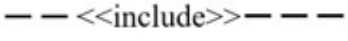
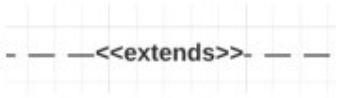
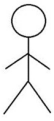
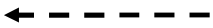
DAFTAR SIMBOL

1. Daftar simbol beserta keterangan pada *use case* Diagram

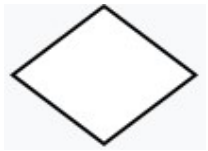
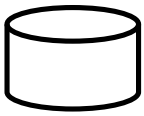
Simbol	Nama Simbol	Fungsi
	Aktor	Entitas yang melakukan interaksi
	*Use Case*	Proses interaksi antar elemen
	Relasi *include*	Interaksi yang harus dipenuhi untuk sebuah *event* dapat terjadi
	Relasi *extend*	Interaksi tambahan dari sebuah *event*, bisa dilakukan dan bisa juga tidak

2. Daftar Simbol Beserta Keterangan pada *sequence* Diagram

Simbol	Nama Simbol	Fungsi
	Aktor	Entitas yang melakukan interaksi
	Objek	Mewakili sebuah objek atau *class*
	*Lifeline*	Menggambarkan aktivitas dari sebuah objek
	*Activation Box*	Menggambarkan waktu yang diperlukan objek untuk melakukan suatu aktivitas

Simbol	Nama Simbol	Fungsi
	Synchronous Message	Komunikasi antara objek dan membutuhkan balasan pesan
	*Asynchronous Message*	Komunikasi antara objek dan tidak membutuhkan balasan pesan

3. **Daftar Simbol Beserta Keterangan pada *flowchart* Diagram**

Simbol	Nama Simbol	Fungsi
	*Terminator*	Kondisi awal atau kondisi akhir
	Proses	Proses berjalannya suatu program
	Masukan atau Keluaran	Data masukan atau hasil dari suatu proses
	Titik Keputusan	Proses / langkah di mana perlu adanya keputusan atau adanya kondisi tertentu
	*Database*	Menunjukkan hasil penyimpanan data

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Event adalah sebuah program atau proyek yang akan dilakukan secara terencana untuk suatu tujuan [1]. Acara dapat diselenggarakan dengan adanya sebuah kepastian jadwal yang dapat mengatur acara. *Scheduling* atau penjadwalan adalah salah satu masalah utama bagi sebuah kepanitiaan atau *event management* [2]. Penjadwalan yang baik merupakan salah satu hal yang dapat menghasilkan kelancaran pada sebuah acara.

Berdasarkan hasil survei yang telah dilakukan kepada mahasiswa Universitas Telkom, terdapat 86,2% responden yang membutuhkan sebuah fitur penjadwalan anggota. Responden juga memiliki beberapa kendala yang dihadapi dalam pencatatan atau pengelolaan penjadwalan, yaitu:

1. Terjadi ke tidak cocokkan antara jadwal kuliah mahasiswa dengan jadwal acara.
2. Penyampaian informasi penjadwalan kepada anggota kepanitiaan yang tidak efektif dan tidak lengkap.

Dengan adanya masalah penjadwalan di atas, dibutuhkan suatu teknologi informasi yang dapat membantu kepanitiaan dalam mengatur penjadwalannya. Cara mengatasi permasalahan di atas adalah dengan membuat sebuah aplikasi sistem *Event Management,* khususnya di bagian penjadwalan anggota panitia dengan menggunakan Algoritma *Particle Swarm Optimization*. Algoritma tersebut digunakan untuk menghasilkan jadwal dengan menggunakan nilai *fitness* yang paling optimal [3]. Penelitian ini menggunakan Algoritma *Particle Swarm Optimization* untuk penjadwalan anggota panitia, dikarenakan menurut penelitian sebelumnya PSO dapat menyelesaikan masalah tentang penjadwalan [4].

1.2. Rumusan Masalah

Berdasarkan latar belakang tersebut, maka penulis merumuskan masalah pada penelitian sebagai berikut:

1. Bagaimana cara membantu panitia dalam sebuah *event* atau kepanitiaan untuk menyusun penjadwalan anggota panitia sesuai dengan jadwal yang dimiliki anggota panitia sehingga tidak terjadi jadwal yang bentrok antara jadwal anggota dengan jadwal *event*?
2. Bagaimana kinerja Algoritma *Particle Swarm Optomization* dalam membatu pembuatan penjadwalan anggota panitia sesuai dengan peraturan yang telah ditentukan?

1.3. Tujuan dan Manfaat

Pengerjaan tugas akhir ini memiliki tujuan, yaitu :

1. Membuat aplikasi *Event Management* berbasis *website* yang memiliki fitur penjadwalan anggota panitia.
2. Melakukan pengujian untuk mengukur kinerja aplikasi pembuatan penjadwalan anggota panitia menggunakan Algoritma PSO dan dapat diakses secara sistematis dan *real time.*

1.4. Batasan Masalah

Adapun Batasan masalah pada aplikasi ini sebagai berikut:

1. Aplikasi ini dibuat berbasis *website*, dibuat menggunakan *framework Codeigniter 3, Bootstrap,* dan *Flask*.
2. Aplikasi ini tidak membahas tentang keamanan *system*.
3. Cakupan penelitian ini adalah manajemen penjadwalan acara yang ada di komunitas Telkom *University*.
4. Data jadwal kuliah yang digunakan sebagai penelitian menggunakan data jadwal kuliah mahasiswa yaitu 70 di Kepanitiaan PERMIB PERFECT
5. Aplikasi ini hanya menggunakan 7 divisi yang berbeda yaitu acara, humas, keamanan, konsumsi, logistik, publikasi, dan *sponsorship*.

6. Hasil dari pembuatan penjadwalan anggota panitia berupa daftar acara yang berisi 7 anggota berbeda-beda divisinya.

1.5. Sistematika Penulisan Tugas Akhir

Sistematika penulisan yang digunakan dalam Tugas Akhir ini adalah:

1. BAB I PENDAHULUAN

   BAB I berisi tentang latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah, sistematika penelitian.
2. BAB II LANDASAN TEORI

   BAB II berisi mengenai dasar-dasar teori yang akan digunakan pada penelitian ini untuk memecahkan masalah yang diambil dari berbagai sumber.
3. BAB III ANALISIS DAN PERANCANGAN

   BAB III berisi mengenai penjelasan gambaran umum sistem yang dibuat, data yang dibutuhkan, dan proses dari algoritma *Particle Swarm Optomization*.
4. BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

   BAB IV berisi tentang Implementasi sistem dan analisis hasil pengujian.
5. BAB V KESIMPULAN DAN SARAN

   BAB V berisi tentang kesimpulan dari hasil penelitian yang dilakukan serta rekomendasi ataupun saran untuk penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1. Penjadwalan

Penjadwalan adalah proses pengambilan keputusan yang digunakan secara teratur di banyak industri manufaktur dan jasa. Ini berkaitan dengan alokasi sumber daya untuk tugas-tugas selama periode waktu tertentu dan tujuannya adalah untuk mengoptimalkan satu atau lebih tujuan. Penjadwalan, sebagai proses pengambilan keputusan, memainkan peran penting dalam sebagian besar sistem produksi di sebagian besar lingkungan pemrosesan informasi [5]. Ini juga penting bagi sebuah kepanitiaan atau *event management* [2].

Penyusunan jadwal anggota dalam sebuah kegiatan kepanitiaan atau *event management*, tentu tidak dapat dilakukan sembarangan, karena keberhasilan acara yang efektif ditentukan oleh proses penjadwalan anggota yang optimal. Dalam penyusunan jadwal anggota, hal yang harus dilakukan adalah pendataan anggota panitia, sanggup mengikuti jadwal acara atau tidak. Ini dilakukan agar nantinya tidak terjadi bentrok antar kegiatan kepanitiaan dengan kesibukan panitia itu sendiri. Sering kali proses ini memakan waktu yang tidak sebentar dikarenakan prosesnya yang manual. Penyampaian informasi jadwal acara secara lengkap dan optimal, juga penting agar tidak terjadinya *miss* komunikasi antar kepanitiaan.

2.2. *Particle Swarm Optimization* (PSO)

Optimasi merupakan aktivitas untuk mendapatkan hasil yang terbaik atau optimal (nilai efektif yang dapat dicapai) dari pilihan yang tersedia. Tujuan dari setiap keputusan adalah untuk meminimalkan usaha yang dilakukan atau memaksimumkan keuntungan yang diperoleh[6]. Tujuan dari Optimasi Penjadwalan anggota panitia adalah untuk merencanakan anggota acara ke dalam penjadwalan acara. Dengan merencanakan penjadwalan anggota panitia dengan baik acara maka acara akan berjalan dengan lancar. Dengan menggunakan Algoritma *Particle Swarm Optimization* panitia acara dapat terbantu dalam pembuatan penjadwalan anggota panitia.

Algoritma *Particle Swarm Optimization* (PSO) diperkenalkan oleh Dr. R. C. Eberhart dan Dr. J. Kennedy pada tahun 1995, merupakan algoritma optimasi yang terinspirasi oleh perilaku sosial hewan, seperti kawanan burung atau kawanan ikan [7]. Sejak diperkenalkan pertama kali, algoritma PSO berkembang cukup pesat, baik dari sisi aplikasi maupun dari sisi pengembangan metode yang digunakan pada algoritma tersebut (R.L., Haupt, S.E Haupt, 2004) [8]. Algoritma PSO menstimulasikan perilaku sosial sekawakan burung. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel menunjukkan, misalnya, seekor burung dalam kawanan burung. Setiap individu atau partikel berperilaku secara terdistribusi dengan cara menggunakan kecerdasannya sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh di kelompok tersebut[9].

Metode optimasi yang didasarkan pada swarm ini disebut algoritma *behaviorally inspired* sebagai alternatif dari algoritma genetika, yang sering disebut *evolution-based procedures*. Dalam konteks optimasi *multivariabel*, kawanan diasumsikan mempunyai ukuran tertentu atau tetap dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/*space* tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi bagusnya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi yang bagus tersebut[10].

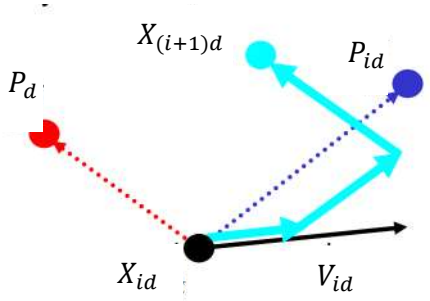
PSO dimulai dengan sekumpulan partikel (solusi) yang dibangkitkan secara acak. Setiap partikel kemudian dievaluasi kualitasnya menggunakan fungsi *fitness*. Selanjutnya, partikel-partikel akan terbang mengikuti partikel yang optimum. Pada saat iterasi, setiap partikel di perbarui mengikuti dua nilai terbaik. Yang pertama adalah *fitness* terbaik yang dicapai oleh satu partikel. Nilai *fitness* terbaik ini dilambangkan dengan *Pbest* dan disimpan di memori. Sedangkan nilai terbaik yang

kedua adalah *fitness* terbaik yang dicapai oleh semua partikel dalam tipologi ke tetangga. Indeks *Gbest* digunakan untuk menunjukkan partikel dengan *fitness* terbaik. Berikut ini merupakan model matematika yang menggambarkan mekanisme memperbarui kecepatan partikel dan posisi partikel [10].

$$V_{id} = \omega V_{id} + C_1 r_1 (P_{id} - X_{id}) + C_1 r_2 (P_d - X_{id}) \quad (2,1)$$

$$X_{id} = X_{id} + V_{id} \quad (2,2)$$

Berikut gambar 2.1 adalah ilustrasi dari algoritma PSO:



Gambar 2.1 Ilustrasi dari Algoritma PSO [11]

Berikut penjelasannya:

i = indek partikel ke-i

d = indek dimensi ke-d

ω = bobot inersia

C_1 = Parameter kognitif

C_2 = Parameter sosial

r_1 dan r_2 = angka acak dengan *range* [0,1]

P_{id} = *Pbest* partikel ke-i dimensi ke-d

P_d = *Gbest* dimensi ke-d

V_{id} = Kecepeatan partikel ke-i dimensi ke-d

X_{id} = Posisi partikel ke-i dimensi ke-d

Tidak seperti *Genetic Algorithm* (GA), PSO tidak memiliki operator evolusi, seperti *crossover* dan mutasi. Baris dalam matriks disebut partikel (sama dengan kromosom GA). Setiap partikel bergerak sekitar di permukaan partikel dengan kecepatan [12].

Berikut istilah yang digunakan dalam penerpaan algoritma PSO [10]:

1. *Swarm* = populasi dari sekawasan partikel.
2. *Particle* = individu pada suatu swarm. Setiap partikel mempresentasikan suatu solusi dari permasalahan yang diselesaikan.
3. *Pbest* = suatu partikel yang menunjukkan posisi terbaik (*Partikel Best*.
4. *Gbest* = posisi terbaik dari seluruh partikel yang ada dalam suatu *swarm (Global Best)*.
5. *Velocity* = kecepatan yang dimiliki oleh setiap partikel dalam menentukan arah perpindahan suatu partikel untuk memperbaiki posisi semula.
6. c_1 = merupakan konstanta pembelajaran kognitif.
7. c_2 = konstanta pembelajaran sosial.

Pada penelitian PSO ini digunakan untuk mencari solusi optimal untuk penjadwalan anggota. Kecerdasan Buatan yang harus memenuhi batasan-batasan yang telah ditentukan.

2.2.1. Parameter Algoritma PSO

Untuk parameter yang digunakan dalam proses algoritma PSO adalah sebagai berikut [13]:

1. Jumlah partikel (*Number of particles*) merupakan faktor yang dianggap sangat penting dalam melakukan penyelesaian masalah.
2. Bobot Inersia (*Inertia Weight*), memainkan peran yang sangat penting dalam kecepatan partikel dari algoritma PSO.
3. Faktor pembelajaran (*Learning factors*). Parameter c1 merupakan pengakuan koefisien, sedangkan c2 adalah komponen sosial.
4. Rentang dan dimensi partikel (*Range and dimension of particles*). Dimensi partikel dan rentang ditentukan berdasarkan masalah yang dioptimalkan.

5. Kecepatan (*Velocity*) merupakan perubahan maksimum pada setiap partikel, dapat diambil selama iterasi yang didefinisikan sebagai kecepatan maksimum.
6. Menghentikan kondisi (*stopping condition*) merupakan salah satu cara apabila kriteria yang dicari sudah tercapai.

2.2.2. Proses Algoritma PSO

Berikut Proses dari Algoritma PSO [10]:

1. Bangkitkan posisi awal sejumlah partikel sekaligus kecepatan awalnya secara acak.
2. Evaluasi nilai *fitness* dari masing-masing partikel berdasarkan posisinya.
3. Tentukan partikel dengan *fitness* terbaik, dan tetapkan sebagai *Gbest*. Untuk setiap partikel, *Pbest* awal akan sama dengan posisi awal.

Ulangi langkah berikut sampai *stopping criteria* dipenuhi

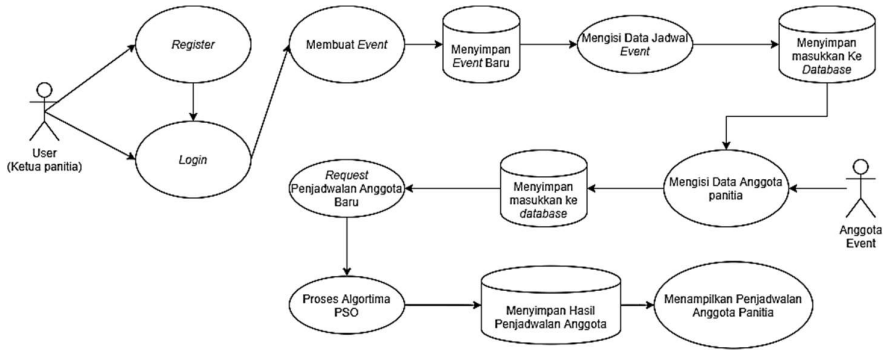
A. Menggunakan *Pbest* dan *Gbest* yang ada, perbarui kecepatan setiap partikel menggunakan persamaan (2,1). Lalu dengan kecepatan baru yang didapat, perbarui posisi setiap partikel menggunakan persamaan (2,2).
B. Evaluasi *fitnnes* dari setiap partikel.
C. Tentukan partikel dengan *finis* terbaik, dan tetapkan sebagai *Best*. Untuk setiap partikel, tentukan *Pbest* dengan membandingkan posisi sekarang dengan *Pbest* dari iterasi sebelumnya.
D. Cek *stopping criteria*. Jika dipenuhi berhenti. Jika tidak kembali ke A.

BAB III
ANALISIS DAN PERANCANGAN

3.1. Gambaran Umum Sistem

Gambaran umum sistem pada penelitian ini merupakan gambaran mengenai sistem pada Aplikasi *Event Management* dalam penjadwalan anggota menggunakan algoritma *Particle Swarm Optimization* (PSO). Adapun *input* yang diberikan oleh *user* (ketua panitia) adalah data jadwal *event* dan data anggotanya. Sedangkan untuk o*utput* yang akan dihasilkan adalah penjadwalan anggota.

Berikut adalah urutan gambaran umum aplikasi penjadwalan acara :



Gambar 3.1 Gambaran Umum Sistem

a. Seluruh *user* dapat memulai aplikasi dengan mendaftar, setelah itu *login* ke aplikasi.
b. *User* membuat *event* baru, dengan mengisi formulir *event* yang berisi nama, penyelenggara, tempat, *budget*, dan kode tambah anggota *event*.
c. Kemudian *user* dapat mengisi data jadwal *event* yang berisi nama jadwal *event*, awal dan akhir waktu acaranya.
d. Selanjutnya *user* mengisi data anggota panitia yang berisi nama anggota, nomor telepon anggota, divisi anggota dan kesanggupan mengikuti suatu jadwal *event* (Anggota *Event* juga dapat mengisi datanya sendiri dengan kode tambah anggota *event*).
e. Setelah *user* mengisi data jadwal *event* dan data anggota *event*, *user* dapat *generate* penjadwalan anggota panitia baru, setelah itu sistem membaca data jadwal anggotanya dan menghasilkan penjadwalan anggota panitia ke *database* dengan menggunakan algoritma PSO.

f. Aplikasi akan menghasilkan penjadwalan anggota panitia dari hasil algoritma PSO.

3.2. Kebutuhan Data

Untuk menghasilkan penjadwalan anggota dari algoritma PSO, Aplikasi *Event Management* ini membutuhkan data-data dari user, yaitu data jadwal acara dan data anggota acara. Sedangkan data divisi acara sudah ditentukan dari sistem. Berikut spesifikasinya adalah sebagai berikut :

1. Data Jadwal *Event*

   Jumlah dari data jadwal *event* ini akan menjadi jumlah partikel di PSO. Dalam penelitian ini menggunakan 20 jadwal acara. Berikut data jadwal acara yang akan di olah di penelitian ini :

   a. Id = Untuk id jadwal acara.
   b. *Schedule* = untuk nama dari jadwal acara.
   c. Mulai = untuk data waktu awal acara.
   d. Selesai = untuk data waktu akhir acara.

Tabel 3.1 Data Jadwal *Event*

Id	*Schedule*	Mulai	Selesai
1	Senin *Shift* 1	06:30	09:30
2	Senin *Shift* 2	09:30	12:30
3	Senin *Shift* 3	12:30	15:30

Untuk data lengkapnya terdapat pada lampiran A.

2. Data Anggota Panitia

   Terdapat syarat yang harus dipenuhi untuk dapat membuat penjadwalan acara dengan algoritma PSO. Yang pertama data anggota panitia diperlukan minimal 1 anggota per-divisi(dikarenakan di sistem ini terdapat 7 divisi, berarti minimal per-acara harus ada 7 anggota di mana setiap anggotanya berbeda divisinya). Yang terakhir jumlah Anggota per-divisinya juga tidak boleh melebihi dari jumlah data jadwal acara (misal data jadwal acaranya terdapat 15, berarti di semua divisi data anggotanya tidak

boleh melebihi 15). Jika syarat-syarat sudah dipenuhi *user* dapat membuat penjadwalan acara dengan algoritma PSO.

Adapun Data Anggota panitia yang akan diolah di penelitian ini yaitu menggunakan jadwal kuliah mahasiswa diambil dari kepanitiaan Permib PERFECT 2019 untuk acara *Try Out* SMB Telkom. Untuk isi datanya yaitu

a. ID = untuk Id data anggota
b. Id_divisi = untuk menunjukkan anggota divisi
c. nama = untuk data nama anggota panitia
d. jadwal kuliah = isi dari jadwal kuliah mahasiswa

Tabel 3.2 Data Jadwal Kuliah Mahasiswa

ID	Nama Lengkap	Id_divisi	Jadwal Kuliah	
			Senin	Selasa
1	Tia Mulyati	1	06.30 - 07.30, 07.30 - 08.30	07.30 - 08.30, 08.30 - 09.30, 13.30 - 14.30, 14.30 - 15.30, 15.30 - 16.30
2	Karina putri setiawan	1	10.30 - 11.30, 11.30 - 12.30, 12.30 - 13.30, 13.30 - 14.30, 14.30 - 15.30, 15.30 - 16.30	09.30 - 10.30, 10.30 - 11.30, 11.30 - 12.30
3	Alifia Salsabila	1	06.30 - 07.30, 07.30 - 08.30, 08.30 - 09.30, 09.30 - 10.30, 12.30 - 13.30, 13.30 - 14.30	10.30 - 11.30, 11.30 - 12.30, 12.30 - 13.30, 13.30 - 14.30

Setelah itu di olah menjadi tabel berikut:

Tabel 3.3 Data Jadwal Mahasiswa Setelah Di Olah

ID	Nama Lengkap	Senin				Selasa
		Shift 1 06:30 - 09:30	*Shift* 2 09:30 - 12:30	*Shift* 3 12:30 - 15:30	*Shift* 4 15:30 - 18:30	*Shift* 1 06:30 - 09:30
1	Tia Mulyati	0	1	1	1	0
2	Karina Putri Setiawan	1	0	0	0	1
3	Alifia Salsabila	0	0	0	1	1

Terdapat nama hari dan *shift*, yang menunjukkan jadwal acaranya mulai dan selesai pukul tersebut. Selanjutnya terdapat angka 0 dan 1 yang menunjukkan jika angka 0 berarti anggota tersebut tidak dapat mengikut acara pada saat itu dikarenakan bertabrakan dengan jadwal kuliahnya, jika angka 1 berarti anggota tersebut dapat mengikuti acara pada saat itu. Untuk data lengkap jadwal kuliah yang belum di olah terdapat di lampiran B, sedangkan data lengkap jadwal kuliah yang sudah di olah terdapat di lampiran C.

3. Data Divisi

Terdapat 7 Divisi yang membutuhkan anggota panitia ke dalam penjadwalan anggota panitia. Untuk jumlah data anggota yang diolah di penelitian ini, dan nama dari divisi tersebut dapat dilihat pada tabel di bawah ini :

Tabel 3.4 Data Divisi

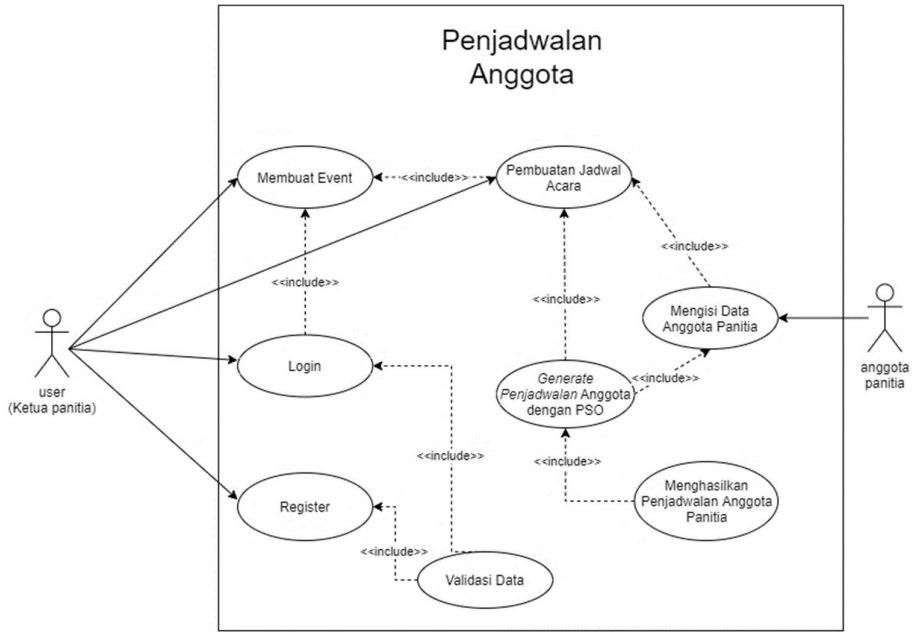
No.	Divisi	Jumlah Anggota di penelitian ini
1.	Acara	10
2.	Humas	10
3.	Keamanan	10
4.	Konsumsi	10
5.	Logistik	10
6.	Publikasi	10
7.	*Sponsorship*	10

3.3. Perancangan Sistem

Sistem penjadwalan anggota menggunakan Algoritma *Particle Swarm Optimiziation* pada aplikasi *Event Management* dibuat dengan menggunakan bahasa *Python* 3.8 untuk memproses kecerdasan buatan Algoritma PSO, sedangkan pada bagian *Back-End* dan *Front-End* menggunakan bahasa *PHP* dan *framework Codeignitner3*.

3.3.1. *Use Case* Diagram

Pada gambar 3.2 terlihat sebuah *Use Case Diagram* yang menunjukkan proses aktivitas secara urut sistem.



Gambar 3.2 *Use Case* Diagram

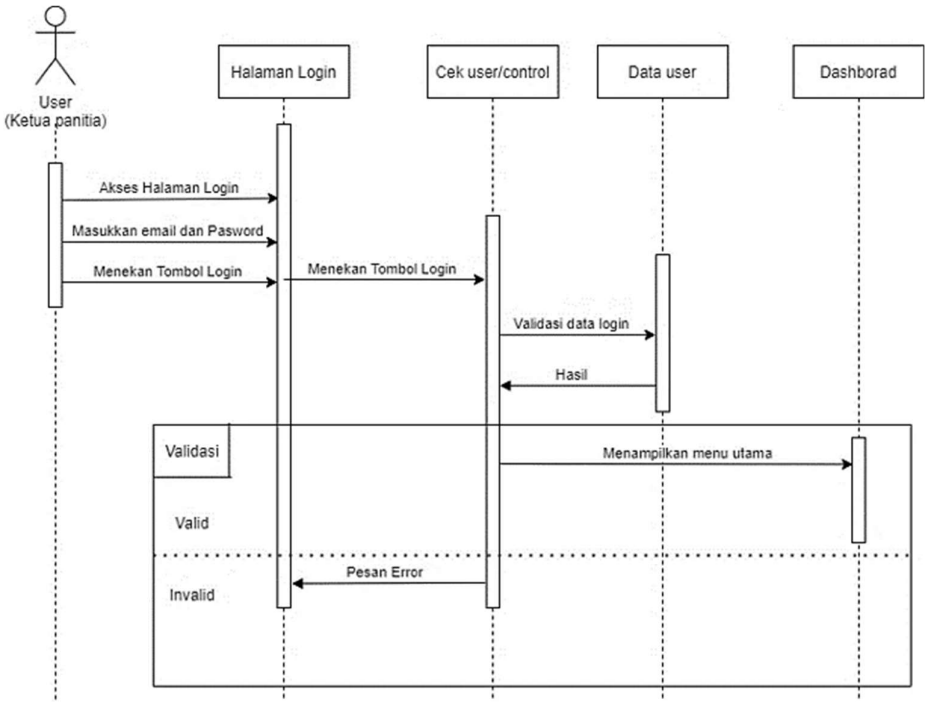
User (ketua *event*) memerlukan sebuah akun untuk dapat mengakses fitur penjadwalan. Setelah berhasil *login*, *user* akan diarahkan untuk mengisi formulir detail pembuatan *event*. Setelah *event* berhasil dibuat maka *user* dapat mengisi jadwal *event*. Setelah *user* mengisi jadwal *event*, *user* dapat mengisi data anggota *event.* Setelah jadwal *event* dan data anggota terisi, *user* dapat membuat penjadwalan anggota secara otomatis menggunakan algoritma PSO dengan cara *generate schedule*.

3.3.2. *Sequence* Diagram

Berdasarkan gambar 3.2 (*Use Case* Diagram) yang telah dijelaskan, akan membuat empat *sequence* diagram untuk menjelaskan urutan interaksi, yaitu :

1. *Sequence* Diagram *Login*

Pada gambar 3.3 adalah *sequence* Diagram *login* yang menjelaskan alur *user* melakukan proses *login.* Berikut *sequence* Diagram proses *login*:

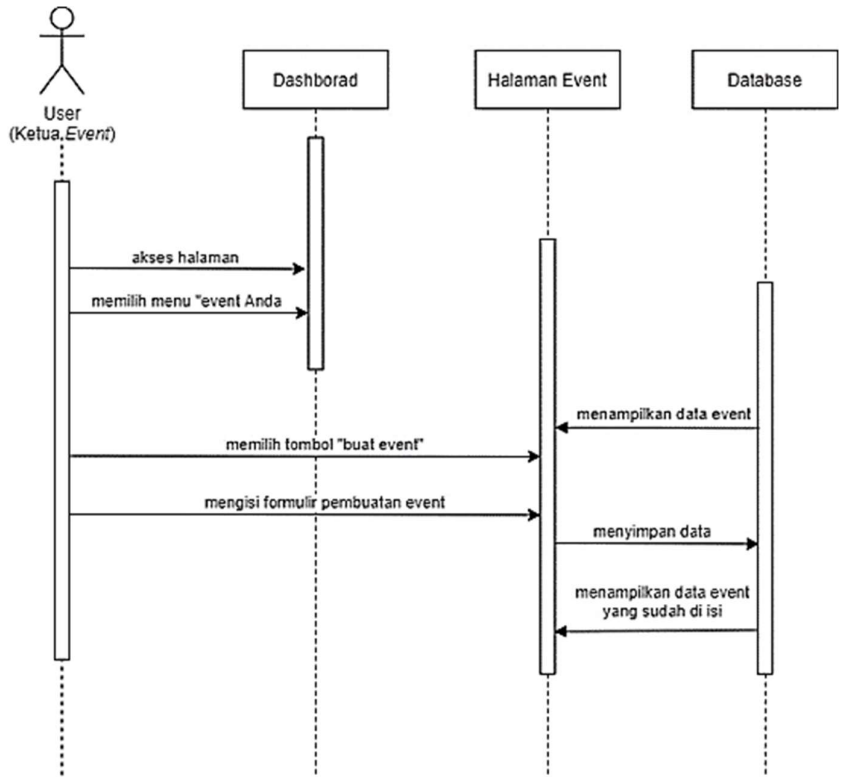


Gambar 3.3 *Sequence* Diagram Login

Langkah pertama *user* dalam menggunakan aplikasi adalah *login* terlebih dahulu. *User* akan mengisi formulir *login* yang berisi email dan *password user* di halaman *login.* Setelah *user* mengisi formulir, user menekan tombol *login.* Lalu sistem akan mengirim data ke *back-end* aplikasi dan melakukan validasi data *login* dan menampilkan hasil. Jika validasi data *login* itu benar maka *user* akan diarahkan ke halaman *dashboard* tetapi bila validasi data *login* maka akan menampilkan pesan *error* dan *user* tetap berada pada halaman *login*.

2. *Sequence* Diagram Membuat *Event*

Pada gambar 3.4 adalah *sequence* Diagram membuat *event* yang menjelaskan alur user melakukan pembuatan *event*. Berikut *sequence* Diagram proses membuat *event*:

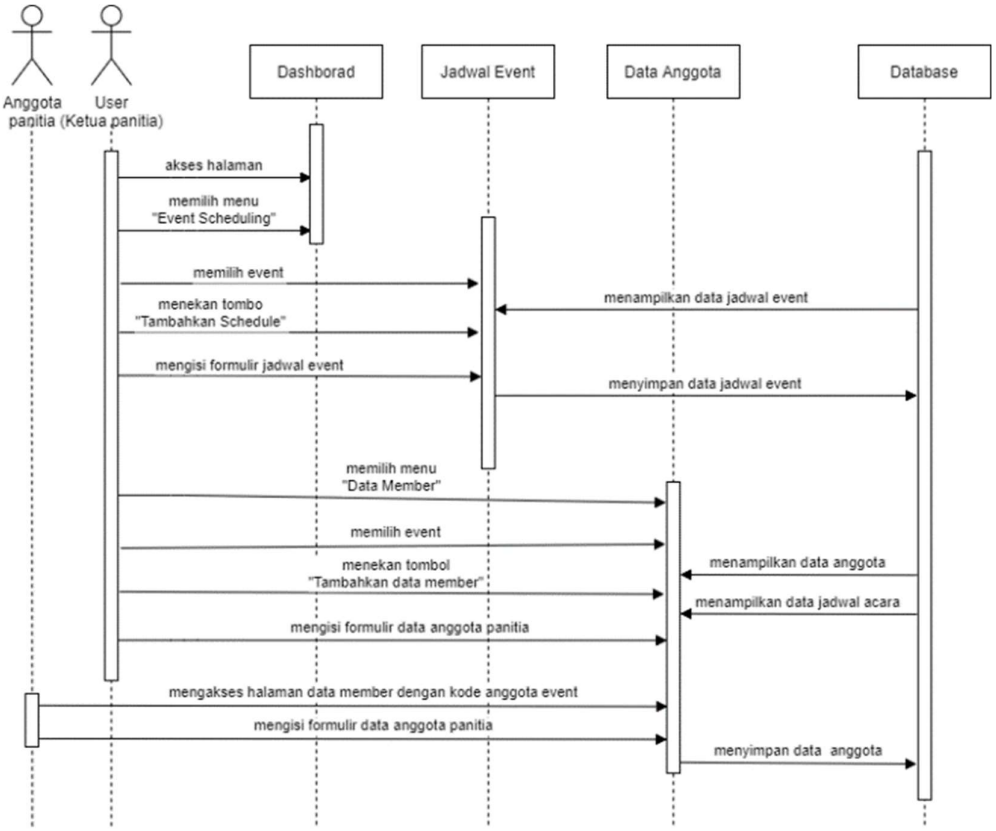


Gambar 3.4 *Sequence* Diagram Membuat *Event*

Ketika *user* berhasil melakukan *login*, *user* dapat membuat *event* di aplikasi. Pada tahap ini *user* akan mengakses halaman *dashboard* dan memilih menu "*Event* Anda". Saat *user* memasuki halaman "*Event* Anda" *database* akan menampilkan *event user*, jika di data *event* ditemukan event *user*. Lalu *user* menekan tombol "Buat *event*" dan mengisi formulir *event*. setelah data *event* diisi akan terjadi proses penyimpanan ke *database* dan menampilkan data *event* yang sudah diisikan.

3. *Sequence* Diagram Mengisi Jadwal *Event* dan Data Anggota Panitia

Pada gambar 3.5 adalah *sequence* Diagram mengisi jadwal *event* dan data anggota panitia yang menjelaskan alur *user* melakukan pengisian jadwal *event* dan data anggota panitia. Berikut *sequence* diagram proses mengisi jadwal *event* dan data anggota panitia*:*



Gambar 3.5 *Sequence* Diagram Mengisi Jadwal *Event* dan Data Anggota Panitia

Ketika *user* berhasil melakukan *login* dan *user* memiliki sebuah *event* di aplikasi, *user* dapat mengakses fitur Penjadwalan di aplikasi. Pada tahap ini *user* akan mengakses halaman *dashboard*, memilih menu "*Event Schedulling*", dan memilih *event*. Saat *user* memasuki halaman "*Event Schedulling*" *database* akan menampilkan jadwal *event user* (jika ada). Lalu *user* menekan tombol "Tambahkan *Schedule*" dan mengisi formulir jadwal *event*. Setelah data jadwal *event* di isi akan terjadi proses penyimpanan ke *database* dan menampilkan data jadwal *event* yang sudah diisikan. Jika dirasa jumlah jadwal *event user* sudah cukup *user* dapat menambahkan data anggota *event* dengan memilih menu "Data Member", memilih *event*, menekan tombol "Tambahkan data Member", dan mengisi formulir data anggota panitia. Anggota *event* juga dapat mengakses formulir data anggota panitia dengan menggunakan kode