Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 12, Desember 2018, hlm. 6752-6760 http://j-ptiik.ub.ac.id

Pengembangan Aplikasi Manajemen *Event* Berbasis Web (Studi Kasus: Fakultas Ilmu Administrasi Universitas Brawijaya Malang)

Hendro Febrian Bachri[1], Bayu Priyambadha[2], Denny Sagita Rusdianto[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]hendrofebrian43@gmail.com, [2]bayu_priyambadha@ub.ac.id, [3]denny.sagita@ub.ac.id

Abstrak

Event dapat didefinisikan sebagai kegiatan yang diselenggarakan guna memperingati hal-hal yang penting sepanjang hidup manusia baik secara individu maupun kelompok (Noor, 2013). Fakultas Ilmu Administrasi Universitas Brawijaya (FIA UB) sangat aktif dalam menyelenggarakan *event* seperti seminar, *workshop* dan lomba. Untuk mengadakan *event* di FIA UB tentunya pengaju *event* harus melalui prosedur tertentu. Prosedur pengajuan *event* berbeda-beda tergantung pengaju event, *resource* yang diperlukan serta lokasi penyelenggaraan *event*. Namun terdapat beberapa permasalahan yang timbul dalam proses pengajuan *event* di FIA UB. Dari perspektif pengaju *event*, masalah yang timbul antara lain pengaju *event* tidak mengetahui status pengajuan *event* yang diajukannya, tidak mengetahui status ketersediaan *venue* dan banyak pengaju *event* yang tidak mengetahui alur pengajuan *event*. Dari perspektif TU Subbag Umum & Perlengkapan, pengecekan ketersediaan *venue* dan pencatatan data peminjaman *venue* yang masih dilakukan didalam *file* kurang efisien dan memakan waktu. Sehingga dibuatlah aplikasi Manajemen *Event*. Untuk mengimplementasikan aplikasi Manajemen *Event*, penulis menggunakan Node.js dan Express.js serta SemanticUI. Pengujian dilakukan dengan menggunakan pengujian unit, validasi dan *browser compatibility*. Pada pengujian *unit*, seluruh *independent paths* telah diuji dan seluruhnya *valid*. Hasil pengujian validasi juga menunjukkan nilai 100% *valid*. Sedangkan hasil pengujian *browser compatibility* menunjukkan aplikasi dapat dijalankan tanpa kehilangan fungsionalitas di peramban Microsoft Edge, Safari, Google Chrome, Firefox dan Opera.

Kata kunci: *Manajemen Event, Node.js, Express.js, SemanticUI*

Abstract

Events can be defined as activities that are held to celebrate the important things throughout human life both individually and in groups (Noor, 2013). Faculty of Administrative Sciences, University of Brawijaya (FIA UB) is really active in organizing events such as seminars, workshops and competitions. There is an event proposal submission procedure for holding an event in FIA UB. The event proposal submission procedure is different and it depends on who is the event organizer, what resources are required and the where the event will be held. There are some problems that arise in the process of event proposal submission in FIA UB. From the event organizers perspective, they don't know the status of the submission of the event proposals that they have submitted, they also don't know the status of the venue's availability and many of event organizer do not know the flow of the event submission. From perspective of the General and Equipment Subdivision, the checking of venue's availability and recording of venue's reservation that are still performed manually in files is less efficient and time-consuming. So, the author have a solution to make an application called Event Management application. This Event Management application will be implemented using Node.js, Express.js framework and SemanticUI. The testing conducted using unit, validation and browser compatibility testing methods. In unit testing, all of independent paths tested and all of them are valid. The validation testing also shows result 100% valid. While browser compatibility testing shows that this application can be run without missing functionality in Microsoft Edge, Safari, Google Chrome, Firefox and Opera.

Keywords: *Event Management, Node.js, Express.js, SemanticUI*

1. PENDAHULUAN

Event dapat didefinisikan sebagai kegiatan yang diselenggarakan guna memperingati hal-hal yang penting sepanjang hidup manusia baik secara individu maupun kelompok dan terikat secara adat, budaya, tradisi dan agama (Noor, 2013).

Fakultas Ilmu Administrasi Universitas Brawijaya (FIA UB) sangat aktif dalam menyelenggarakan *event-event* seperti seminar, *workshop*, lomba dan lain-lain. Untuk menyelenggarakan *event* di FIA UB, tentunya harus melewati suatu prosedur. Alur pengajuan *event* berbeda-beda tergantung pengaju event, *resource* yang diperlukan serta lokasi penyelenggraan *event*.

Terdapat beberapa permasalahan yang timbul dalam proses pengajuan *event* di FIA UB. Dari perspektif pengaju *event*, masalah yang timbul antara lain pengaju *event* tidak mengetahui status pengajuan *event* yang diajukannya, tidak mengetahui status ketersediaan *venue* dan banyak pengaju *event* yang tidak mengetahui alur pengajuan *event*. Dari perspektif TU Subbag Umum & Perlengkapan, pengecekan ketersediaan *venue* dan pencatatan data peminjaman *venue* yang masih dilakukan didalam *file* kurang efisien dan memakan waktu. Selain itu pengarsipan proposal *event* juga menjadi kendala tersendiri dikarenakan FIA UB belum memiliki ruang arsip.

Sehingga untuk mempermudah proses pengajuan *event* dan untuk membantu mengatasi kekurangan yang timbul akibat prosedur manual pengajuan *event* di FIA UB, penulis memiliki solusi yaitu dengan membangun aplikasi Manajemen *Event*.

Untuk mengimplementasikan aplikasi ini penulis akan menggunakan Node.js dengan bantuan *framework* Express.js sebagai *back-end* dan SemanticUI untuk membantu mempercepat pengimplementasian *front-end* dari aplikasi ini. Penulis juga akan menggunakan aplikasi Google Calendar untuk menampilkan kalender peminjaman *venue*.

Dari latar belakang yang telah dijelaskan, dibuatlah rumusan masalah yaitu bagaimana hasil analisis dan elisitasi kebutuhan sistem dari prosedur pengajuan *event* yang berlaku di Fakultas Ilmu Administrasi Universitas Brawijaya? dan bagaimana hasil perancangan, implementasi dan pengujian aplikasi Manajemen *Event* yang dibangun menggunakan SDLC *Waterfall* dan menggunakan *framework* Express.js?

Tujuan dari penelitian ini adalah untuk membangun aplikasi Manajemen *Event* yang dapat digunakan untuk membantu proses pengajuan *event*, peminjaman *venue* dan pengajuan dana di Fakultas Ilmu Administrasi Universitas Brawijaya Malang.

Penelitian ini diharapkan dapat membawa manfaat bagi Fakultas Ilmu Administrasi Universitas Brawijaya untuk membantu proses pengajuan *event*, mempermudah pengarsipan proposal *event*, peminjaman *venue*, pengajuan dana serta mempermudah mendapatkan informasi event di Fakultas Ilmu Administrasi Universitas Brawijaya.

Batasan masalah pada penelitian ini yaitu penelitian ini hanya membahas proses analisis dan kebutuhan, perancangan, pengimplementasian dan pengujian aplikasi Manajemen *Event* di Fakultas Ilmu Administrasi Universitas Brawijaya yang dibangun menggunakan SDLC *Waterfall* dan menggunakan *framework* Express.js.

2. KAJIAN TEORI

2.1. *Event*

Event dapat didefinisikan sebagai kegiatan yang diselenggarakan guna memperingati hal-hal yang penting sepanjang hidup manusia baik secara individu maupun kelompok dan terikat secara adat, budaya, tradisi dan agama (Noor, 2013).

Menurut Any Noor (2013) *event* dapat dikategorikan berdasarkan kategori *special event*, yaitu:

1. *Leisure Event*
2. *Cultural Event*
3. *Personal Event*
4. *Organizational Event*.

2.3. Node.js, Express.js dan SemanticUI

Node.js adalah *framework* Javascript yang sangat *powerful* yang dibangun diatas mesin Javascript Google Chrome v8. Node.js menggunakan *event-driven* dan *non-blocking* I/O *model* yang membuatnya efisien dan *lightweight*. ExpressJS merupakan *framework* Node.js yang fleksibel dan minimalis yang menyediakan sekumpulan fitur yang dapat digunakan untuk membuat aplikasi berbasis Node.js (Express, 2017).

Pada penelitian ini, penulis menggunakan

beberapa *packages* untuk menyederhanakan dan mempercepat proses implementasi. *Packages* utama yang digunakan adalah:

a. Passport.js
   Passport merupakan suatu *middleware* authentikasi untuk Node.js. Passport menyediakan ratusan *strategy* mulai dari *local strategy*, Facebook, Twitter, Gmail dan lainnya.
b. Express-mysql-session
   Merupakan suatu *package* yang digunakan untuk membantu pengimplementasian *persistent session*.
c. Node-google-calendar
   Adalah suatu *package* yang digunakan untuk mempermudah pengimplementasian Google Calendar API dengan menggunakan Node.js.

SemanticUI merupakan *framework* CSS yang digunakan untuk mempercepat proses pembangunan *user interface* dari suatu aplikasi. SemanticUI menyediakan berbagai macam elemen mulai dari elemen yang sederhana seperti *button* hingga elemen kompleks seperti *accordion, modal dan sidebar* yang tentunya akan memakan waktu jika dibangun dari awal.

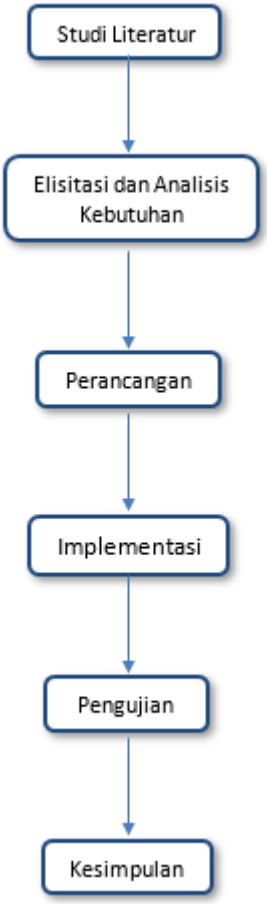
2.4. *Software Development Life Cycle Waterfall*

Menurut penelitian yang berjudul "*A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*", SDLC *Waterfall* dapat digunakan ketika kebutuhan-kebutuhan dari suatu sistem sudah diketahui diawal, *clear* dan *fixed* (Alshamrani & Bahattab, 2015).

3. METODOLOGI PENELITIAN

Gambar 1 merupakan metodologi pada penelitian ini. Pada penelitian ini, penulis akan menggunakan SDLC (*Software Development Life Cycle*) *Waterfall*. Pada penelitian ini terdapat beberapa tahapan dimulai dari studi literatur sampai penarikan kesimpulan.

Tahap pertama adalah studi literatur. Pada tahap studi literatur penulis akan membahas kajian teoritis dan empiris yang digunakan sebagai dasar teori. Tahap kedua adalah analisis dan elisitasi kebutuhan. Untuk mengelisitasi kebutuhan, penulis akan menggunakan metode *interview* secara langsung kepada Mahasiswa, Wakil Dekan 2, Wakil Dekan 3, Kepala/*Staff* TU Subbag Umum dan Perlengkapan, Kepala/*Staff* TU Subbag Keuangan dan Kepala Lab Sistem Informasi Manajemen Fakultas Ilmu Administrasi Universitas Brawijaya.



Gambar 1. Metodologi Penelitian

Kemudian dilanjutkan dengan tahap analisis kebutuhan dilanjutkan dengan validasi dan verifikasi kebutuhan serta manajemen kebutuhan. Penulis akan melakukan validasi dan verifikasi kebutuhan kepada Kepala Gugus Jaminan Mutu FIA UB. Pada tahap ini juga akan dilakukan pemodelan kebutuhan dengan menggunakan *use case diagram* dan *use case scenario* serta *state transition diagram*.

Tahap ketiga dalah perancangan. Karena didalam penelitian ini penulis menggunakan paradigma pemrograman berorientasi objek, sehingga dalam memodelkan perancangan sistem penulis akan menggunakan UML Diagram. Perancangan sistem akan dibagi menjadi 4 yaitu perancangan arsitektur yang berisi *sequence* diagram dan kelas diagram, perancangan komponen yang berisi *pseudocode*, perancangan data yang berisi *entity relationship diagram* dan perancangan *user interface.*

Setelah itu dilanjutkan dengan tahap implementasi. Pada penelitian ini, penulis akan

menggunakan Node.js dengan bantuan *framework* Express.js untuk mengimplementasikan kode program aplikasi ini. Sedangkan untuk implementasi antarmuka, penulis akan menggunakan *framework* CSS yaitu SemanticUI. Kemudian penulis juga akan melakukan implementasi data dengan membuat *physical data model*.

Pada tahap pengujian penulis akan menggunakan metode *manual testing* (tidak menggunakan *tool*) untuk menguji kebutuhan fungsional pada penelitian ini. Sedangkan untuk kebutuhan non-fungsional akan diuji secara *automate* (menggunakan *tool*). Fase pengujian pada penelitian ini akan dibagi menjadi 3 tahap yaitu pengujian unit menggunakan metode *basis path testing*, pengujian validasi menggunakan *scenario based testing* dan pengujian *browser compatibility*.

Tahap yang terakhir adalah penarikan kesimpulan yang digunakan untuk menentukan apakah perangkat lunak sudah layak atau belum untuk di antarkan dan digunakan oleh pengguna. Pada tahap ini juga terdapat saran yang menjelaskan tentang peluang pengembangan selanjutnya dari penelitian ini. Kesimpulan akan menjawab seluruh permasalahan yang didefinisikan sebelumnya pada rumusan masalah.

4. ANALISIS KEBUTUHAN SISTEM

4.1. Analisis dan Elisitasi Kebutuhan

Pada tahap elisitasi, penulis mengelisitasi kebutuhan dengan menggunakan teknik *interview* atau wawancara secara langsung. Penulis telah melakukan wawancara kepada Wakil Dekan 2 (Dr. Hamidah Nayati Utami, M.Si), Wakil Dekan 3 (Dr. Mohammad Rozikin, M.AP), Staff TU Umum dan Perlengkapan (Dyah Susanti), Kepala TU Subbag Keuangan (Jaedi, SP), Kepala Lab Sistem Informasi Manajemen (Rizky Yudhi Dewantara, S.Sos., M.AP) dan Mahasiswa Fakultas Ilmu Administrasi Universitas Brawijaya (Aria Adi Negoro dan Aan Suryana).

Penulis telah melakukan wawancara kepada Wakil Dekan 2 untuk mengetahui alur pengajuan *event* jika pengaju *event* adalah unit internal fakultas. Sedangkan untuk mengetahui alur pengajuan *event* organisasi mahasiswa, penulis melakukan wawancara kepada Wakil Dekan 3. Untuk mengetahui prosedur peminjaman fasilitas fakultas khususnya peminjaman *venue*, penulis telah melakukan wawancara terhadap *staff* Tata Usaha Subbag Umum dan Perlengkapan.

Kemudian penulis melakukan wawancara kepada Kepala Tata Usaha Subbag Keuangan Keuangan untuk mengetahui prosedur pencairan dana. Selain itu, penulis juga mewawancarai Kepala Lab SIM FIA untuk mengetahui permasalahan yang timbul dari pengajuan event secara manual dari sisi pengaju event (unit internal fakultas).

Kemudian penulis juga mewawancarai mahasiswa FIA UB untuk mengetahui masalah yang timbul dari pengajuan *event* secara manual dari perspektif pengaju *event* dalam hal ini organisasi mahasiswa. Setelah proses elisitasi kebutuhan dilakukan, dilanjutkan dengan proses analisis kebutuhan. Hasil pada tahap analsis kebutuhan akan dijelaskan secara lebih detail pada subbab berikut ini.

4.2. Gambaran Umum Aplikasi

Aplikasi Manajemen *Event* merupakan aplikasi yang dapat digunakan untuk mengajukan *event*, menampilkan kalender peminjaman *venue* serta memeriksa ketersediaan *venue*. Untuk mengajukan *event*, aktor harus mengisi data pengajuan *event* seperti meng*upload file* proposal, mengisi dana yang diajukan dan asal dana serta mengisi data peminjaman *venue*. Untuk peminjaman venue, aktor dapat memilih peminjaman *venue* untuk *event* rutin dan bukan *event* rutin.

Setelah aktor menekan tombol ajukan *event*, sistem akan menentukan *event* tersebut membutuhkan *approval* dari siapa saja. Sistem menentukan hal tersebut berdasarkan pengaju *event* (organisasi mahasiswa atau unit internal fakultas), *resource* yang dibutuhkan (venue dan dana) dan tempat penyelenggaraan *event* (didalam kota Malang atau diluar kota Malang).

4.3. Identifikasi Aktor

Pada penelitian ini terdapat 16 aktor yang terdiri dari 15 *primary actor* dan 1 *secondary actor* (Google Calendar). Tabel 1 dibawah ini berisi identifikasi aktor dari sistem yang dikembangkan pada penelitian ini.

Tabel 1. Identifikasi Aktor

Nama Aktor	Deskripsi
Non Member	Merupakan aktor yang belum masuk kedalam sistem (hanya dapat melihat halaman Login).
Organisasi Mahasiswa	Merupakan aktor yang

	dapat mengajukan *event* yang berasal dari organisasi mahasiswa.
Unit Internal Fakultas	Merupakan aktor yang dapat mengajukan *event* yang berasal dari unit internal fakultas.
TU Subbag Kemahasiswaan	Aktor ini dapat mengajukan *event* sekaligus melihat seluruh *event* yang diajukan mahasiswa.
Superadmin	Merupakan aktor yang berperan dalam mengelola data *event*, data *venue*, data akun dan data peminjaman *venue* di aplikasi ini.
Dekan	Aktor ini dapat mengajukan *event* sekaligus menyetujui pengajuan *event*
Wakil Dekan 1	Aktor ini dapat mengajukan *event* sekaligus menyetujui pengajuan *event*
Wakil Dekan 2	Aktor ini dapat mengajukan *event* sekaligus menyetujui pengajuan *event*
Wakil Dekan 3	Aktor ini dapat mengajukan *event* sekaligus menyetujui pengajuan *event*
Kepala Tata Usaha	Aktor ini dapat mengajukan *event* sekaligus menyetujui pengajuan *event*
TU Subbag Keuangan	Aktor ini dapat mengajukan *event* sekaligus memasukkan data pencairan dana
TU Subbag Umum dan Perlengkapan	Aktor ini dapat mengajukan *event* sekaligus menyetujui peminjaman *venue*
TU Subbag Akademik	Aktor ini dapat mengajukan *event* sekaligus menyetujui peminjaman *venue*
Kepala Jurusan	Aktor ini dapat mengajukan *event* sekaligus menyetujui pengajuan *event*
BEM	Aktor ini dapat mengajukan *event* sekaligus menyetujui pengajuan *event*
Google Calendar	Aktor ini merupakan sistem eksternal yang akan digunakan oleh aplikasi Manajemen *Event* untuk menampilkan *event* dalam format kalender.

4.4. Kebutuhan Fungsional dan Non Fungsional

Kebutuhan fungsional merujuk kepada fitur-fitur atau *service* apa saja yang disediakan oleh sistem. Sedangkan kebutuhan non-fungsional lebih menekankan kepada batasan *service* dari sistem. Karena keterbatasan *space*, maka penulis tidak dapat menyertakan definisi dan spesifikasi kebutuhan fungsional pada *paper* ini. Pada penelitian ini terdapat 59 kebutuhan fungsional dan 1 kebutuhan non-fungsional. Tabel 2 dibawah ini berisi kebutuhan non-fungsional dari sistem yang dikembangkan pada penelitian ini.

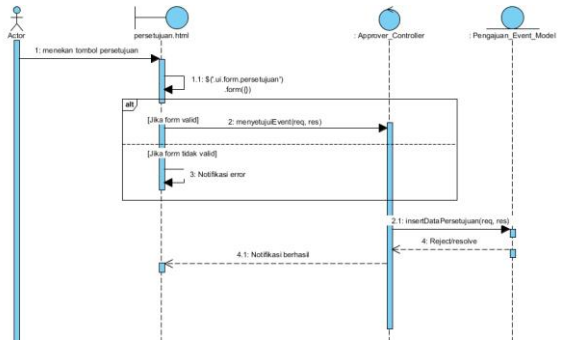
Tabel 2. Kebutuhan Non Fungsional

No	Kode Kebutuhan	Parameter	Deskripsi
1	AME_NF _100	*Browser Compatibility Testing*	Sistem harus dapat dijalankan tanpa kehilangan fungsionalitas *browser* yaitu Google Chrome, Microsoft Edge, Firefox, Opera dan Safari.

5. PERANCANGAN

5.1. Perancangan Arsitektur

Bab perancangan arsitektur membahas secara terperinci mengenai *Sequence* dan *Class* Diagram pada aplikasi Manajemen *Event*. *Sequence* diagram digunakan untuk menunjukkan komunikasi dinamis antar *object* selama proses eksekusi (Pressman, 2015).



Gambar 2. *Sequence Diagram* Menyetujui *Event*

Gambar 2 diatas merupakan *sequence* diagram dari salah satu fungsionalitas bernama menyetujui *event*. Aktor di *sequence* diagram menyetujui *event* adalah Dekan, KTU, WD1, Ketua Jurusan dan BEM. Pada *sequence* diagram menyetujui *event* terdapat 1 *boundary* yaitu pengajuan_event.html. Terdapat satu

control yaitu Approver_Controller dan satu *entity* yaitu Pengajuan_Event_Model. Pada *sequence* diagram ini, juga terdapat 1 *alternative fragment*.

Pada tahap perancangan arsitektur, penulis juga membuat *class* diagram. *Class* diagram merupakan diagram yang menggambarkan struktur kelas (*property* dan *method*) dan relasi antar kelas. Pada OOD (*Object Oriented Design*), *candidate class* didapat dari fase OOA (*Object Oriented Analysis*). Untuk mengidentifikasi *candidate class* pada fase OOA, penulis melakukan analisis terhadap *noun* (kata benda) dan *pronoun* (kata benda pengganti) yang ada pada *main flow* dan *alternative flow* yang terdapat di *use case scenario*.

Pada class diagram di penelitian ini terdapat 2 jenis relasi yaitu *association* dan *generalization*. SuperAdmin_Controller dan Pengaju_Event_ Controller merupakan *subclass* dari kelas Auth_Controller. Approver_Controller merupakan *subclass* dari Pengaju_Event_Controller. Relasi antar kelas yang lainnya adalah *association*.

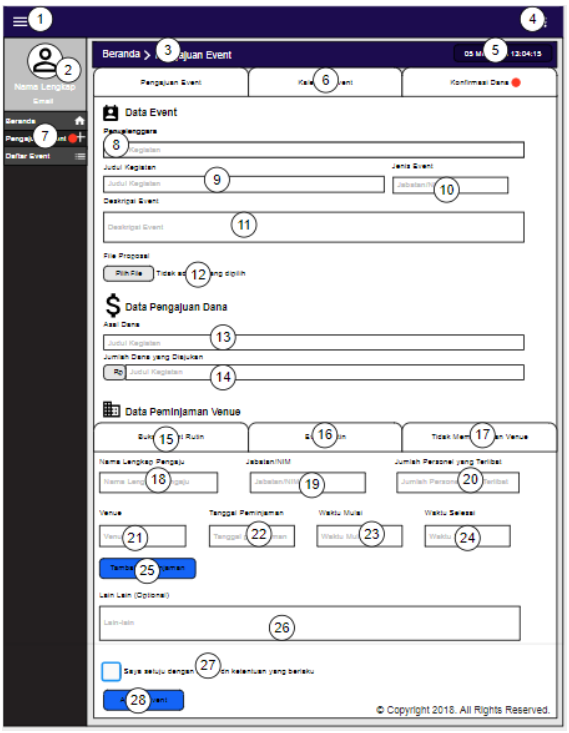
5.2. Perancangan Komponen

Pada perancangan komponen akan dijelaskan tentang algoritma yang digunakan dan akan menjadi dasar pada tahap implementasi kode program. Pada perancangan komponen pada penelitian ini, penulis membuat *pseudocode* dari 3 *method* yang terdapat di aplikasi ini yaitu *method* yang digunakan untuk melihat halaman persetujuan, mengkonfirmasi dana dan memasukkan *venue* baru.

5.3. Perancangan Antarmuka

Perancangan antarmuka dilakukan dengan membuat *mock-up* dari halaman yang akan diimplementasikan pada tahap implementasi antarmuka. Gambar 3 merupakan perancangan UI dari halaman mengajukan *event*. Pada perancangan UI mengajukan *event* terdapat nomor yang digunakan untuk memudahkan identifikasi komponen pada perancangan antarmuka.

Komponen no. 1 digunakan untuk menampilkan dan menyembunyikan *sidebar*, komponen no.2 berisi inisial (gabungan huruf pertama dari *First Name* dan *Last Name* Organisasi/Unit) kemudian nama unit/organisasi dan email, komponen no. 3 menampilkan lokasi halaman yang sedang diakses aktor (*breadcrumb*). Komponen no.4 merupakan komponen yang jika ditekan akan memunculkan *button* yang mengarahkan ke halaman edit profil dan *button* untuk *logout*.



Gambar 3. Perancangan UI Mengajukan *Event*

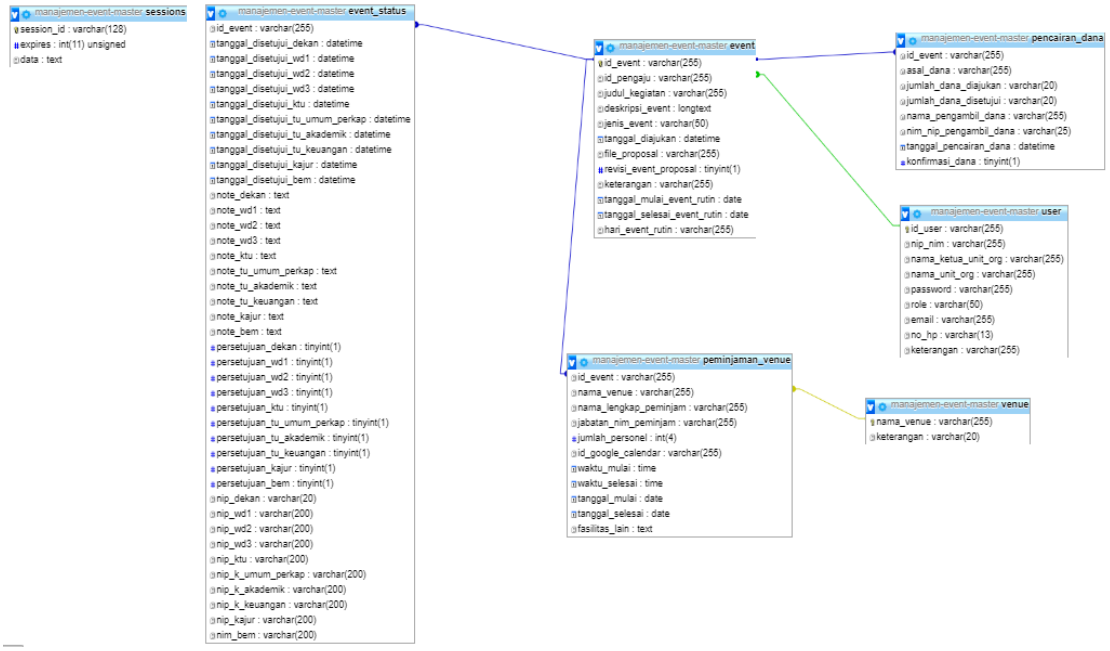
Komponen no.5 digunakan untuk menampilkan waktu, komponen no.6 digunakan untuk berpindah *tab*, komponen no.7 digunakan untuk menampilkan list menu, komponen no.8 sampai no.28 merupakan *field* yang terdapat pada form pengajuan *event* yang digunakan untuk memasukkan data pengajuan *event*.

6. IMPLEMENTASI

6.1. Implementasi Kode Program

Implementasi kode program adalah penterjemahan *pseudocode* yang sebelumnya telah dibuat pada proses perancangan kedalam bahasa pemrograman yang spesifik. Pada penelitian ini penulis menggunakan Node.js dengan bantuan *framework* Express.js untuk mengimplementasikan *back-end* dari aplikasi ini. Hasil dari pengkodean *back-end* akan diletakkan di folder, *models* dan *controllers*. Pada penelitian ini terdapat 4 *controllers* yaitu, Auth_Controller, Pengaju_Event_Controller, Approver_ Controller dan SuperAdmin_ Controller dan 3 *models* yaitu, Pengajuan_ Event_Model,Venue_Model, User_Model.

6.2. Implementasi Data

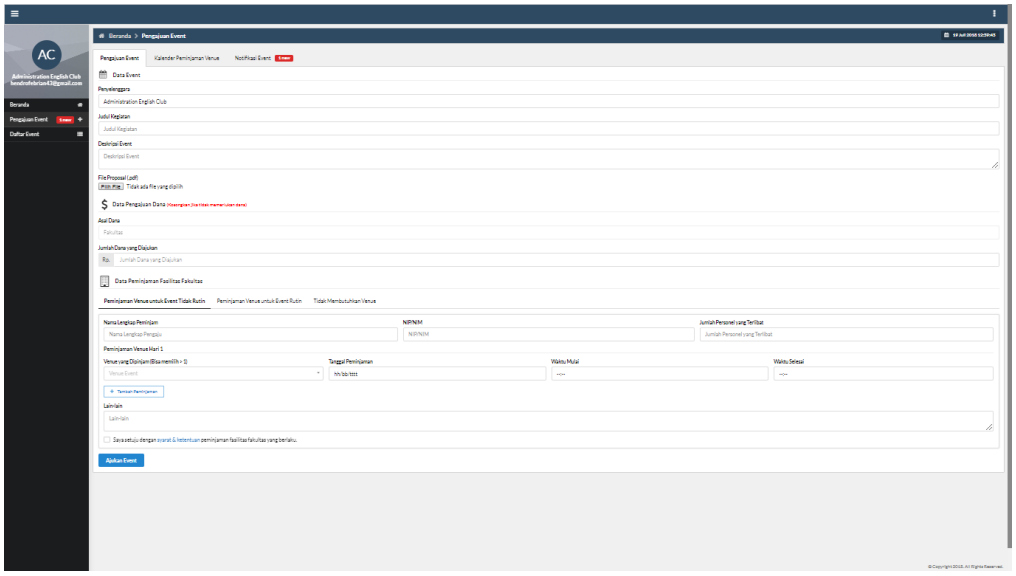


Gambar 4. Physical Data Model

Gambar 4 diatas merupakan *Physical Data Model* (PDM) menggambarkan struktur skema database sesungguhnya yang digunakan saat implementasi. Pada aplikasi ini terdapat 7 tabel.

Tabel user digunakan untuk menyimpan data akun. Tabel event, tabel event_status, tabel pencairan_dana, tabel peminjaman_venue digunakan untuk menyimpan data pengajuan *event*. Tabel venue digunakan untuk menyimpan seluruh *list venue* dan tabel *sessions* yang digunakan untuk mengimplementasikan *persistent session* di Node.js.

6.3. Implementasi Antarmuka



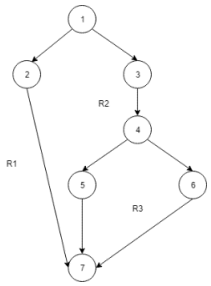
Gambar 5. Implementasi Antarmuka Halaman Pengajuan *Event*

Implementasi atau pengkodean antarmuka sistem dilakukan menggunakan HTML dengan bantuan *framework* CSS yaitu SemanticUI. Selain itu penulis juga menggunakan *library* Javacript yaitu jQuery. Gambar 5 merupakan implementasi dari antarmuka halaman pengajuan *event.*

7. PENGUJIAN

7.1. Pengujian Unit

Pengujian unit adalah proses untuk menguji komponen program seperti *method* atau objek dari suatu kelas (Sommerville, 2011). Pada pengujian *unit* di penelitian ini, penulis menggunakan metode *basis path testing*. Gambar 6 merupakan *flow graph* yang digunakan untuk menentukan *cyclomatic complexity* dan *independent path*.



Gambar 6. *Flow Graph*

a. Cyclomatic Complexity (V(G))

1. V(G) = jumlah *region* = 3
2. V(G) = jumlah *edge* - jumlah *node* + 2 = 8 – 7 + 2 = 3
3. V(G) = jumlah *predicate node* + 1 = 2 + 1 = 3

b. Independent Path

1. 1-2-7
2. 1-3-4-5-7
3. 1-3-4-6-7

7.2. Pengujian Validasi

Pada pengujian validasi atau pengujian sistem, penulis akan menggunakan metode *functional testing.* Penulis akan menguji setiap definisi kebutuhan yang ada serta skenario alternatif dari masing-masing definisi kebutuhan yang terdapat di *use case scenario*. Pada pengujian validasi diperoleh hasil 100% *valid*.

7.3. Pengujian *Browser Compatibility*

Gambar 7. Hasil Pengujian *Browser Compatibility*

Pada pengujian *browser compatibility*, penulis menggunakan metode *automate testing*. Untuk melakukan *automate testing* ini, penulis menggunakan *tool* bernama SortSite. SortSite bekerja dengan cara memeriksa beberapa hal berikut ini:

a. *Tag* HTML yang tidak didukung oleh beberapa *browser*
b. Fitur CSS yang tidak didukung oleh beberapa *browser*
c. *Vendor specific* HTML dan Javascript
d. Format gambar yang tidak didukung beberapa *browser*
e. Teknologi yang tidak didukung oleh beberapa *browser*

Gambar 7 menunjukkan hasil pengujian *browser compatibility* dengan menggunakan SortSite yang menunjukkan aplikasi dapat berjalan tanpa kehilangan konten atau fungsionalitas, tidak ada masalah *major* pada *layout* dan tidak ada masalah *minor* pada *layout* di *browser* Edge, Firefox, Safari Opera dan Chrome. Sehingga dapat disimpulkan aplikasi dapat berjalan secara sempurna di browser Edge, Firefox, Safari, Opera dan Chrome.

8. KESIMPULAN DAN SARAN

8.1. Kesimpulan

Kesimpulan diambil berdasarkan hasil dari tahap analisis kebutuhan, perancangan, implementasi dan pengujian yang sebelumnya telah dilakukan di penelitian ini. Sehingga pada penelitian ini dapat ditarik kesimpulan sebagai berikut:

1. Berdasarkan hasil analisis kebutuhan, aplikasi Manajemen *Event* mempunyai 59 kebutuhan fungsional dan 1 kebutuhan non-fungsional serta memiliki 16 aktor yang terdiri dari 15 aktor primer dan 1 aktor sekunder. Selain itu pada tahap analisis kebutuhan telah dibuat diagram *Business Process Modelling* (BPM) yang mengilustrasikan prosedur pengajuan *event* di FIA UB. Pada pemodelan kebutuhan terdapat *use case diagram* yang berisi 59 *use case* kemudian setiap

use case didetailkan di *use case scenario*. Kemudian penulis juga membuat *State Transition Diagram* (STD) yang menggambarkan alur dan status disposisi proposal atau persetujuan *event*.

2. Pada tahap perancangan aplikasi Manajemen *Event* ini, telah dibuat *sequence diagram* dari 3 kebutuhan yang ada di aplikasi Manajemen *Event* sebagai sample. Pada *class diagram* diperoleh 8 kelas yang terdiri dari 4 kelas *controller* dan 3 kelas *model* serta 1 kelas dari aplikasi eksternal yaitu Google Calendar. Pada *entity relationship diagram* diperoleh 5 entitas dimana kelima entitas tersebut dihubungkan dengan 4 relasi. Pada perancangan komponen telah dibuat *pseudocode* dari 3 *method* yang ada di aplikasi ini serta pada perancangan *user interface* penulis membuat *mock-up* dari 4 halaman yang ada di aplikasi ini.
3. Pada tahap impelementasi penulis menggunakan Node.js dengan menggunakan *framework* Express.js. Selain itu untuk mengimplementasikan Google Calendar API, penulis menggunakan *package* node-google-calendar. Untuk mengimplementasikan *user interface,* penulis menggunakan *framework* SemanticUI. Untuk implementasi data, penulis membuat *physical data model.* Pada *physical data model* terdapat 7 entitas.
4. Pada pengujian *unit,* seluruh *independent paths* telah diuji dan seluruhnya *valid*. Hasil pengujian validasi juga menunjukkan nilai 100% *valid***.** Sedangkan pada pengujian *browser compatibility* diperoleh hasil yang menunjukkan bahwa aplikasi dapat dijalankan secara sempurna di peramban Opera, Google Chrome, Firefox, Microsoft Edge dan Safari.

8.2. Saran

Saran yang dapat penulis berikan dari penelitian yang telah dilakukan dan dapat digunakan untuk pengembangan sistem dan penelitian selanjutnya adalah:

1. Melakukan penambahan fitur untuk menyimpan jadwal kuliah rutin dan jadwal kuliah pengganti. Sehingga sistem selain dapat mengecek ketersediaan *venue* yang bukan merupakan ruang kelas, juga dapat mengecek ketersediaan *venue* yang merupakan ruang kelas.
2. Perlunya pengimplementasian *transaction* pada beberapa proses yang berhubungan dengan manipulasi *database* yang bertujuan untuk menjaga *consistency* data.
3. Saran agar mengganti pengimplementasian *promise* dengan *async await* agar kode lebih *maintainable.*
4. Mengganti penggunaan aplikasi eksternal yaitu Google Calendar dengan mengimplementasikan kalender sendiri. Hal ini bertujuan untuk menurunkan *coupling* dari aplikasi Manajemen *Event*.

9. DAFTAR PUSTAKA

ALSHAMRANI, A., BAHATTAB, A., 2015. A Comparison Between Three SDLC Models Waterfall, Spiral Model, and Incremental /Iterative Model, [e-journal] pp 106-111. Tersedia melalui: IJCSI <https://www.ijcsi.org/papers/IJCSI-12-1-1-106-111.pdf> [Diakses 22 September 2017].

EXPRESS, 2017. Express.js, [online] Tersedia di:<https://expressjs.com/> [Diakses 16 September 2017].

NOOR, A., 2013. Manajemen Event. Bandung: Alfabeta.

PRESSMAN, R. S., 2015. Software Engineering A Practitioner's Approach 8th Edition, [e-book]. New York: McGraw-Hill Education. Tersedia melalui: <https://downloadnema.com/wpcontent/uploads/2017/02/Software%20Engineering%20A%20Practitioner%E2%80%99s%20Approach%20eighth%20edition(www.downloadnema .com).pdf> [Diakses 22 September 2017].

SOMMERVILE, I. 2011. Software Engineering 9th ed, [e-book]. London: AddisonWesley. Tersedia di: < https://www.pdfdrive.net/ software engineering-9th-edition-e12180831.html> [Diakses 22 September 2017]