(1,'Admin'),(2,'User'),(3,'Menu');

/*Table structure for table `user_sub_menu` */

DROP TABLE IF EXISTS `user_sub_menu`;

CREATE TABLE `user_sub_menu` (
  `sub_menu_id` int(11) NOT NULL AUTO_INCREMENT,
```

```
  `menu_id` int(11) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `url` varchar(255) DEFAULT NULL,
  `icon` varchar(255) DEFAULT NULL,
  `is_active` int(1) DEFAULT NULL,
  PRIMARY KEY (`sub_menu_id`),
  KEY `menu_id` (`menu_id`),
  CONSTRAINT `user_sub_menu_ibfk_1` FOREIGN KEY (`menu_id`)
REFERENCES `user_menu` (`menu_id`)
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=latin1;

/*Data for the table `user_sub_menu` */

insert  into
`user_sub_menu`(`sub_menu_id`,`menu_id`,`title`,`url`,`icon`,`is_active`) values
(1,1,'Dashboard','admin','fas fa-fw fa-tachometer-alt',1),(2,2,'My Profile','user','fas
fa-fw fa-user',1),(3,2,'Team','user/teamlist','fab fa-fw fa-
teamspeak',1),(5,2,'Tournament','user/tournament','fas fa-fw fa-
trophy',1),(11,3,'Menu','menu','fas fa-fw fa-folder',1),(12,3,'Submenu
Management','menu/submenu','fas fa-fw fa-
folder',1),(19,1,'Match','admin/match','fas fa-fw fa-
```

```
trophy',1),(22,1,'Result','admin/result','fas fa-fw fa-
folder',1),(23,1,'Team','admin/teamlist','fab fa-fw fa-
teamspeak',1),(24,1,'Tournament','admin/tournament','fas fa-fw fa-
trophy',1),(25,2,'Payment','user/payment','fas fa-fw fa-file-invoice-
dollar',1),(26,1,'Payment Setting','admin/rekening','fas fa-fw fa-file-invoice-
dollar',1),(27,1,'Payment Confirmation','admin/payment_confirm','fas fa-fw fa-
file-invoice-dollar',1),(28,1,'Streaming Setting','admin/stream','fas fa-fw fa-
stream',1),(29,1,'Store Setting','admin/store','fas fa-fw fa-file-invoice-
dollar',1),(30,1,'Galeri','admin/galeri','fas fa-fw fa-image',1);

/*Table structure for table `user_verify` */

DROP TABLE IF EXISTS `user_verify`;

CREATE TABLE `user_verify` (
  `id_verify` int(11) NOT NULL AUTO_INCREMENT,
  `email` varchar(255) NOT NULL,
  `token` varchar(255) CHARACTER SET utf8 NOT NULL,
  `date_created` date NOT NULL,
  PRIMARY KEY (`id_verify`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=latin1;
```

```
/*Data for the table `user_verify` */

/*Table structure for table `galeri_view` */

DROP TABLE IF EXISTS `galeri_view`;

/*!50001 DROP VIEW IF EXISTS `galeri_view` */;

/*!50001 DROP TABLE IF EXISTS `galeri_view` */;

/*!50001 CREATE TABLE  `galeri_view`(
 `title_id` int(11) ,
 `title_name` varchar(255) ,
 `galeri_id` int(11) ,
 `image` varchar(255) ,
 `detail` text
)*/;

/*Table structure for table `matches` */

DROP TABLE IF EXISTS `matches`;

/*!50001 DROP VIEW IF EXISTS `matches` */;

/*!50001 DROP TABLE IF EXISTS `matches` */;
```

```
/*!50001 CREATE TABLE  `matches`(
 `tournament_name` varchar(255) ,
 `match_name` varchar(255) ,
 `group_name` int(11) ,
 `date` date ,
 `time` time ,
 `team_name` varchar(32) ,
 `tournament_id` int(11) ,
 `match_id` int(11) ,
 `team_id` int(11)
)*/;
/*Table structure for table `matches1` */
DROP TABLE IF EXISTS `matches1`;
/*!50001 DROP VIEW IF EXISTS `matches1` */;
/*!50001 DROP TABLE IF EXISTS `matches1` */;
```

```
/*!50001 CREATE TABLE  `matches1`(
 `match_id` int(11) ,
 `team1` varchar(11) ,
 `team2` varchar(11) ,
 `team3` varchar(11) ,
 `team4` varchar(11) ,
 `team5` varchar(11) ,
 `team6` varchar(11) ,
 `team7` varchar(11) ,
 `team8` varchar(11) ,
 `team9` varchar(11) ,
 `team10` varchar(11) ,
 `team11` varchar(11) ,
 `team12` varchar(11) ,
 `team13` varchar(11) ,
```

```
 `team14` varchar(11) ,

 `team15` varchar(11) ,

 `team16` varchar(11)

)*/;

/*Table structure for table `teamname` */

DROP TABLE IF EXISTS `teamname`;

/*!50001 DROP VIEW IF EXISTS `teamname` */;

/*!50001 DROP TABLE IF EXISTS `teamname` */;

/*!50001 CREATE TABLE  `teamname`(

 `match_id` int(11) ,

 `team1` varchar(11) ,

 `team2` varchar(11) ,

 `team3` varchar(11) ,

 `team4` varchar(11) ,

 `team5` varchar(11) ,

 `team6` varchar(11) ,
```

```
 `team7` varchar(11) ,

 `team8` varchar(11) ,

 `team9` varchar(11) ,

 `team10` varchar(11) ,

 `team11` varchar(11) ,

 `team12` varchar(11) ,

 `team13` varchar(11) ,

 `team14` varchar(11) ,

 `team15` varchar(11) ,

 `team16` varchar(11) ,

 `team_name` varchar(32) ,

 `team_id` int(11)

)*/;

/*Table structure for table `teamplayername` */

DROP TABLE IF EXISTS `teamplayername`;
```

```
/*!50001 DROP VIEW IF EXISTS `teamplayername` */;

/*!50001 DROP TABLE IF EXISTS `teamplayername` */;

/*!50001 CREATE TABLE  `teamplayername`(

 `PlayerName` varchar(32) ,

 `TeamName` varchar(32)

)*/;

/*View structure for view galeri_view */

/*!50001 DROP TABLE IF EXISTS `galeri_view` */;

/*!50001 DROP VIEW IF EXISTS `galeri_view` */;

/*!50001 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `galeri_view` AS select `galeri_title`.`title_id`
AS `title_id`,`galeri_title`.`title_name` AS `title_name`,`galeri`.`galeri_id` AS
`galeri_id`,`galeri`.`image` AS `image`,`galeri`.`detail` AS `detail` from (`galeri`
join `galeri_title` on((`galeri`.`title_id` = `galeri_title`.`title_id`))) */;

/*View structure for view matches */

/*!50001 DROP TABLE IF EXISTS `matches` */;

/*!50001 DROP VIEW IF EXISTS `matches` */;
```

```
/*!50001 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `matches` AS select
`tournament`.`tournament_name` AS
`tournament_name`,`matchup`.`match_name` AS
`match_name`,`matchup`.`group_id` AS `group_name`,`matchup`.`date` AS
`date`,`matchup`.`time` AS `time`,`team`.`team_name` AS
`team_name`,`tournament`.`tournament_id` AS
`tournament_id`,`matchup`.`match_id` AS `match_id`,`team`.`team_id` AS
`team_id` from ((`matchup` join `tournament` on((`matchup`.`tournament_id` =
`tournament`.`tournament_id`))) join `team` on((`matchup`.`team_id` =
`team`.`team_id`))) */;

/*View structure for view matches1 */

/*!50001 DROP TABLE IF EXISTS `matches1` */;

/*!50001 DROP VIEW IF EXISTS `matches1` */;

/*!50001 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `matches1` AS (select `matchup`.`match_id`
AS `match_id`,substring_index(substring_index(`matchup`.`team_id`,',',1),',',-(1))
AS `team1`,substring_index(substring_index(`matchup`.`team_id`,',',2),',',-(1)) AS
`team2`,substring_index(substring_index(`matchup`.`team_id`,',',3),',',-(1)) AS
`team3`,substring_index(substring_index(`matchup`.`team_id`,',',4),',',-(1)) AS
`team4`,substring_index(substring_index(`matchup`.`team_id`,',',5),',',-(1)) AS
```

```
`team5`,substring_index(substring_index(`matchup`.`team_id`,',',6),',',-(1)) AS
`team6`,substring_index(substring_index(`matchup`.`team_id`,',',7),',',-(1)) AS
`team7`,substring_index(substring_index(`matchup`.`team_id`,',',8),',',-(1)) AS
`team8`,substring_index(substring_index(`matchup`.`team_id`,',',9),',',-(1)) AS
`team9`,substring_index(substring_index(`matchup`.`team_id`,',',10),',',-(1)) AS
`team10`,substring_index(substring_index(`matchup`.`team_id`,',',11),',',-(1)) AS
`team11`,substring_index(substring_index(`matchup`.`team_id`,',',12),',',-(1)) AS
`team12`,substring_index(substring_index(`matchup`.`team_id`,',',13),',',-(1)) AS
`team13`,substring_index(substring_index(`matchup`.`team_id`,',',14),',',-(1)) AS
`team14`,substring_index(substring_index(`matchup`.`team_id`,',',15),',',-(1)) AS
`team15`,substring_index(substring_index(`matchup`.`team_id`,',',16),',',-(1)) AS
`team16` from `matchup`) */;

/*View structure for view teamname */

/*!50001 DROP TABLE IF EXISTS `teamname` */;

/*!50001 DROP VIEW IF EXISTS `teamname` */;

/*!50001 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `teamname` AS (select
`matches1`.`match_id` AS `match_id`,`matches1`.`team1` AS
`team1`,`matches1`.`team2` AS `team2`,`matches1`.`team3` AS
`team3`,`matches1`.`team4` AS `team4`,`matches1`.`team5` AS
`team5`,`matches1`.`team6` AS `team6`,`matches1`.`team7` AS
```

```
`team7`,`matches1`.`team8` AS `team8`,`matches1`.`team9` AS
`team9`,`matches1`.`team10` AS `team10`,`matches1`.`team11` AS
`team11`,`matches1`.`team12` AS `team12`,`matches1`.`team13` AS
`team13`,`matches1`.`team14` AS `team14`,`matches1`.`team15` AS
`team15`,`matches1`.`team16` AS `team16`,`team`.`team_name` AS
`team_name`,`team`.`team_id` AS `team_id` from (`matches1` join `team`
on(((`matches1`.`team1` = `team`.`team_id`) or (`matches1`.`team2` =
`team`.`team_id`) or (`matches1`.`team3` = `team`.`team_id`) or
(`matches1`.`team4` = `team`.`team_id`) or (`matches1`.`team5` =
`team`.`team_id`) or (`matches1`.`team6` = `team`.`team_id`) or
(`matches1`.`team7` = `team`.`team_id`) or (`matches1`.`team8` =
`team`.`team_id`) or (`matches1`.`team9` = `team`.`team_id`) or
(`matches1`.`team10` = `team`.`team_id`) or (`matches1`.`team11` =
`team`.`team_id`) or (`matches1`.`team12` = `team`.`team_id`) or
(`matches1`.`team13` = `team`.`team_id`) or (`matches1`.`team14` =
`team`.`team_id`) or (`matches1`.`team15` = `team`.`team_id`) or
(`matches1`.`team16` = `team`.`team_id`)))) where ((`team`.`team_id` =
`matches1`.`team1`) or (`matches1`.`team2` <> 0) or (`matches1`.`team3` <> 0) or
(`matches1`.`team4` <> 0) or (`matches1`.`team5` <> 0) or (`matches1`.`team6`
<> 0) or (`matches1`.`team7` <> 0) or (`matches1`.`team8` <> 0) or
(`matches1`.`team9` <> 0) or (`matches1`.`team10` <> 0) or (`matches1`.`team11`
<> 0) or (`matches1`.`team12` <> 0) or (`matches1`.`team13` <> 0) or
```

(`matches1`.`team14` <> 0) or (`matches1`.`team15` <> 0) or

(`matches1`.`team16` <> 0))) */;

/*View structure for view teamplayername */

/*!50001 DROP TABLE IF EXISTS `teamplayername` */;

/*!50001 DROP VIEW IF EXISTS `teamplayername` */;

/*!50001 CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`

SQL SECURITY DEFINER VIEW `teamplayername` AS (select (case when

(`players`.`team_id` = `team`.`team_id`) then `players`.`player_name` end) AS

`PlayerName`,(case when (`players`.`team_id` = `team`.`team_id`) then

`team`.`team_name` end) AS `TeamName` from (`team` left join `players`

on((`players`.`team_id` = `team`.`team_id`)))) */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS

*/;

/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;

/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;