

LAPORAN PROYEK AKHIR

“BIPOL TRACKER”



Untuk Memenuhi Tugas Akhir Mata Kuliah Pemrograman Web

Dosen Pengampu:

Ariawan Andi Suhandana, S.Kom., M.T.I. (198501292010121003)

Disusun Oleh:

Eza Musyarof	(2107421002)
Robby Akbar Abdullah	(2107421004)
Shoffan Darul Mufti	(2107421009)
Fajar Firdaus de Roock	(2107421014)
M. Daffa Rasyid	(2107421020)

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK NEGERI JAKARTA

2022

DAFTAR ISI

DAFTAR ISI	2
DAFTAR TABEL	6
DAFTAR GAMBAR	7
BAB I	
PENDAHULUAN	9
1. 1 Latar Belakang	9
1. 2 Rumusan Masalah	9
1. 3 Tujuan	9
1. 4 Manfaat	10
BAB II	
LANDASAN TEORI	11
2. 1 Visual Studio Code	11
2. 2 Git dan Github	11
2. 3 HTML	11
2. 4 CSS	11
2. 5 JavaScript	12
2. 6 Bootstrap	12
2. 7 MySQL	12
2. 8 Python	12
2. 9 Flask	12
2. 10 API	13
BAB III	
HASIL DAN PEMBAHASAN	14
3. 1 Metode Penelitian	14
3. 2 Tahap Penelitian	14
3. 3 Timeline Penggerjaan	15
3. 4 Tugas dan Tanggungjawab	15

3. 5 Analisa Kebutuhan	16
3. 6 Desain Sistem	23
3. 7 Perancangan Database	30
3. 8 Desain Prototype	31
3. 9 Desain Antarmuka	33
3. 10 Program API	38
3. 11 Program Flask	61
3. 12 Pengujian	84
BAB V	
PENUTUP	93
5.1. Kesimpulan	93
5.2. Saran	93
DAFTAR PUSTAKA	94

DAFTAR TABEL

Tabel 3.1. Anggota dan Job Description	17
Tabel 3.2. Kebutuhan Perangkat Lunak	23
Tabel 3.3. Black Box halaman login admin	86
Tabel 3.4. Black Box halaman login admin	86
Tabel 3.5. Black Box halaman database jadwal	86
Tabel 3.6. Black Box halaman database posisi	87
Tabel 3.7. Black Box halaman database bipol	88
Tabel 3.8. Black Box halaman database driver	89
Tabel 3.9. Black Box halaman index utama	90
Tabel 3.10. Black Box halaman login driver	90
Tabel 3.11. Black Box halaman login driver	90
Tabel 3.12. Black Box halaman dashboard driver	91
Tabel 3.13. Black Box halaman landing page	92
Tabel 3.14. UAT Testing	92

DAFTAR GAMBAR

Gambar 3. 1 Web Development Life Cycle	14
Gambar 3. 2 Timeline Pengerjaan	15
Gambar 3. 3 Dokumentasi Survey Bipol Tracker	18
Gambar 3. 4 Dokumentasi Responden Survey Bipol Tracker	18
Gambar 3. 5 Pertanyaan Pertama Survey Bipol Tracker	19
Gambar 3. 6 Jawaban Pertanyaan Pertama	19
Gambar 3. 7 Pertanyaan Kedua Survey Bipol Tracker	19
Gambar 3. 8 Jawaban Pertanyaan Kedua	20
Gambar 3. 9 Use Case Diagram Bipol Tracker	23
Gambar 3. 10 Activity Diagram Halaman Landing Page	24
Gambar 3. 11 Activity Diagram Halaman Dashboard Driver	25
Gambar 3. 12 Activity Diagram Halaman Database Jadwal	26
Gambar 3. 13 Activity Diagram Halaman Database Posisi	27
Gambar 3. 14 Activity Diagram Halaman Bipol	28
Gambar 3. 15 Activity Diagram Halaman Driver	29
Gambar 3. 16 Relasi Database Bipol Tracker	30
Gambar 3. 17 Prototype Jadwal Bipol	31
Gambar 3. 18 Prototype Posisi Bipol	31
Gambar 3. 19 Prototype Login Page	32
Gambar 3. 20 Prototype Dashboard Driver	32
Gambar 3. 21 Prototype Dashboard Driver	33
Gambar 3. 22 Tampilan Jadwal Bipol	33
Gambar 3. 23 Tampilan Posisi Bipol	34
Gambar 3. 24 Tampilan Login Page	34
Gambar 3. 25 Tampilan Dashboard Driver	35

Gambar 3. 26 Tampilan Dashboard Admin Database Jadwal	35
Gambar 3. 27 Tampilan Dashboard Admin Database Posisi	36
Gambar 3. 28 Tampilan Dashboard Admin Database Bipol	36
Gambar 3. 29 Tampilan Dashboard Admin Database Driver	37
Gambar 3. 30 Program API Bipol Tracker	38
Gambar 3. 31 Schema API	39
Gambar 3. 32 Default (/)	40
Gambar 3. 33 Driver Read (/api/driver/)	41
Gambar 3. 34 Driver Read by ID (/api/driver/{id})	42
Gambar 3. 35 Driver Create (/api/driver/create)	43
Gambar 3. 36 Driver Update (/api/driver/update/{id})	44
Gambar 3. 37 Driver Delete (/api/driver/delete/{id})	45
Gambar 3. 38 Bipol Read (/api/bipol/)	46
Gambar 3. 39 Bipol Read by ID (/api/bipol/{id})	47
Gambar 3. 40 Bipol Create (/api/bipol/create)	48
Gambar 3. 41 Bipol Update (/api/bipol/update/{id})	49
Gambar 3. 42 Bipol Delete (/api/bipol/delete/{id})	50
Gambar 3. 43 Jadwal Read (/api/jadwal/)	51
Gambar 3. 44 Jadwal Read by ID (/api/jadwal/{id})	52
Gambar 3. 45 Jadwal Create (/api/jadwal/create)	53
Gambar 3. 46 Jadwal Update (/api/jadwal/update/{id})	54
Gambar 3. 47 Jadwal Delete (/api/jadwal/delete/{id})	55
Gambar 3. 48 Posisi Read (/api/posisi/)	56
Gambar 3. 49 Posisi Read by ID (/api/posisi/{id})	57
Gambar 3. 50 Posisi Read by ID (/api/posisi/{id})	58
Gambar 3. 51 Posisi Update (/api/posisi/update/{id})	59
Gambar 3. 52 Posisi Delete (/api/posisi/delete/{id})	60

BAB I

PENDAHULUAN

1.1 Latar Belakang

Bis Politeknik atau Bipol adalah transportasi yang sering digunakan oleh mahasiswa PNJ karena memudahkan untuk berangkat dan pulang kuliah. Selain itu, bipol juga naik dan turunnya dapat langsung di PNJ tidak seperti Bis Kuning (bikun) yang hanya sampai halte SOR saja. Namun, yang menjadi permasalahan adalah mahasiswa PNJ tidak mengetahui informasi yang pasti mengenai jadwal dan posisi bipol sehingga membuat mahasiswa PNJ menunggu kedatangan bipol tanpa adanya kepastian. Jadwal bipol ini memudahkan mahasiswa PNJ untuk mengatur waktunya lebih efektif dan juga dapat mengetahui bahwa bipol sedang beroperasi atau tidak sehingga dapat mencari alternatif transportasi lain. Oleh karena itu, kami menghadirkan sebuah solusi yang dapat menjawab permasalahan-permasalahan tersebut, yaitu **Bipol Tracker**. Bipol Tracker adalah sebuah aplikasi berbasis website untuk memantau jadwal dan posisi Bis Politeknik (bipol) yang digunakan oleh Mahasiswa PNJ dalam transportasi di lingkungan kampus.

1.2 Rumusan Masalah

Berdasarkan masalah yang ada, permasalahan dapat dirumuskan sebagai berikut:

1. Bagaimana Bipol Tracker ini dapat memudahkan mahasiswa PNJ dalam mengatur waktu menjadi lebih efektif?
2. Bagaimana Bipol Tracker ini dapat meminimalisir kemungkinan mahasiswa PNJ tertinggal bipol?
3. Bagaimana Bipol Tracker ini dapat memberikan informasi yang akurat untuk mahasiswa PNJ?

1.3 Tujuan

Tujuan yang ingin dicapai dalam pembuatan website ini adalah sebagai berikut:

1. Membuat Bipol Tracker yang dapat memudahkan mahasiswa PNJ dalam mengatur waktu menjadi lebih efektif.
2. Membuat Bipol Tracker yang dapat meminimalisir kemungkinan mahasiswa PNJ tertinggal bipol.
3. Membuat Bipol Tracker yang dapat memberikan informasi yang akurat untuk mahasiswa PNJ.

1. 4 Manfaat

1. Bagi Penulis

Sebagai sarana mengimplementasikan ilmu Teknik Multimedia dan Jaringan khususnya mata kuliah Pemrograman Web dalam pengembangan teknologi yang semakin berkembang terutama pada pembuatan website Bipol Tracker.

2. Bagi Pembaca

Sebagai sarana referensi untuk proses pembelajaran dan penelitian selanjutnya.

3. Bagi Pendidikan

Sebagai sumbangsi ilmu yang telah didapat selama belajar di Politeknik Negeri Jakarta untuk menjadi referensi pustaka.

BAB II

LANDASAN TEORI

2. 1 Visual Studio Code

Visual Studio Code adalah editor source code yang dikembangkan oleh Microsoft untuk Windows, Linux dan MacOS. Ini termasuk dukungan untuk debugging, GIT Control yang disematkan, penyorotan sintaks, penyelesaian kode cerdas, cuplikan, dan kode refactoring. Hal ini juga dapat disesuaikan, sehingga pengguna dapat mengubah tema editor, shortcut keyboard, dan preferensi. Visual Studio Code gratis dan open-source, meskipun unduhan resmi berada di bawah lisensi proprietary..

Kode Visual Studio didasarkan pada Elektron, kerangka kerja yang digunakan untuk menyebarkan aplikasi Node.js untuk desktop yang berjalan pada Blinklayout. Meskipun menggunakan kerangka Elektron, Visual Studio Code tidak menggunakan Atom dan menggunakan komponen editor yang sama (diberi kode nama "Monaco") yang digunakan dalam Visual Studio Team Services yang sebelumnya disebut Visual Studio Online (Lardinois, 2015) .

2. 2 Git dan Github

Git adalah sistem version control bersifat open-source khusus yang dibuat oleh Linus Torvalds pada tahun 2005. Pada dasarnya, Git merupakan version control versi yang terdistribusi. Jadi, seluruh basis kode dan riwayatnya dapat tersedia di setiap komputer developer sehingga memungkinkan branching dan merging dapat dilakukan dengan mudah. Saat ini, sebagian besar proyek pengembangan perangkat lunak mengandalkan Git sebagai version control, baik itu proyek komersial ataupun open source.

Github adalah platform yang menawarkan layanan repository Git berbasis cloud. Tim developer dapat memanfaatkan GitHub untuk kolaborasi pengembangan software serta sebagai sistem version control. Jadi, dengan menggunakan GitHub, Anda dan tim developer yang lain dapat bekerja sama dalam suatu proyek dengan lebih mudah.

2. 3 HTML

Hypertext markup language (HTML) merupakan bahasa dasar pembuatan web. HTML menggunakan tanda (mark), untuk menandai bagian-bagian dari text. HTML disebut sebagai bahasa dasar, karena dalam membuat web, jika hanya menggunakan HTML maka tampilan web terasa hambar (Rerung, 2018:18). Hypertext markup language (HTML) merupakan bahasa pemrograman dasar untuk mengelola website. Akan tetapi, HTML hanya terbatas pada pembuatan website statis (website yang tidak dapat berinteraksi aktif dengan user). Maka dari itu, HTML biasa dikombinasikan dengan bahasa pemrograman web lainnya (Wardana, 2016:3)

2. 4 CSS

Cascading Style Sheet (CSS) merupakan aturan untuk mengendalikan beberapa komponen dalam sebuah web sehingga akan lebih terstruktur dan seragam. CSS bukan merupakan bahasa pemrograman. CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran border, warna border, warna hyperlink, warna mouse over, spasi antar paragraf, spasi antar teks, margin kiri, kanan, atas, bawah, dan parameter lainnya. CSS adalah bahasa style sheet yang digunakan untuk mengatur tampilan dokumen. Dengan adanya CSS memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda.

2. 5 JavaScript

JavaScript merupakan sebuah scripting language yang biasa digunakan untuk pembangunan sebuah dynamic website. JavaScript banyak digunakan karena relasi yang sangat besar dengan HTML dan CSS. JavaScript memiliki banyak fungsi pada basis Website, namun javascript hanya bisa dijalankan dengan server-side (Node.JS).

2. 6 Bootstrap

Bootstrap merupakan framework HTML dan CSS yang menyediakan kumpulan komponen-komponen antarmuka dasar pada web yang telah dirancang sedemikian rupa untuk mempercepat pekerjaan. Menurut Hasin (2015) Bootstrap juga menyediakan sarana untuk membangun layout halaman dengan mudah dan rapi, serta modifikasi pada tampilan dasar HTML untuk membuat seluruh halaman web yang dikembangkan senada dengan komponen-komponen lainnya.

2. 7 MySQL

MySQL merupakan RDBMS (Relational Database Management System) server. RDBMS adalah program yang memungkinkan pengguna database untuk membuat, mengelola, dan menggunakan data pada suatu model relational. Dengan demikian, tabel-tabel yang ada pada database memiliki relasi antara satu tabel dengan tabel lainnya.

2. 8 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, utamanya, namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis.

Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa script. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi[12], Saat ini kode python dapat dijalankan di berbagai platform sistem operasi.

2. 9 Flask

Flask adalah sebuah web framework yang ditulis dengan bahasa Python dan tergolong sebagai jenis microframework. Flask berfungsi sebagai kerangka kerja aplikasi dan tampilan dari suatu web. Dengan menggunakan Flask dan bahasa Python, pengembang dapat membuat sebuah web yang terstruktur dan dapat mengatur behaviour suatu web dengan lebih mudah.

Flask termasuk pada jenis microframework karena tidak memerlukan suatu alat atau pustaka tertentu dalam penggunaannya. Sebagian besar fungsi dan komponen umum seperti validasi form, database, dan sebagainya tidak terpasang secara default di Flask. Hal ini dikarenakan fungsi dan komponen-komponen tersebut sudah disediakan oleh pihak ketiga dan Flask dapat menggunakan ekstensi 29 yang membuat fitur dan komponen-komponen tersebut seakan diimplementasikan oleh Flask sendiri.

2. 10 API

API (Application Programming Interface) merupakan sebuah interface yang dapat diimplementasikan dengan menggunakan perangkat lunak (software) sehingga perangkat 7 lunak tersebut dapat berinteraksi dengan perangkat lunak lainnya, seperti halnya tampilan interface user yang memungkinkan user untuk berinteraksi dengan komputer (Prasetyadi, 2011). Dengan memanfaatkan API, developer dapat memanfaatkan beberapa perangkat lunak untuk melakukan suatu proses. Selain itu, tujuan dari penggunaan API adalah mempercepat proses pengembangan sebuah sistem atau aplikasi dengan menggunakan fungsi-fungsi secara terpisah, sehingga developer tidak perlu membuat fungsi atau fitur yang serupa.

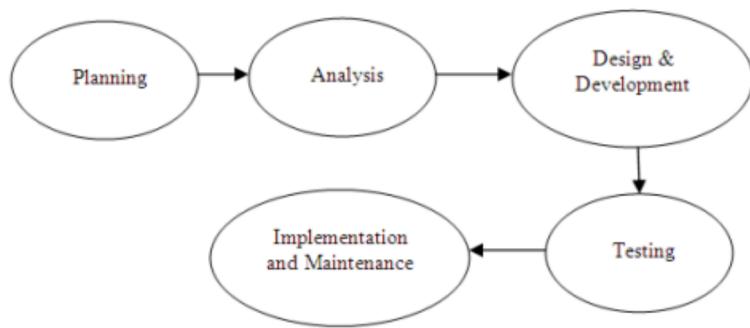
API bekerja dengan cara membantu aplikasi berinteraksi dengan library dengan mengikuti serangkaian aturan yang ditentukan sebelumnya oleh API itu sendiri. Pendekatan ini memudahkan developer untuk membuat aplikasi yang berkomunikasi dengan berbagai library tanpa harus memikirkan kembali strategi yang digunakan selama semua library mengikuti API yang sama.

BAB III

HASIL DAN PEMBAHASAN

3. 1 Metode Penelitian

Metode penelitian yang digunakan untuk mebangun website ini adalah metode Web Development Life Cycle (WDLC).



Gambar 3. 1 Web Development Life Cycle

3. 2 Tahap Penelitian

a. *Planning*

Merupakan tahapan awal perancangan sebuah website dengan metode WDLC. Langkah pertama yaitu dengan mengidentifikasi tujuan dan sasaran dari website yang akan dibangun. merupakan langkah awal dalam proses perencanaan. Setelah tujuan diketahui, selanjutnya yaitu memahami kriteria pengguna sistem. Kemudian menentukan teknologi website yang akan digunakan dan mengidentifikasi siapa saja nantinya akan terlibat dalam website.

b. *Analysis*

Pada tahapan dilakukannya identifikasi kebutuhan pengguna dengan mengumpulkan informasi dari pengguna dan menganalisa fungsi dari sistem yang akan dibuat, data apa saja yang dibutuhkan dan dari mana data tersebut dikumpulkan serta apa hasil yang ingin didapatkan dari sistem. Setelah hal tersebut dilakukan, hal tersebut dapat mendukung fitur yang akan ada didalam website

c. *Design & Development*

Tahapan ini merupakan persiapan untuk merancang desain atau blueprint dari website yang akan dibuat. Selain itu juga mempersiapkan berbagai represtasi diagram dari objek logis dan fisik. Hasilnya yaitu seperti model data, model proses, model penyajian, prototype, hingga development website.

d. *Testing*

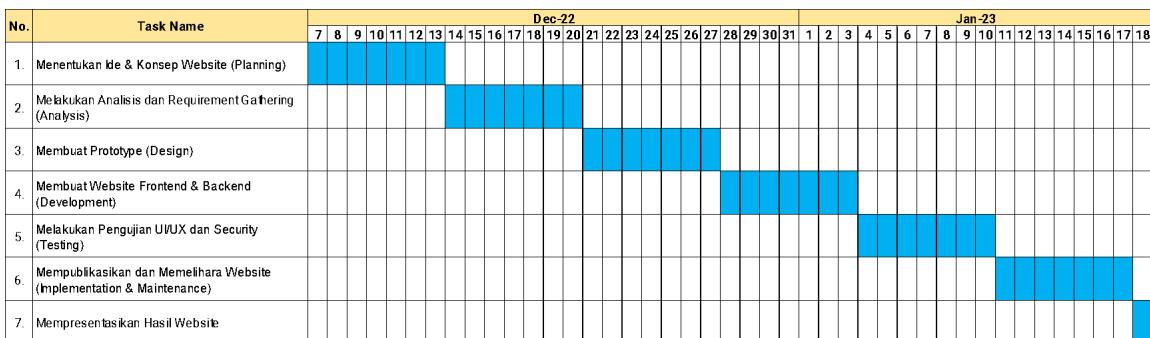
Testing merupakan tahapan untuk menguji hasil kerja dari web site yang dirancang, apakah hasil website yang sudah dibuat sama seperti harapan dari pengguna, mulai dari informasi yang dibutuhkan hingga performa yang didapatkan. Hal-hal yang di uji dalam tahap ini mencakup konten, fungsi, usability dan akurasi sistem.

e. *Implementation & Maintenance*

Dalam tahap implementasi, website sudah bisa digunakan oleh pengguna dan dapat berinteraksi langsung dengan sistem dan pengguna mendapatkan kesempatan untuk menggunakan website tersebut.

3.3 Timeline Pengerjaan

Pengerjaan website ini dilakukan dalam waktu 43 hari dari tanggal 7 Desember 2022 sampai tanggal 18 Januari 2023, dengan keterangan sebagai berikut.



Gambar 3.2 Timeline Pengerjaan

3.4 Tugas dan Tanggungjawab

3.4.1 Anggota dan Job Description

Tabel 3.1. Anggota dan Job Description

No	NIM	Nama	Job Desc
1	2107421004	Robby Akbar Abdullah	Project Manager, Back-End Developer
2	2107421002	Eza Musyarof	Back-End Developer
3	2107421009	Shoffan Darul Mufti	Back-End Developer

4	2107421014	Fajar Firdaus de Roock	UI/UX, Front-End Developer
5	2107421020	M. Daffa Rasyid	Front-End Developer

3. 4. 2 Tugas dan Tanggungjawab

a. Project Manager

- Mengatur & Memantau Jalannya Kerja Tim (Robby)
- Mengatur Biaya (Robby)

b. UI/UX

- Membuat Logo (Fajar)
- Membuat Prototype (Fajar)

c. Front End

- Membuat Landing Page (Fajar & Daffa)
- Membuat Dashboard Admin & Driver (Fajar & Daffa)

d. Back End

- Database (Eza)
- Flask (Robby & Eza)
- API (Shoffan)
- Hosting (Shoffan)

3. 5 Analisa Kebutuhan

3. 5. 1 Kebutuhan Pengguna

Untuk mengetahui kebutuhan pengguna, maka perlu dilakukan sebuah survei yang harus dilakukan kepada pengguna yang bersangkutan, yaitu mahasiswa PNJ sebagai penumbang dari bipol nantinya.

a. Kuesioner

Platform : Google Form

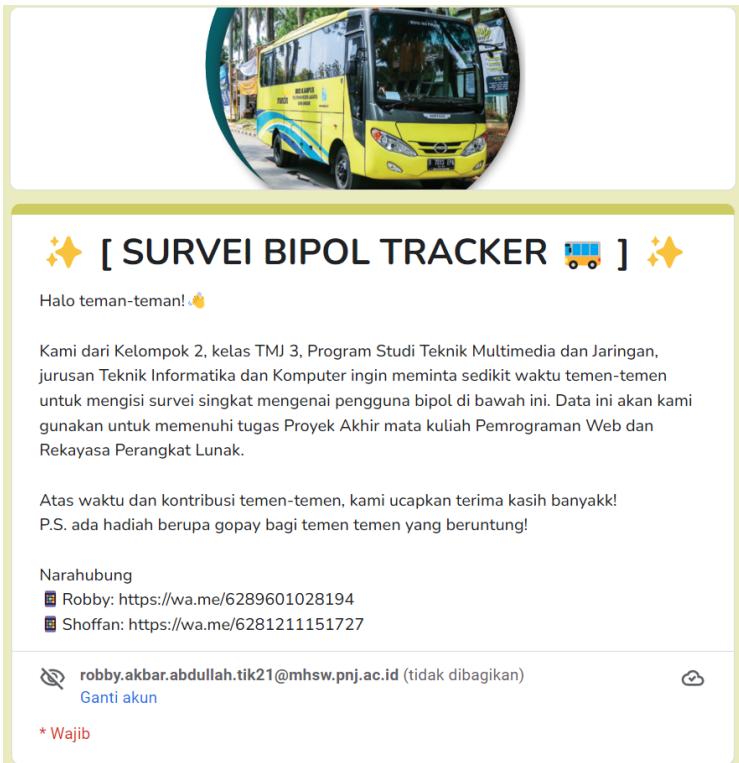
Link survei : <https://s.pnj.ac.id/survei-bipol-tracker>

Target survei : Mahasiswa PNJ

b. Pertanyaan

- 1) Nama
- 2) Jurusan
- 3) No. HP
- 4) Kamu sering naik bipol gaa?
 - Sering
 - Kadang-kadang
 - Ga pernah
- 5) Biasanya kamu naik/turun bipol di halte mana ajaa?
(pilihan halte yang dilalui bipol)
- 6) Puas gaa dengan layanan bipol?
(skala 1 – 5, yaitu tidak puas – puas bgt bgt bgt)
- 7) Alasannya apaa puas/tidak puas dengan layanan bipol?
- 8) Kamu butuh website buat tracking jadwal bipol ga sihh?
(skala 1 – 5, yaitu tidak butuh – butuh bgt bgt bgt)
- 9) Alasannya apaa butuh/tidak butuh website tersebut?
- 10) Kalau butuh, mau fitur apa ajaa?
 - Tampilan jadwal bipol
 - Posisi bipol
 - Lainnya

c. Dokumentasi Survey



Gambar 3. 3 Dokumentasi Survey Bipol Tracker

A screenshot of a survey response summary. At the top, there are tabs for "Pertanyaan", "Jawaban 72", and "Setelan". The "Jawaban 72" tab is selected. Below it, it says "72 jawaban". There are three buttons: a green plus sign for adding more responses, a three-dot menu, and a yellow circle with a checkmark labeled "Menerima jawaban". At the bottom, there are three categories: "Ringkasan" (selected), "Pertanyaan", and "Individual".

Gambar 3. 4 Dokumentasi Responden Survey Bipol Tracker

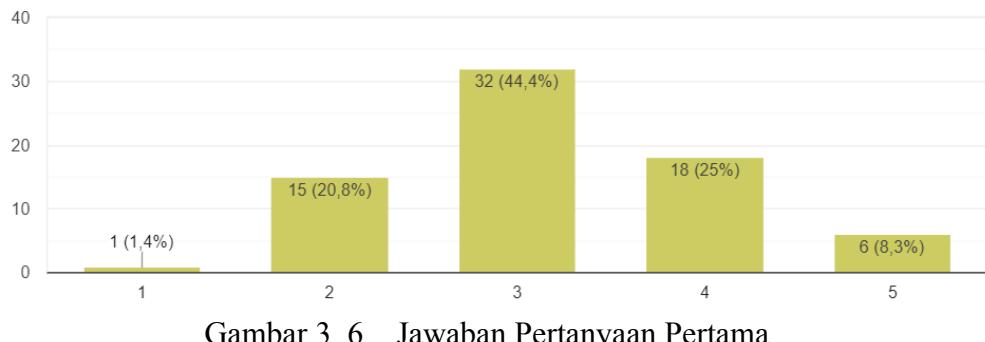
d. Hasil Survey

Berdasarkan hasil survei kami kepada 72 mahasiswa dari semua jurusan di PNJ, sebanyak 63,9% sering naik bipol, 33,3% kadang-kadang naik bipol, dan 2,8% ga pernah naik bipol. Mahasiswa paling banyak naik/turun bipol di 3 halte utama, yaitu halte Stasiun UI, halte Pocin, dan PNJ, sedangkan halte sisanya hanya beberapa orang saja.

Persentase kepuasan mahasiswa terhadap layanan bipol dengan skala 1-5 sebagai berikut:



Gambar 3. 5 Pertanyaan Pertama Survey Bipol Tracker

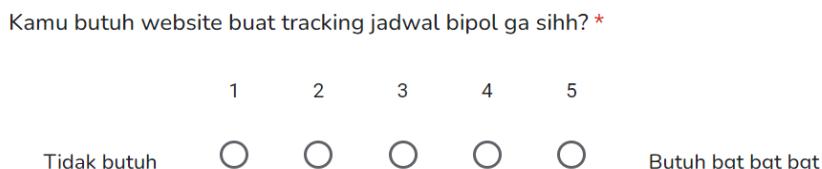


Gambar 3. 6 Jawaban Pertanyaan Pertama

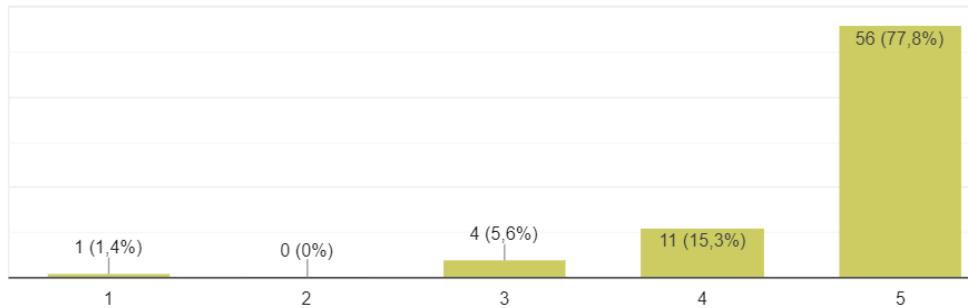
Alasan mahasiswa puas / tidak puas dengan layanan bipol, yaitu:

- Puas membantu mahasiswa dalam transportasi ke PNJ, tidak perlu jalan dari halte SOR, dan bisnya bersih dan terawat
- Tidak puas jadwal jarak kedatangan antarbis tidak menentu, tidak adanya handgrip, armada kurang banyak sehingga terjadi kepadatan penumpang, dan bipol hanya beroperasi pagi dan sore hari

Persentase kebutuhan mahasiswa terhadap website Bipol Tracker dengan skala 1-5 sebagai berikut:



Gambar 3. 7 Pertanyaan Kedua Survey Bipol Tracker



Gambar 3. 8 Jawaban Pertanyaan Kedua

Alasan mahasiswa butuh / tidak butuh dengan layanan bipol, yaitu:

- Butuh melihat jadwal dan posisi bipol sehingga tidak tertinggal bipol, dapat menyesuaikan jadwal berangkat-pulang, dapat mengetahui bipol beroperasi atau tidak, dan decision making antara bipol atau bikun
- Tidak butuh berdasarkan diagram di atas mayoritas membutuhkan website Bipol Tracker

Fitur-fitur yang diinginkan mahasiswa, yaitu:

- Tampilan jadwal bipol
- Posisi bipol
- Status kapasitas penumpang

e. *Functional Requirement* dan *Non-Functional Requirement*

1) Functional Requirement (FR)

- Tampilan jadwal bipol
- Posisi bipol
- Status kapasitas penumpang

2) Non-Functional Requirement (NFR)

- Reliability : website dapat berjalan dengan baik
- Performance : performa website tinggi
- Usability : website responsif dan mudah digunakan
- Security : semua data dipastikan aman

3. 5. 2 Kebutuhan Perangkat Lunak

Tabel 3.2. Kebutuhan Perangkat Lunak

No.	Kebutuhan	Alat
1.	Operation System	Windows 11
2.	Code Editor	Visual Studio Code
3.	Code Manager	Github dan Git
4.	Programming Language	Python
5.	Front End	HTML, CSS, dan Javascript
6.	Framework	Flask dan Bootstrap
7.	Database	MySQL
8.	Database Manager	phpMyAdmin
9.	API	Fast API
10.	Server	Lokal

3. 5. 3 Ide dan Konsep Website

a. Logo



b. Deskripsi

Bipol Tracker adalah sebuah aplikasi berbasis website untuk memantau jadwal dan posisi Bis Politeknik (bipol) yang digunakan oleh Mahasiswa PNJ dalam transportasi di lingkungan kampus.

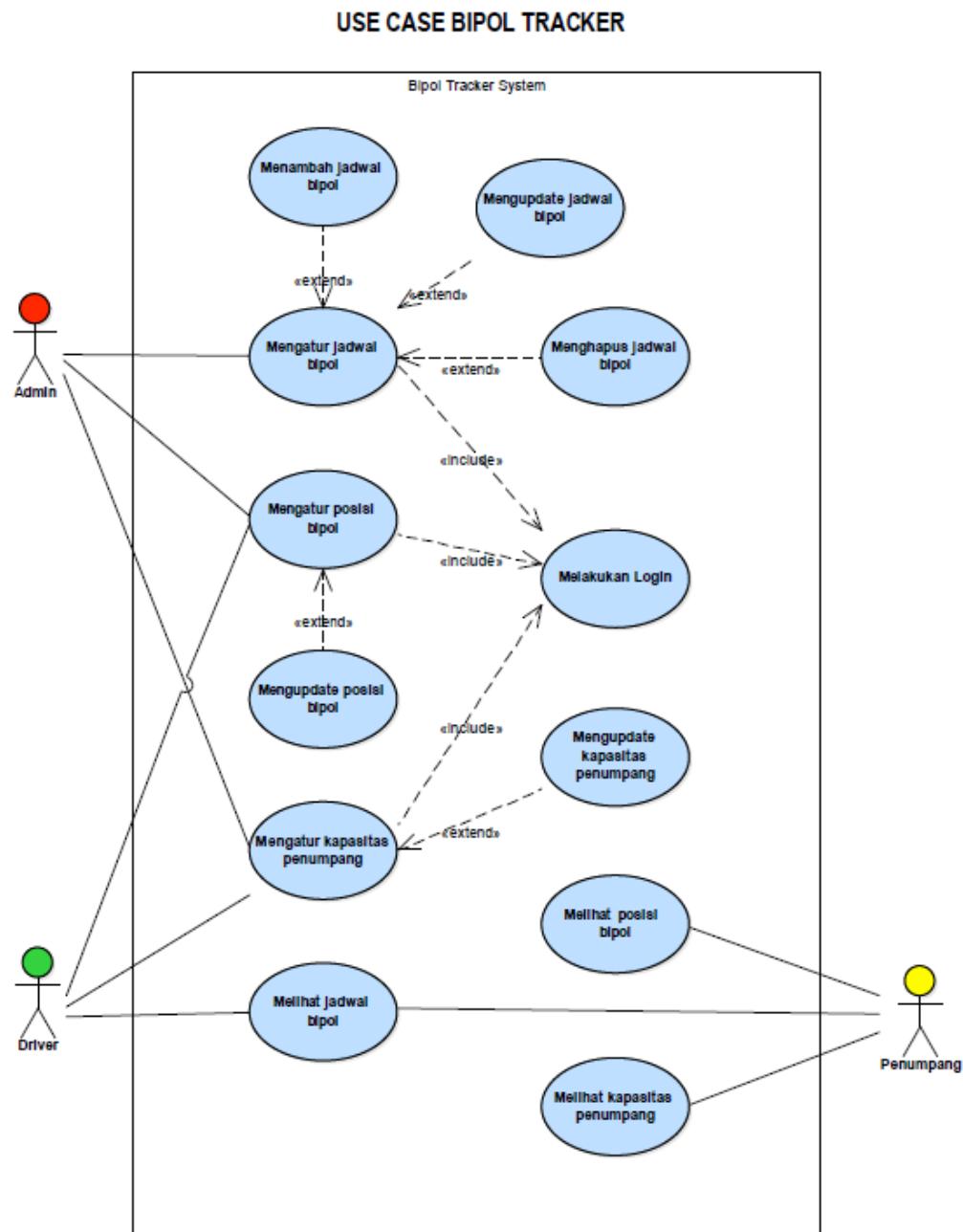
c. Cara Kerja

- Admin dapat mengatur jadwal bipol
- Driver dapat menyesuaikan jadwal dan posisi sesuai waktu yang telah ditentukan
- Mahasiswa dapat memantau jadwal bipol dari website

3.6 Desain Sistem

3.6.1 Use Case Diagram

Berikut adalah use case diagram dari website bipolar tracker. Sistem terdiri dari 3 aktor yaitu admin, driver, dan penumpang.

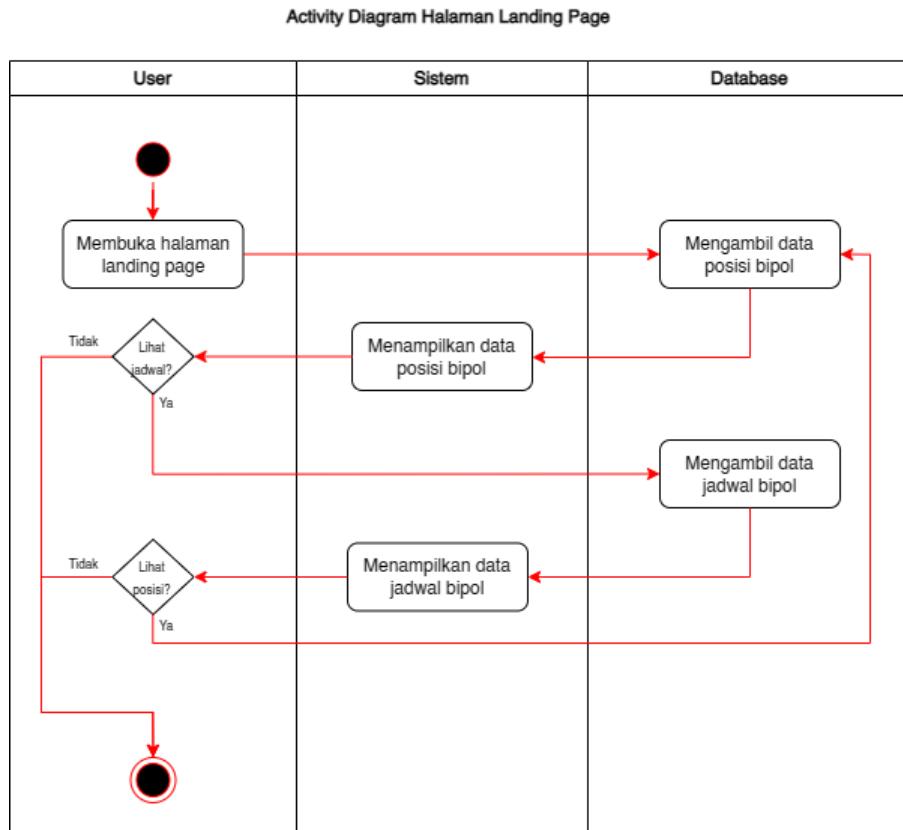


Gambar 3.9 Use Case Diagram Bipol Tracker

3. 6. 2 Activity Diagram

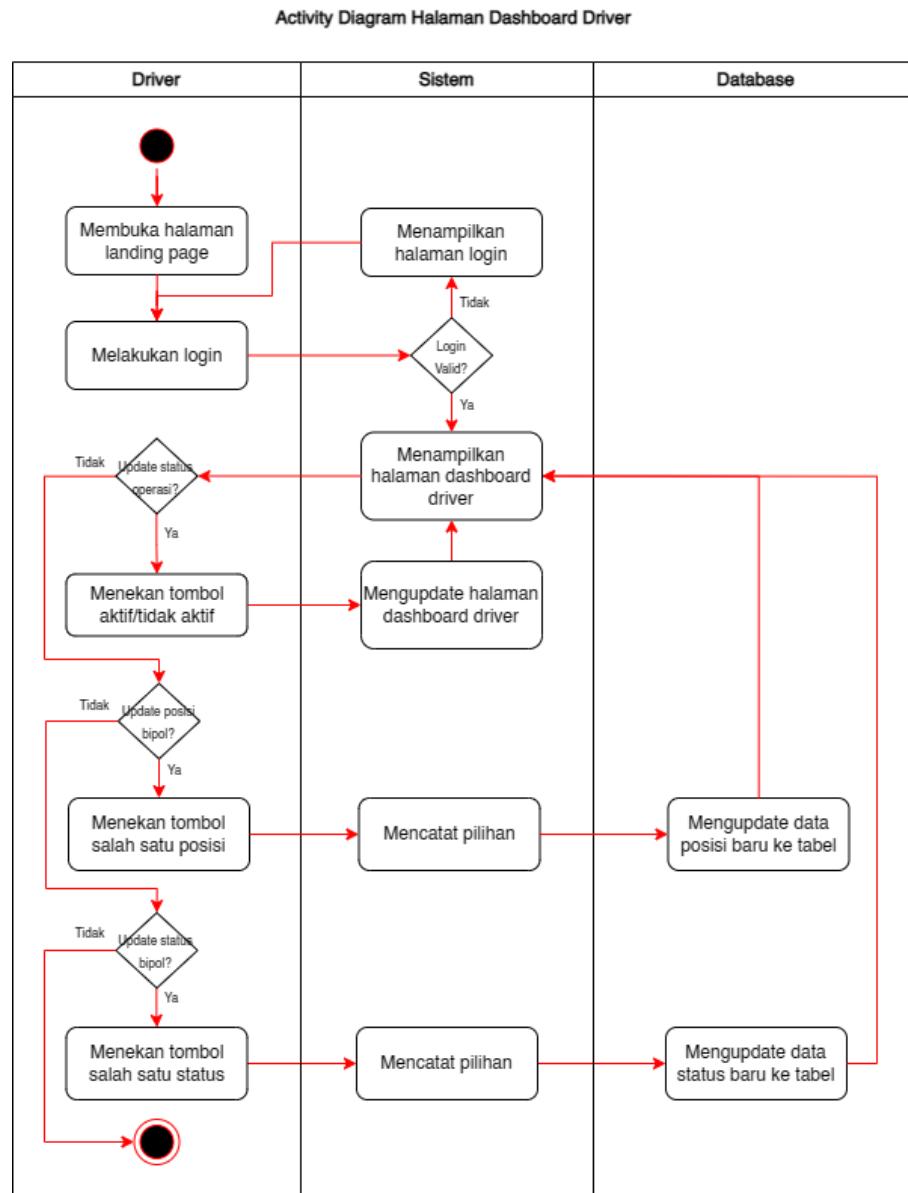
Activity diagram merupakan gambaran dari alur kerja yang berisi aktivitas dan tindakan, yang juga dapat berisi pilihan, pengulangan, dan *concurrency*. Activity diagram dibuat untuk menjelaskan aktivitas dari aktor dalam suatu sistem.

a. Activity Diagram Halaman Landing Page



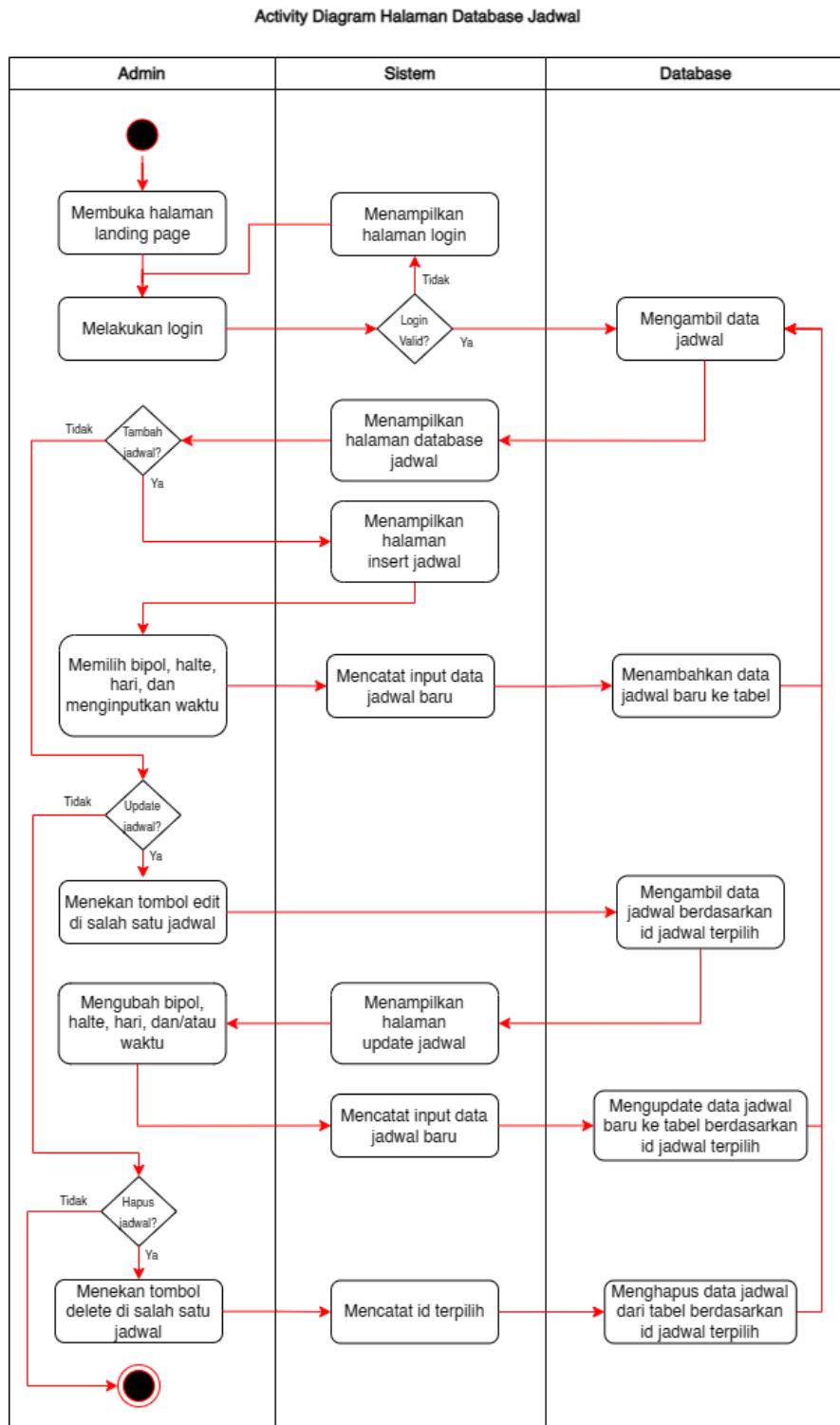
Gambar 3. 10 Activity Diagram Halaman Landing Page

b. Activity Diagram Halaman Dashboard Driver



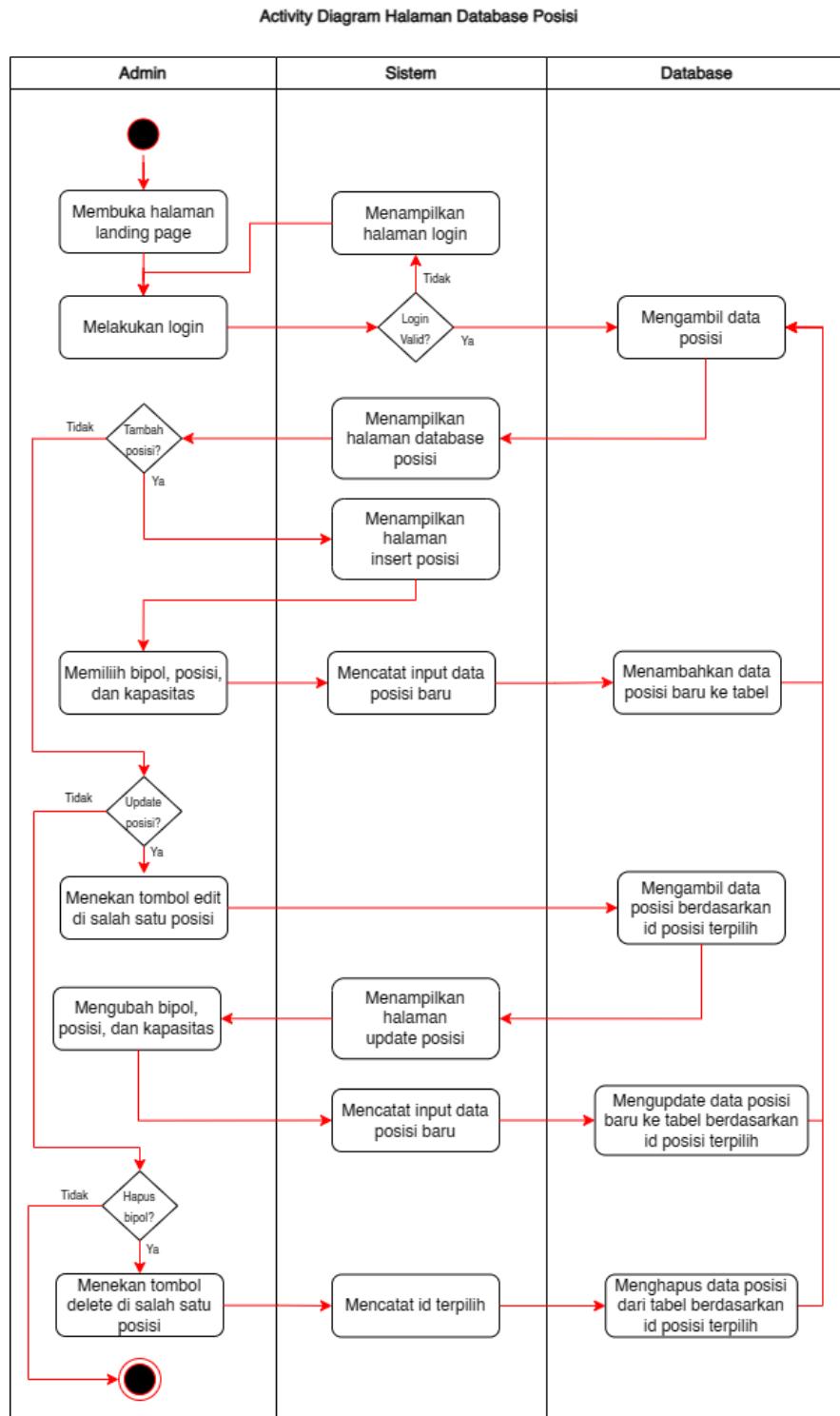
Gambar 3. 11 Activity Diagram Halaman Dashboard Driver

c. Activity Diagram Halaman Database Jadwal



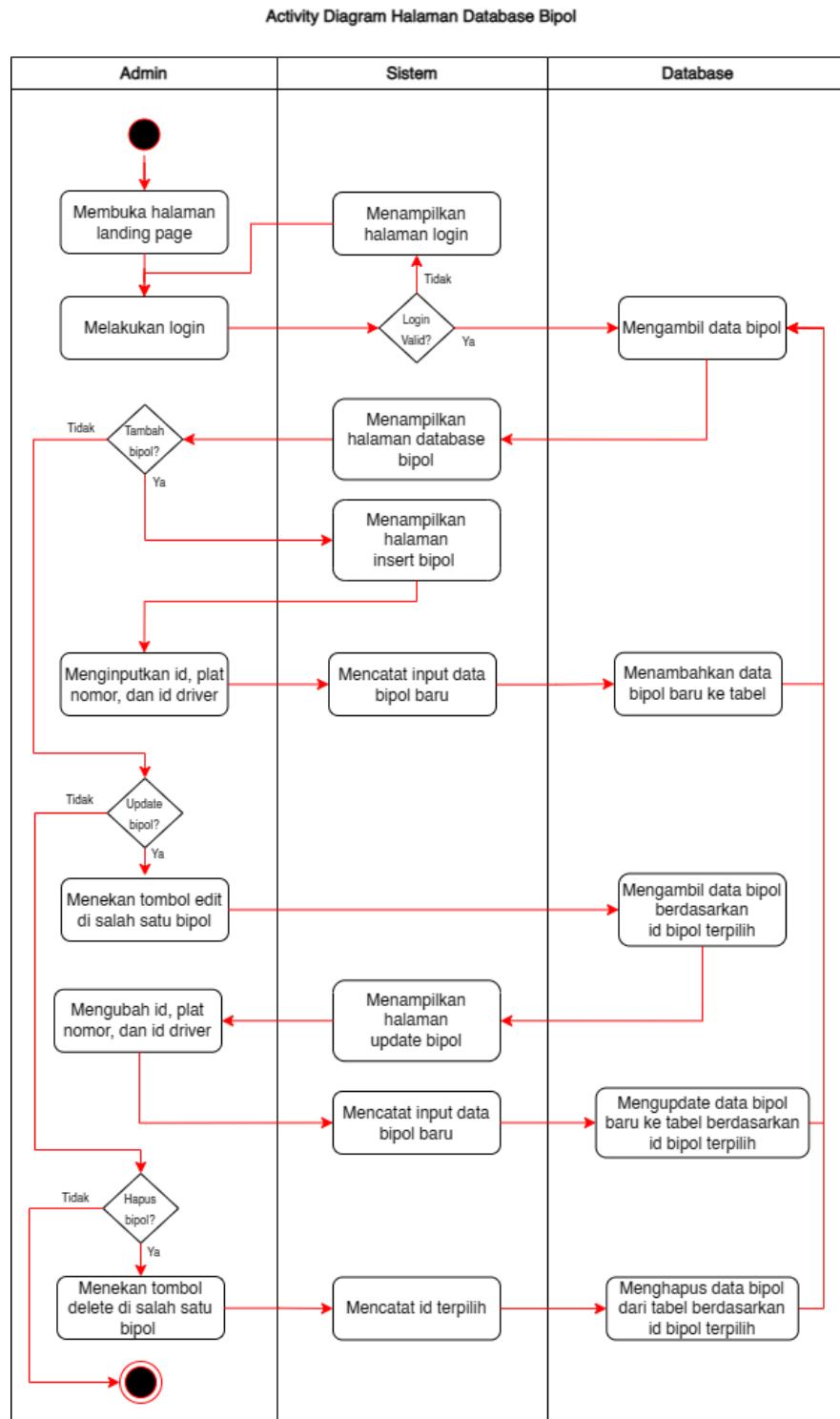
Gambar 3. 12 Activity Diagram Halaman Database Jadwal

d. Activity Diagram Halaman Database Posisi



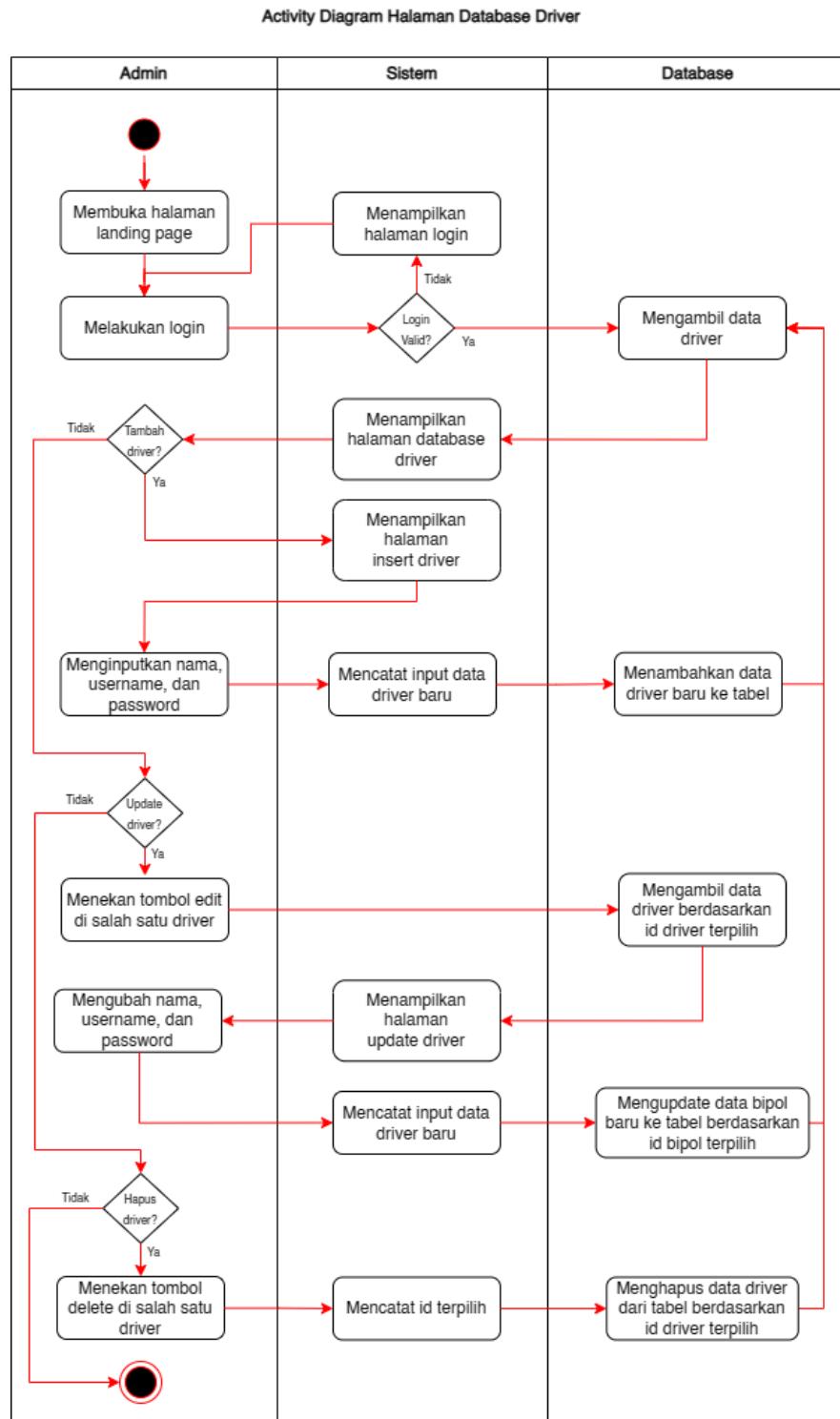
Gambar 3. 13 Activity Diagram Halaman Database Posisi

e. Activity Diagram Halaman Database Bipol



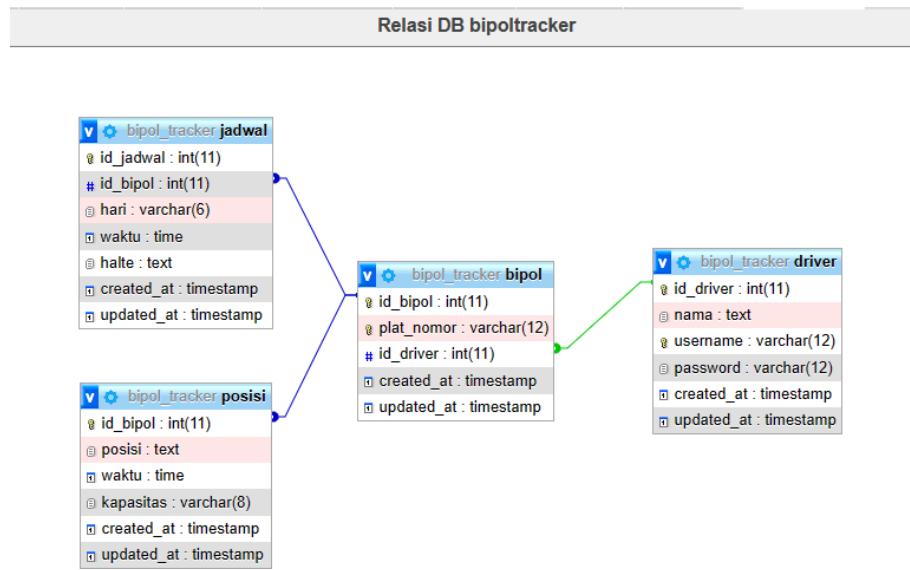
Gambar 3. 14 Activity Diagram Halaman Bipol

f. Activity Diagram Halaman Database Driver



Gambar 3. 15 Activity Diagram Halaman Driver

3.7 Perancangan Database



Gambar 3.16 Relasi Database Bipol Tracker

Relasi database yang digunakan:

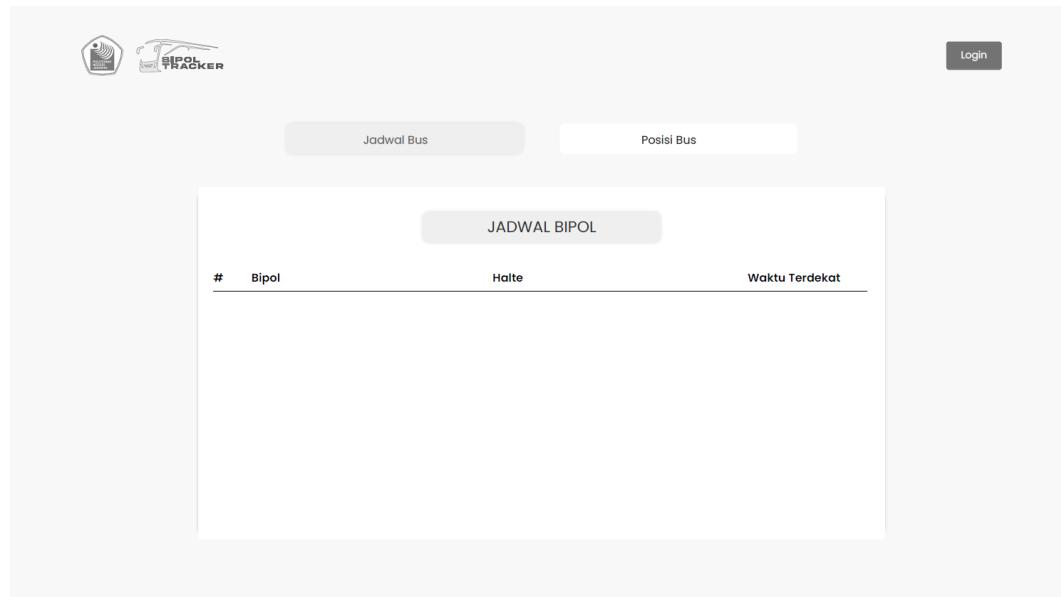
- Tabel bipol to tabel jadwal (**one to many**) = satu bipol punya banyak jadwal
- Tabel bipol to tabel posisi (**one to one**) = satu bipol punya satu posisi
- Tabel bipol to tabel driver (**one to one**) = satu bipol punya satu driver

Keterangan:

- **Tabel jadwal relasi dengan tabel bipol**, yaitu id_bipol di tabel jadwal mengambil dari id_bipol di tabel bipol
- **Tabel bipol relasi dengan tabel driver**, yaitu id_driver di tabel bipol mengambil dari id_driver di tabel driver
- **Tabel posisi relasi dengan tabel bipol**, yaitu id_bipol di tabel posisi mengambil dari id_bipol di tabel bipol

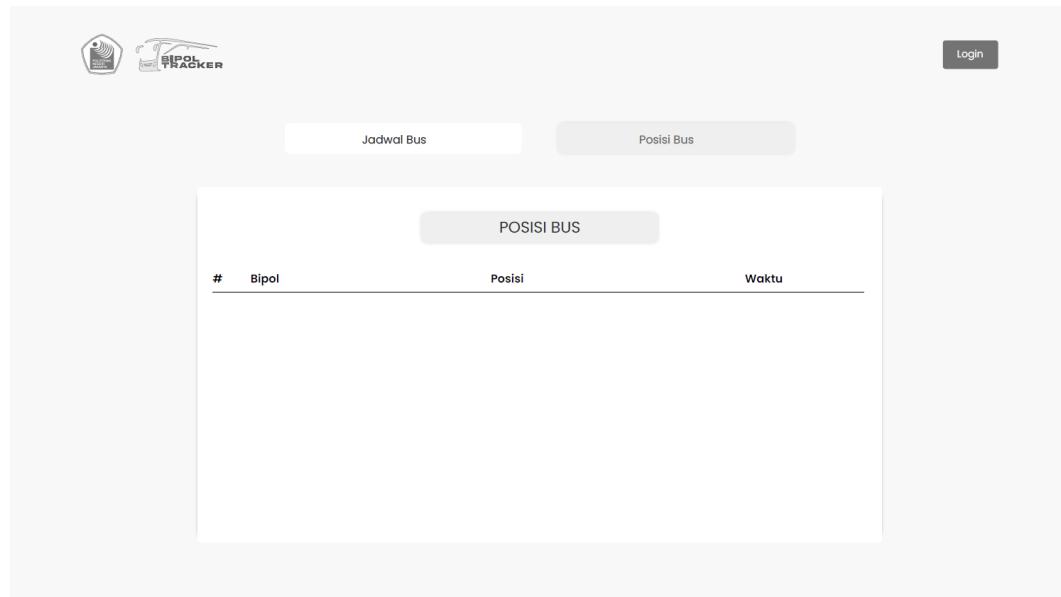
3.8 Desain Prototype

3.7.1 Prototype Jadwal Bipol



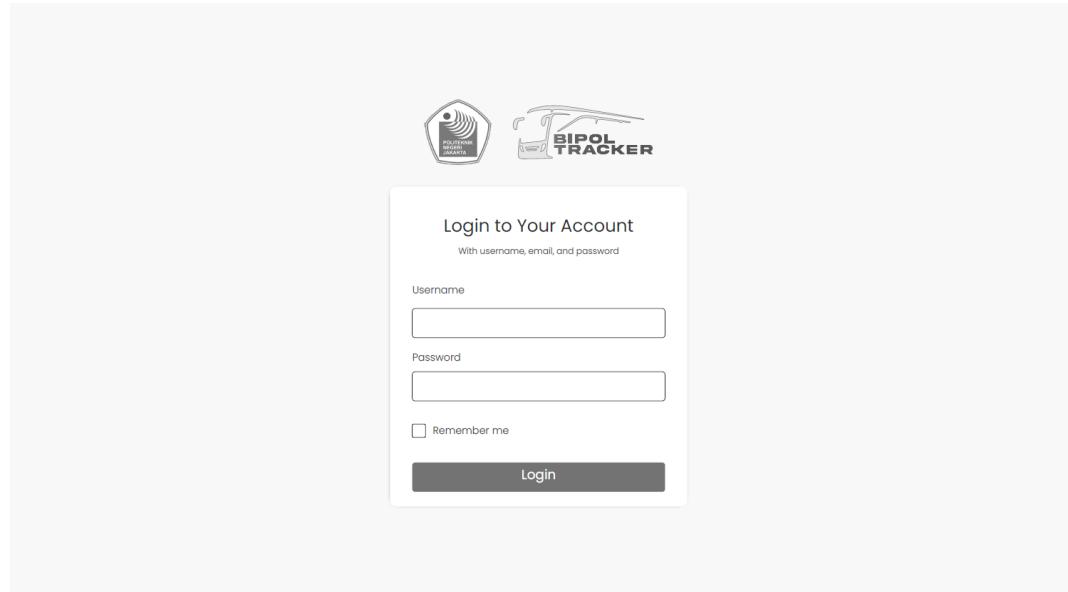
Gambar 3. 17 Prototype Jadwal Bipol

3.7.2 Prototype Posisi Bipol



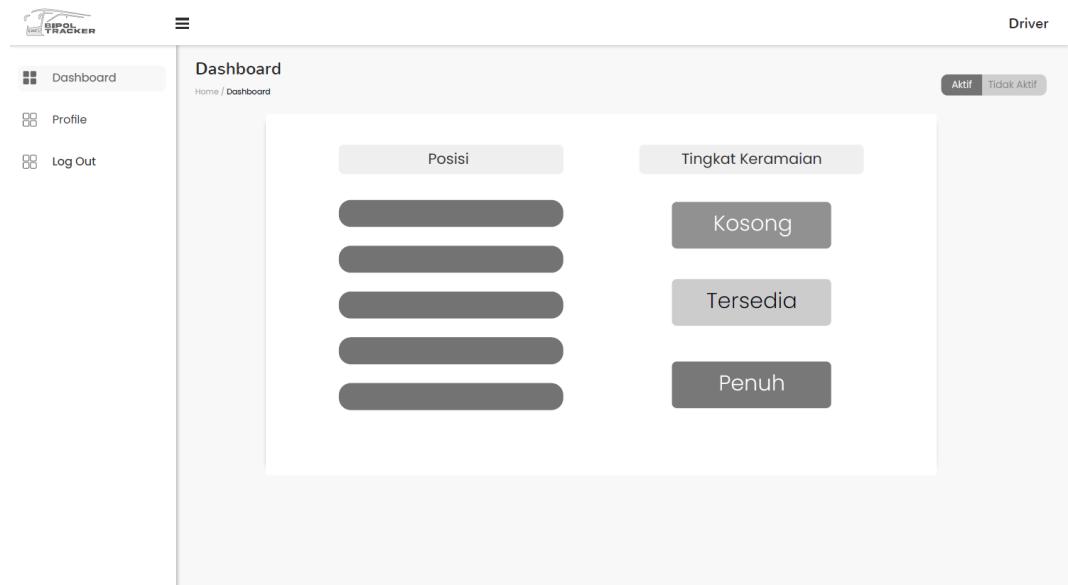
Gambar 3. 18 Prototype Posisi Bipol

3. 7. 3 Prototype Login Page



Gambar 3. 19 Prototype Login Page

3. 7. 4 Prototype Dashboard Driver



Gambar 3. 20 Prototype Dashboard Driver

3. 7. 5 Prototype Dashboard Admin

The screenshot shows the 'Database Jadwal' section of the prototype dashboard. On the left, there is a sidebar with icons for Dashboard, Database Jadwal (selected), Database Posisi, Database Bipol, Database Driver, My Website, Profile, and Log Out. The main area has a header 'Database Jadwal' with 'Home / Database Jadwal'. Below it is a table with columns 'Bipol', 'Halte', 'Waktu', and 'Aksi'. A 'Add' button is at the top left of the table area.

Gambar 3. 21 Prototype Dashboard Driver

3. 9 Desain Antarmuka

3. 8. 1 Tampilan Jadwal Bipol

The screenshot shows the 'jadwal Bipol' section. At the top right is a 'Login' button. Below it is a navigation bar with 'jadwal' (highlighted in blue) and 'Posisi'. The main content is a table titled 'jadwal Bipol' with columns 'Bipol', 'Halte', and 'Estimasi'. The table lists bus routes and their estimated arrival times at various stops.

Bipol	Halte	Estimasi
1 - B 2301 PNJ	PNJ	07:00
2 - B 2302 JNP	PNJ	07:05
1 - B 2301 PNJ	Pondok Cina	07:10
3 - B 2303 NJP	PNJ	07:10
2 - B 2302 JNP	Pondok Cina	07:15
1 - B 2301 PNJ	Stasiun UI	07:20
3 - B 2303 NJP	Pondok Cina	07:20
4 - B 2304 PJN	PNJ	07:20
2 - B 2302 JNP	Stasiun UI	07:25
5 - B 2305 JPN	PNJ	07:25

Gambar 3. 22 Tampilan Jadwal Bipol

3. 8. 2 Tampilan Posisi Bipol

The screenshot shows a web-based application for tracking bipol positions. At the top right is a 'Login' button. Below it, there are two tabs: 'Jadwal' (selected) and 'Posisi'. A sub-header 'Posisi Bipol' is displayed above a table. The table has three columns: 'Bipol', 'Posisi', and 'Status'. The data is as follows:

Bipol	Posisi	Status
1 - B 2301 PNJ	Stasiun UI	Tersedia
2 - B 2302 JNP	Pondok Cina	Tersedia
3 - B 2303 NJP	Stasiun UI	Kosong
4 - B 2304 PJN	PNJ	Tersedia
5 - B 2305 JPN	PNJ	Kosong

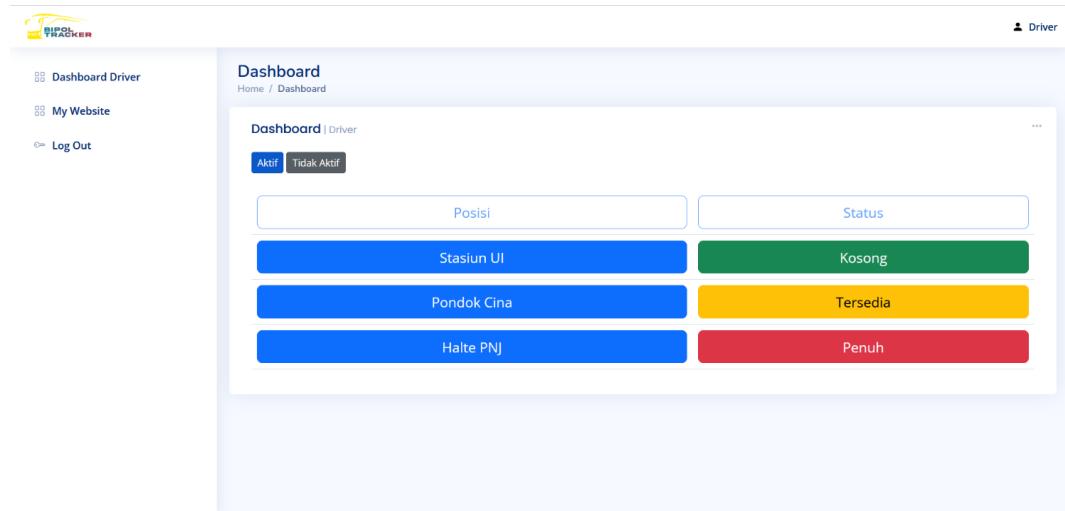
Gambar 3. 23 Tampilan Posisi Bipol

3. 8. 3 Tampilan Login Page

The screenshot shows a login page for the Bipol Tracker application. At the top left is the 'BIPOL TRACKER' logo. In the center is a login form with the title 'LOGIN ADMIN & DRIVER'. The form contains fields for 'Username' and 'Password', both with placeholder text. There is also a 'Remember me' checkbox and a 'Login' button.

Gambar 3. 24 Tampilan Login Page

3. 8. 4 Tampilan Dashboard Driver



Gambar 3. 25 Tampilan Dashboard Driver

3. 8. 5 Tampilan Dashboard Admin Database Jadwal

The screenshot shows the Bipol Tracker dashboard for admin database jadwal. At the top right, it says 'Admin'. On the left sidebar, there are links for 'Database Jadwal', 'Database Posisi', 'Database Bipol', 'Database Driver', 'My Website', and 'Log Out'. The main area is titled 'Database Jadwal' and shows a table of scheduled stops. The columns are 'Bipol', 'Halte', 'Waktu', and 'Aksi'. Each row contains a stop ID, location, time, and edit/delete buttons. There are 10 rows in total.

Bipol	Halte	Waktu	Aksi
1 - B 2301 PNJ	PNJ	07:00	<button>Edit</button> <button>Delete</button>
1 - B 2301 PNJ	Pondok Cina	07:10	<button>Edit</button> <button>Delete</button>
1 - B 2301 PNJ	Stasiun UI	07:20	<button>Edit</button> <button>Delete</button>
1 - B 2301 PNJ	Pondok Cina	07:30	<button>Edit</button> <button>Delete</button>
1 - B 2301 PNJ	PNJ	07:40	<button>Edit</button> <button>Delete</button>
1 - B 2301 PNJ	Pondok Cina	07:50	<button>Edit</button> <button>Delete</button>
1 - B 2301 PNJ	Stasiun UI	08:00	<button>Edit</button> <button>Delete</button>
2 - B 2302 JNP	PNJ	07:05	<button>Edit</button> <button>Delete</button>

Gambar 3. 26 Tampilan Dashboard Admin Database Jadwal

3. 8. 6 Tampilan Dashboard Admin Database Posisi

The screenshot shows the 'Database Posisi' section of the Bipol Tracker admin dashboard. On the left sidebar, 'Database Posisi' is selected. The main area displays a table titled 'Database Posisi' with columns: Bipol, Posisi, Waktu, Kapasitas, and Aksi. The data is as follows:

Bipol	Posisi	Waktu	Kapasitas	Aksi
1 - B 2301 PNJ	Pondok Cina	14:48	Penuh	<button>Edit</button> <button>Delete</button>
2 - B 2302 JNP	Pondok Cina	08:15	Tersedia	<button>Edit</button> <button>Delete</button>
3 - B 2303 NJP	Stasiun UI	09:04	Kosong	<button>Edit</button> <button>Delete</button>
4 - B 2304 PJN	Stasiun UI	14:36	Kosong	<button>Edit</button> <button>Delete</button>
5 - B 2305 JPN	Stasiun UI	14:31	Kosong	<button>Edit</button> <button>Delete</button>

Gambar 3. 27 Tampilan Dashboard Admin Database Posisi

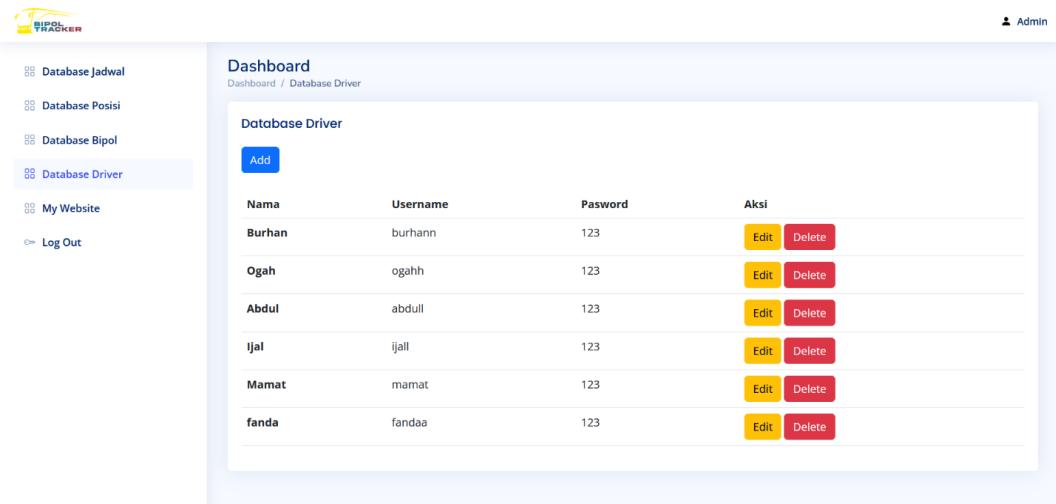
3. 8. 7 Tampilan Dashboard Admin Database Bipol

The screenshot shows the 'Database Bipol' section of the Bipol Tracker admin dashboard. On the left sidebar, 'Database Bipol' is selected. The main area displays a table titled 'Database Bipol' with columns: ID, Plat Nomor, Driver, and Aksi. The data is as follows:

ID	Plat Nomor	Driver	Aksi
1	B 2301 PNJ	2	<button>Edit</button> <button>Delete</button>
2	B 2302 JNP	4	<button>Edit</button> <button>Delete</button>
3	B 2303 NJP	5	<button>Edit</button> <button>Delete</button>
4	B 2304 PJN	3	<button>Edit</button> <button>Delete</button>
5	B 2305 JPN	1	<button>Edit</button> <button>Delete</button>
6	B 2023 TRY	6	<button>Edit</button> <button>Delete</button>

Gambar 3. 28 Tampilan Dashboard Admin Database Bipol

3. 8. 8 Tampilan Dashboard Admin Database Driver



The screenshot shows the Bipol Tracker admin dashboard. On the left is a sidebar with navigation links: Database Jadwal, Database Posisi, Database Bipol, Database Driver (which is highlighted in blue), My Website, and Log Out. The main area is titled "Database Driver" and shows a table with the following data:

Nama	Username	Password	Aksi
Burhan	burhann	123	<button>Edit</button> <button>Delete</button>
Ogah	ogahh	123	<button>Edit</button> <button>Delete</button>
Abdul	abdull	123	<button>Edit</button> <button>Delete</button>
Ijal	ijall	123	<button>Edit</button> <button>Delete</button>
Mamat	mamat	123	<button>Edit</button> <button>Delete</button>
fanda	fandaa	123	<button>Edit</button> <button>Delete</button>

Gambar 3. 29 Tampilan Dashboard Admin Database Driver

3. 10 Program API

API ini akan digunakan untuk CRUD website dengan database bipol tracker. Terdapat 21 *endpoint* yang dapat digunakan.

GET	/ Index	▼
GET	/api/driver/ Driver Read	▼
GET	/api/driver/{id} Driver Readbyid	▼
POST	/api/driver/create Driver Create	▼
PUT	/api/driver/update/{id} Driver Update	▼
DELETE	/api/driver/delete/{id} Driver Delete	▼
GET	/api/bipol/ Bipol Read	▼
GET	/api/bipol/{id} Bipol Readbyid	▼
POST	/api/bipol/create Bipol Create	▼
PUT	/api/bipol/update/{id} Bipol Update	▼
DELETE	/api/bipol/delete/{id} Bipol Delete	▼
GET	/api/jadwal/ Jadwal Read	▼
GET	/api/jadwal/{id} Jadwal Readbyid	▼
POST	/api/jadwal/create Jadwal Create	▼
PUT	/api/jadwal/update/{id} Jadwal Update	▼
DELETE	/api/jadwal/delete/{id} Jadwal Delete	▼
GET	/api/posisi/ Posisi Read	▼
GET	/api/posisi/{id} Posisi Readbyid	▼
POST	/api/posisi/create Posisi Create	▼
PUT	/api/posisi/update/{id} Posisi Update	▼
DELETE	/api/posisi/delete/{id} Posisi Delete	▼

Gambar 3. 30 Program API Bipol Tracker

3. 10. 1 Schema

Merupakan *blueprint* rancangan database bipol tracker. Terdapat empat tabel, yaitu tabel bipol, driver, jadwal, dan posisi. Masing-masing tabel memiliki field yang berbeda-beda.



Gambar 3. 31 Schema API

3. 10. 2 Default (/)

Merupakan *endpoint* yang akan tampil secara default.

The screenshot shows a REST API documentation interface for a '/ Index' endpoint. At the top, there's a 'GET' button and a 'Cancel' button. Below that is a 'Parameters' section with a note 'No parameters'. There are 'Execute' and 'Clear' buttons. The 'Responses' section contains a 'Curl' command, a 'Request URL' (http://localhost:8000/), and a 'Server response' block. The 'Server response' block shows a status code of 200, a JSON response body, and response headers. The response body is:

```
{ "title": "API for Bipol Tracker", "developer": "kelompok 2 - THD 3" }
```

The response headers are:

```
content-length: 66 content-type: application/json date: Wed, 11 Jan 2023 23:57:38 GMT server: unicorn
```

Below the responses, there's a 'Code' dropdown set to 'application/json' (which is highlighted in green) and a 'Description' field containing 'Successful Response'. A 'Media type' dropdown also has 'application/json' selected. On the right side, there are 'Links' and 'No links' sections.

Gambar 3. 32 Default (/)

3. 10. 3 Driver Read (/api/driver/)

Merupakan endpoint yang digunakan untuk membaca semua data driver yang ada di *database*.

The screenshot shows a REST API documentation interface for the `/api/driver/` endpoint. At the top, there is a blue header bar with the method `GET` and the endpoint `/api/driver/`. Below the header, there are sections for **Parameters** (No parameters), **Responses**, and **Code** (200). The **Responses** section contains a **Curl** command and a **Request URL** (HTTP://localhost:8000/api/driver/). The **Server response** section shows a JSON response body with four driver objects. The **Response headers** section lists content-length: 388, content-type: application/json, date: Wed, 12 Jan 2023 23:58:12 GHT, and server: unicorn. The **Code** section also includes a **Description** (Successful Response) and a **Media type** dropdown set to application/json.

```
curl -X 'GET' \
  'http://localhost:8000/api/driver/' \
  -H 'accept: application/json'
Request URL
HTTP://localhost:8000/api/driver/
Server response
Code Details
200 Response body
{
  "results": [
    {
      "id_driver": 1,
      "name": "Burhan",
      "username": "burhan",
      "password": "123"
    },
    {
      "id_driver": 2,
      "name": "Qash",
      "username": "qashah",
      "password": "123"
    },
    {
      "id_driver": 3,
      "name": "Abdu",
      "username": "abdu11",
      "password": "123"
    },
    {
      "id_driver": 4,
      "name": "Izaal",
      "username": "izaal11",
      "password": "123"
    }
  ]
}
Response headers
content-length: 388
content-type: application/json
date: Wed, 12 Jan 2023 23:58:12 GHT
server: unicorn
Responses
Code Description Links
200 Successful Response No links
Media type
application/json
Controls Accept header
Example Value | Schema
"string"
```

Gambar 3. 33 Driver Read (/api/driver/)

3. 10. 4 Driver Read by ID (/api/driver/{id})

Merupakan *endpoint* yang digunakan untuk membaca data driver berdasarkan id driver yang ada di *database*.

The screenshot displays a REST API interface for testing the '/api/driver/{id}' endpoint. At the top, a GET request is shown with the path '/api/driver/{id}' and a parameter 'id' set to '3'. Below the request, the 'Responses' section is expanded, showing a successful 200 response. The response body contains a JSON object with a single result:

```
{ "results": [ { "id_driver": 3, "name": "Abdul", "username": "abdull", "password": "123" } ] }
```

Below the response body, the 'Response headers' section shows the following details:

```
content-length: 81
content-type: application/json
date: Wed, 11 Jan 2023 23:58:43 GMT
server: uvicorn
```

At the bottom of the interface, there are sections for 200 and 422 responses. The 200 response is labeled 'Successful Response' and shows a media type of 'application/json'. The 422 response is labeled 'Validation Error' and shows a media type of 'application/json'.

Gambar 3. 34 Driver Read by ID (/api/driver/{id})

3. 10. 5 Driver Create (/api/driver/create)

Merupakan *endpoint* yang digunakan untuk menambah data driver ke *database*.

The screenshot shows a REST API documentation interface for the `/api/driver/create` endpoint. The top section is a configuration panel with a green "POST" button, the endpoint URL, and a "Driver Create" title. It includes sections for "Parameters" (none), "Request body" (required, type `application/json`), and "Responses". The "Request body" field contains the following JSON example:

```
{
  "id_driver": 6,
  "name": "fendy",
  "username": "fendya",
  "password": "123"
}
```

The "Responses" section has two entries:

- Curl**: A terminal command using curl to make a POST request to `http://localhost:8000/api/driver/create` with the provided JSON body.
- Request URL**: The URL `http://localhost:8000/api/driver/create`.

The "Server response" section shows a successful 200 status with the following details:

- Code**: 200
- Details**: Response body: `{ "message": "sukses" }`, Response headers: `content-length: 20 content-type: application/json date: Wed, 11 Jan 2023 23:59:45 GMT server: unicorn`.

The "Responses" section also lists a validation error response:

- Code**: 422
- Description**: Validation Error
- Media type**: `application/json`
- Example Value**: `"string"`
- Links**: No links

Gambar 3. 35 Driver Create (/api/driver/create)

3. 10. 6 Driver Update (/api/driver/update/{id})

Merupakan endpoint yang digunakan untuk mengubah data driver berdasarkan id driver yang ada di *database*.

The screenshot shows a REST API tool interface for the 'Driver Update' endpoint (`/api/driver/update/{id}`).
Parameters:
Name: `id` (required, integer, path) Value: 6
Request body (required): application/json

```
{
  "id_driver": 6,
  "name": "shfandas",
  "username": "shfandas",
  "password": "123"
}
```

Responses:
Curl:

```
curl -X PUT \
  http://localhost:8000/api/driver/update/6 \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id_driver": 6,
    "name": "shfandas",
    "username": "shfandas",
    "password": "123"
}'
```

Request URL: `http://localhost:8000/api/driver/update/6`

Server response:

Code	Details	Links
200	Response body: ({"message": "sukses"} Response headers: content-length: 30 content-type: application/json date: Thu, 12 Jan 2023 00:00:38 GMT server: unicorn	Download
200	Successful Response Media type: application/json Controls Accept header. Example Value Schema "string"	No links
422	Validation Error Media type: application/json Example Value Schema <pre>{ "detail": [{ "loc": ["msg"], "msg": "string", "type": "string" }] }</pre>	No links

Gambar 3. 36 Driver Update (/api/driver/update/{id})

3. 10. 7 Driver Delete (/api/driver/delete/{id})

Merupakan endpoint yang digunakan untuk menghapus data driver berdasarkan id driver yang ada di *database*.

The screenshot shows a REST API documentation interface for the `DELETE /api/driver/delete/{id}` endpoint. The interface is divided into several sections:

- Parameters:** A table showing a single parameter: `id` (required, integer, path) with value `6`. Buttons for `Execute` and `Clear` are at the bottom.
- Responses:** A section for `Curl` command and `Request URL` (`http://localhost:8000/api/driver/delete/6`). Below it is a `Server response` section for `Code 200`, showing a JSON response body with `{"message": "sukses"}` and response headers including `content-length: 20`, `content-type: application/json`, `date: Thu, 12 Jan 2023 00:00:51 GMT`, and `server: unicorn`. A `Download` button is present.
- Responses:** A section for `Code 200` labeled `Successful Response`, showing a media type dropdown set to `application/json` and an example value of `"string"`.
- Responses:** A section for `Code 422` labeled `Validation Error`, showing a media type dropdown set to `application/json` and a complex JSON schema definition for the error message.

Gambar 3. 37 Driver Delete (/api/driver/delete/{id})

3. 10. 8 Bipol Read (/api/bipol/)

Merupakan *endpoint* yang digunakan untuk membaca semua data bipol yang ada di *database*.

The screenshot shows a REST API documentation interface for the `/api/bipol/` endpoint. At the top, there is a blue header bar with the method `GET` and the endpoint path `/api/bipol/ Bipol Read`. Below the header, there are sections for **Parameters** (No parameters), **Responses**, and **Code** (Details).

Responses section:

- Curl**: A code snippet for running a GET request to `http://localhost:8000/api/bipol/` and accepting JSON.
- Request URL**: `http://localhost:8000/api/bipol/`
- Server response**: A detailed view of the response body, headers, and example value.

Code section (Details):

- 200 Response body**:

```
{ "results": [ { "id_bipol": 1, "plat_nomor": "B 2381 PHN", "id_driver": 2 }, { "id_bipol": 2, "plat_nomor": "B 2382 JNP", "id_driver": 4 }, { "id_bipol": 3, "plat_nomor": "B 2383 KNP", "id_driver": 5 }, { "id_bipol": 4, "plat_nomor": "B 2384 PHN", "id_driver": 3 }, { "id_bipol": 5, "plat_nomor": "B 2385 JPN", "id_driver": 1 } ] }
```
- Response headers**:

```
content-length: 288  
content-type: application/json  
date: Thu, 12 Jan 2023 00:04:07 GHT  
server: unicorn
```
- Responses**:
 - Code**: 200
 - Description**: Successful Response
 - Media type**: application/json
 - Example Value**: string

Gambar 3. 38 Bipol Read (/api/bipol/)

3. 10. 9 Bipol Read by ID (/api/bipol/{id})

Merupakan endpoint yang digunakan untuk membaca data bipol berdasarkan id bipol yang ada di *database*.

The screenshot displays a REST API documentation interface. At the top, a blue header bar indicates a GET request to the endpoint `/api/bipol/{id}` labeled "Bipol Read by id".
Parameters:
- **Name**: `id` (required, integer, path), value: 3
Responses:
- **Curl**: curl -X 'GET' \ http://localhost:8000/api/bipol/3 \ -H 'Accept: application/json'
- **Request URL**: http://localhost:8000/api/bipol/3
- **Server response**:
 - **Code**: 200
 - **Details**:
 - **Response body**:

```
{  
  "results": [  
    {  
      "id_bipol": 3,  
      "plat_nomor": "B 2383 NJP",  
      "id_driver": 5  
    }  
  ]  
}
```


 - **Response headers**:
 content-length: 68
 content-type: application/json
 date: Thu, 13 Jan 2023 00:04:32 GMT
 server: uvicorn

- **Code**: 200
- **Description**: Successful Response
- **Media type**: application/json
- **Example Value**: "string"

- **Code**: 422
- **Description**: Validation Error
- **Media type**: application/json
- **Example Value**:

```
{  
  "detail": [  
    {  
      "loc": [  
        "string"  
      ],  
      "msg": "string",  
      "type": "string"  
    }  
  ]  
}
```

Gambar 3. 39 Bipol Read by ID (/api/bipol/{id})

3. 10. 10 Bipol Create (/api/bipol/create)

Merupakan *endpoint* yang digunakan untuk menambah data bipol ke *database*.

The screenshot displays a REST API interface for creating a bipol entry. At the top, a POST request is shown to the endpoint `/api/bipol/create`. The request body is set to `application/json` and contains the following JSON payload:

```
{
  "id_bipol": 6,
  "plat_nomor": "B 2023 TRV",
  "id_driver": 6
}
```

Below the request, there are sections for `Curl` and `Request URL`:

```
curl -X 'POST' \
  'http://localhost:8000/api/bipol/create' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id_bipol": 6,
    "plat_nomor": "B 2023 TRV",
    "id_driver": 6
}'
```

`Request URL`: <http://localhost:8000/api/bipol/create>

The `Server response` section shows a successful 200 status code. The response body is a JSON object with a single key `message` containing the value `"sukses"`. The response headers include `content-length: 20`, `content-type: application/json`, `date: Mon, 05 Jun 2023 00:05:02 GMT`, and `server: uvicorn`.

The `Responses` section for a 200 status code indicates a `Successful Response` with a media type of `application/json`. The example value is `"string"`.

The `422 Validation Error` section shows a detailed schema for validation errors, indicating that the `msg` field is required and must be a string.

Gambar 3. 40 Bipol Create (/api/bipol/create)

3. 10. 11 Bipol Update (/api/bipol/update/{id})

Merupakan endpoint yang digunakan untuk mengubah data bipol berdasarkan id bipol yang ada di *database*.

The screenshot displays a REST API testing interface. At the top, a PUT request is shown to the endpoint `/api/bipol/update/{id}` with the sub-path `Bipol Update`. The `Parameters` section contains a single parameter `id` (required, integer, path) with the value `6`. The `Request body` section shows a JSON object:

```
{
  "id_bipol": 6,
  "plat_nomor": "# 2023 NH",
  "id_driver": 6
}
```

The `application/json` media type is selected for the request body. Below the request, there are `Execute` and `Clear` buttons. The `Responses` section shows the curl command for the request:

```
curl -X PUT \
  http://localhost:8000/api/bipol/update/6 \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id_bipol": 6,
    "plat_nomor": "# 2023 NH",
    "id_driver": 6
}'
```

The `Request URL` is listed as `http://localhost:8000/api/bipol/update/6`. The `Server response` section details the successful response (200 OK). The response body contains the message `"sukses"`. The response headers include `content-length: 20`, `content-type: application/json`, `date: Thu, 12 Jan 2023 00:07:19 GMT`, and `server: unicorn`. The `Responses` section also lists a `200 Successful Response` with a `Media type: application/json` dropdown set to `application/json`. The `422 Validation Error` section shows a complex JSON schema for validation errors.

Gambar 3. 41 Bipol Update (/api/bipol/update/{id})

3. 10. 12 Bipol Delete (/api/bipol/delete/{id})

Merupakan *endpoint* yang digunakan untuk menghapus data bipol berdasarkan id bipol yang ada di *database*.

The screenshot displays a REST API interface for the Bipol Delete endpoint. At the top, a red header bar indicates a **DELETE** request to `/api/bipol/delete/{id}` with the operation name **Bipol Delete**. Below this is a **Parameters** section with a single entry: **id** (required, integer, path) set to **6**. To the right of the parameters is a **Cancel** button. A large blue **Execute** button is positioned at the bottom of the main input area, with a **Clear** button to its right.

The interface then transitions into a **Responses** section. It starts with a **Curl** example:

```
curl -X 'DELETE' \
  'http://localhost:8000/api/bipol/delete/6' \
  -H 'accept: application/json'
```

Below the curl example is the **Request URL**:

```
http://localhost:8000/api/bipol/delete/6
```

Under the **Server response** heading, there are two rows for **Code** and **Details**. The first row corresponds to a **200** status code, showing the **Response body** as:

```
{ "message": "sukses" }
```

Next is the **Response headers** section:

```
content-length: 20
content-type: application/json
date: Thu, 12 Jan 2023 00:07:31 GMT
server: unicorn
```

Below the 200 response, there is a **Download** link. The interface then continues with another **Responses** section for **Code** and **Description**. The **Code** row for **200** has a **Description** of **Successful Response**, a **Media type** dropdown set to **application/json**, and a note that there are **No links**. The **Example Value** is shown as **"string"**. The **Code** row for **422** has a **Description** of **Validation Error**, a **Media type** dropdown set to **application/json**, and a note that there are **No links**. The **Example Value** is a JSON object:{"detail": [{ "loc": ["msg", "type"], "msg": "string", "type": "string" }]}

Gambar 3. 42 Bipol Delete (/api/bipol/delete/{id})

3. 10. 13 Jadwal Read (/api/jadwal/)

Merupakan *endpoint* yang digunakan untuk membaca semua data jadwal yang ada di *database*.

The screenshot shows a REST API documentation interface for the `/api/jadwal/` endpoint. At the top, there is a blue header bar with the method `GET` and the endpoint path `/api/jadwal/ - Jadwal Read`. Below the header, there are sections for **Parameters** (No parameters), **Responses**, and **Code** (Details).

Responses section:

- Curl**: A command-line example using curl to make a GET request to `http://localhost:8000/api/jadwal/`.
- Request URL**: The URL `http://localhost:8000/api/jadwal/`.
- Server response**: A detailed JSON response body is shown, representing a list of movie schedules. The JSON structure includes arrays of objects for each day of the week (Senin, Selasa, Rabu, Kamis, Jumat, Sabtu, Minggu) with properties like `id_jadwal`, `id_kipas`, `hari`, `jam`, `ukur`, and `halte`.
- Response headers**: Headers include `content-length: 2889`, `content-type: application/json`, `date: Thu, 12 Jan 2023 00:09:34 GHT`, and `server: uvicorn`.

Code section:

- Code**: Shows the status code `200` and the description `Successful Response`.
- Description**: A dropdown menu shows the media type `application/json`, with other options like `text/html` and `application/xml` available.
- Links**: A link labeled `No links` is present.

Gambar 3. 43 Jadwal Read (/api/jadwal/)

3. 10. 14 Jadwal Read by ID (/api/jadwal/{id})

Merupakan endpoint yang digunakan untuk membaca data jadwal berdasarkan id jadwal yang ada di database.

The screenshot displays the API documentation for the 'Jadwal Read by ID' endpoint. At the top, a GET request is shown with the URL `/api/jadwal/{id}`. A parameter `id` is defined as required, an integer, and part of the path, with a value of 3 entered. Below the request, there are sections for 'Responses', 'Request URL', 'Server response', and 'Response body'. The 'Responses' section includes a 200 status code entry with a 'Successful Response' description, a media type of `application/json`, and an example value of `"string"`. It also includes a 422 status code entry with a 'Validation Error' description, a media type of `application/json`, and a detailed error message:

```
{"detail": [{"loc": ["msg"], "msg": "string", "type": "string"}]}
```

.

Gambar 3. 44 Jadwal Read by ID (/api/jadwal/{id})

3. 10. 15 Jadwal Create (/api/jadwal/create)

Merupakan endpoint yang digunakan untuk menambah data jadwal ke database.

The screenshot displays a REST API interface for creating a jadwal entry. The top part shows the request configuration:

- Method:** POST
- Endpoint:** /api/jadwal/create
- Parameters:** No parameters
- Request body (required):** application/json
- Body Content:** A JSON object with fields: id_jadwal (36), id_bipol1 (1), her1 ("Seuin"), her2 ("Selin"), waktu ("17:00"), and halte ("PM3").

The bottom part shows the server's response:

- Code:** 200
- Response body:** An array with one element: { "message": "sukses" }.
- Response headers:** content-length: 28, content-type: application/json, date: Thu, 09 Mar 2023 01:07:24 GMT, server: uvicorn.
- Responses (for 200):** Successful Response (Media type: application/json).
- Code:** 422
- Description:** Validation Error (Media type: application/json).
Example Value | Schema:
{"string"}
The schema is defined as: application/json, controls Accept header.
Example Value: {"string"}
The validation error details are shown in the schema:

```
{  "detail": [    {      "loc": [        "string",        "string"      ],      "msg": "string",      "type": "string"    }  ]}
```

Gambar 3. 45 Jadwal Create (/api/jadwal/create)

3. 10. 16 Jadwal Update (/api/jadwal/update/{id})

Merupakan endpoint yang digunakan untuk mengubah data jadwal berdasarkan id jadwal yang ada di database.

The screenshot shows a REST API testing interface with the following details:

PUT /api/jadwal/update/{id} Jadwal Update

Parameters

Name	Description
id * required	integer (path)
36	

Request body required

```
{ "id_jadwal": 36, "id_bipol": 1, "jam": "06:00", "waktu": "17:00", "halte": "St. UI" }
```

Responses

Curl

```
curl -X 'PUT' \ -d '{ "id_jadwal": 36, "id_bipol": 1, "jam": "06:00", "waktu": "17:00", "halte": "St. UI" }' \ -H 'accept: application/json' \ -H 'Content-Type: application/json'
```

Request URL

<http://localhost:8000/api/jadwal/update/36>

Server response

Code	Details
200	Response body <pre>{ "message": "sukses" }</pre> Response headers <pre>content-length: 20 content-type: application/json date: 12 Jan 2023 01:07:52 GMT server: unicorn</pre>

Responses

Code	Description	Links
200	Successful Response	No links
422	Validation Error	No links

Media type: application/json

Example Value: Schema

```
"string"
```

Media type: application/json

Example Value: Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        "string"
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Gambar 3. 46 Jadwal Update (/api/jadwal/update/{id})

3. 10. 17 Jadwal Delete (/api/jadwal/delete/{id})

Merupakan *endpoint* yang digunakan untuk menghapus data jadwal berdasarkan id jadwal yang ada di *database*.

The screenshot shows a REST API testing tool with the following details:

- DELETE /api/jadwal/delete/{id} Jadwal Delete**
- Parameters**:
 - Name**: id **Description**: required Integer (path) **Value**: 36
- Responses**:
 - Curl**:

```
curl -X 'DELETE' \
  'http://localhost:8000/api/jadwal/delete/36' \
  -H 'accept: application/json'
```
 - Request URL**: <http://localhost:8000/api/jadwal/delete/36>
 - Server response**:
 - Code**: 200 **Details**: Response body:

```
{
  "message": "sukses"
}
```

 Response headers:

```
content-length: 28
content-type: application/json
date: Thu, 12 Jan 2023 01:08:23 GMT
server: unicorn
```
 - Responses**:
 - Code**: 200 **Description**: Successful Response **Media type**: application/json **Example Value**: string
 - Code**: 422 **Description**: Validation Error **Media type**: application/json **Example Value**: Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Gambar 3. 47 Jadwal Delete (/api/jadwal/delete/{id})

3. 10. 18 Posisi Read (/api/posisi/)

Merupakan endpoint yang digunakan untuk membaca semua data posisi bipol yang ada di *database*.

The screenshot shows a REST API tool interface with the following details:

- Method:** GET
- Endpoint:** /api/posisi/ Posisi Read
- Parameters:** No parameters
- Responses:**
 - Curl:** curl -X 'GET' '\http://localhost:8000/api/posisi/' \ -H 'accept: application/json'
 - Request URL:** http://localhost:8000/api/posisi/
 - Server response:**
 - Code:** 200
 - Response body:**

```
{
  "results": [
    {
      "id_bipol": 1,
      "posisi": "Stasiun UI",
      "waktu": "02:00:00",
      "kapasitas": "Teredia"
    },
    {
      "id_bipol": 2,
      "posisi": "Pondok Cina",
      "waktu": "03:15:00",
      "kapasitas": "Teredia"
    },
    {
      "id_bipol": 3,
      "posisi": "Stasiun UI",
      "waktu": "04:00:00",
      "kapasitas": "Kosong"
    },
    {
      "id_bipol": 4,
      "posisi": "PMII",
      "waktu": "05:45:00",
      "kapasitas": "Teredia"
    }
  ]
}
```
 - Response headers:**

```
content-length: 308
content-type: application/json
date: Thu, 12 Jan 2023 01:09:33 GMT
server: unicorn
```
 - Responses:**
 - Code:** 200
 - Description:** Successful Response
 - Media type:** application/json
 - Example Value:** string

Gambar 3. 48 Posisi Read (/api/posisi/)

3. 10. 19 Posisi Read by ID (/api/posisi/{id})

Merupakan *endpoint* yang digunakan untuk membaca data posisi bipol berdasarkan id bipol yang ada di *database*.

The screenshot displays a REST API documentation interface for the 'Posisi Read by ID' endpoint. At the top, a blue bar indicates a GET request to the URL `/api/posisi/{id}`. Below this, the 'Parameters' section shows a single parameter named 'id' with a value of 3. The 'Responses' section is divided into two main parts: 'Code 200' and 'Code 422'.
Code 200: This section shows the curl command, Request URL (`http://localhost:8000/api/posisi/3`), and the Server response. The response body is a JSON object:

```
{
  "results": [
    {
      "id_bipol": 3,
      "posisi": "Stadium UI",
      "waktu": "9:04:00",
      "kapasitas": "Kosong"
    }
  ]
}
```

The response headers are:

```
content-length: 89
content-type: application/json
date: Thu, 12 Jan 2023 01:09:43 GMT
server: uvicorn
```

Code 422: This section shows the validation error example. The media type is set to application/json, and the example value is:

```
"string"
```

Gambar 3. 49 Posisi Read by ID (/api/posisi/{id})

3. 10. 20 Posisi Create (/api/posisi/create)

Merupakan endpoint yang digunakan untuk menambah data posisi bipol ke database.

The screenshot shows the 'Posisi Create' endpoint configuration in a REST API documentation tool. The 'Request body' section contains the following JSON example:

```
{
  "id_bipol": 6,
  "posisi": "PN2",
  "waktu": "17:00",
  "kapasitas": "Tersedia"
}
```

The 'Responses' section shows the server response for a successful 200 status code:

```
Curl
curl -X 'POST' \
  'http://localhost:8000/api/posisi/create' \
  -H 'Content-Type: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id_bipol": 6,
    "posisi": "PN2",
    "waktu": "17:00",
    "kapasitas": "Tersedia"
  }'

Request URL
http://localhost:8000/api/posisi/create
```

And for a validation error 422 status code:

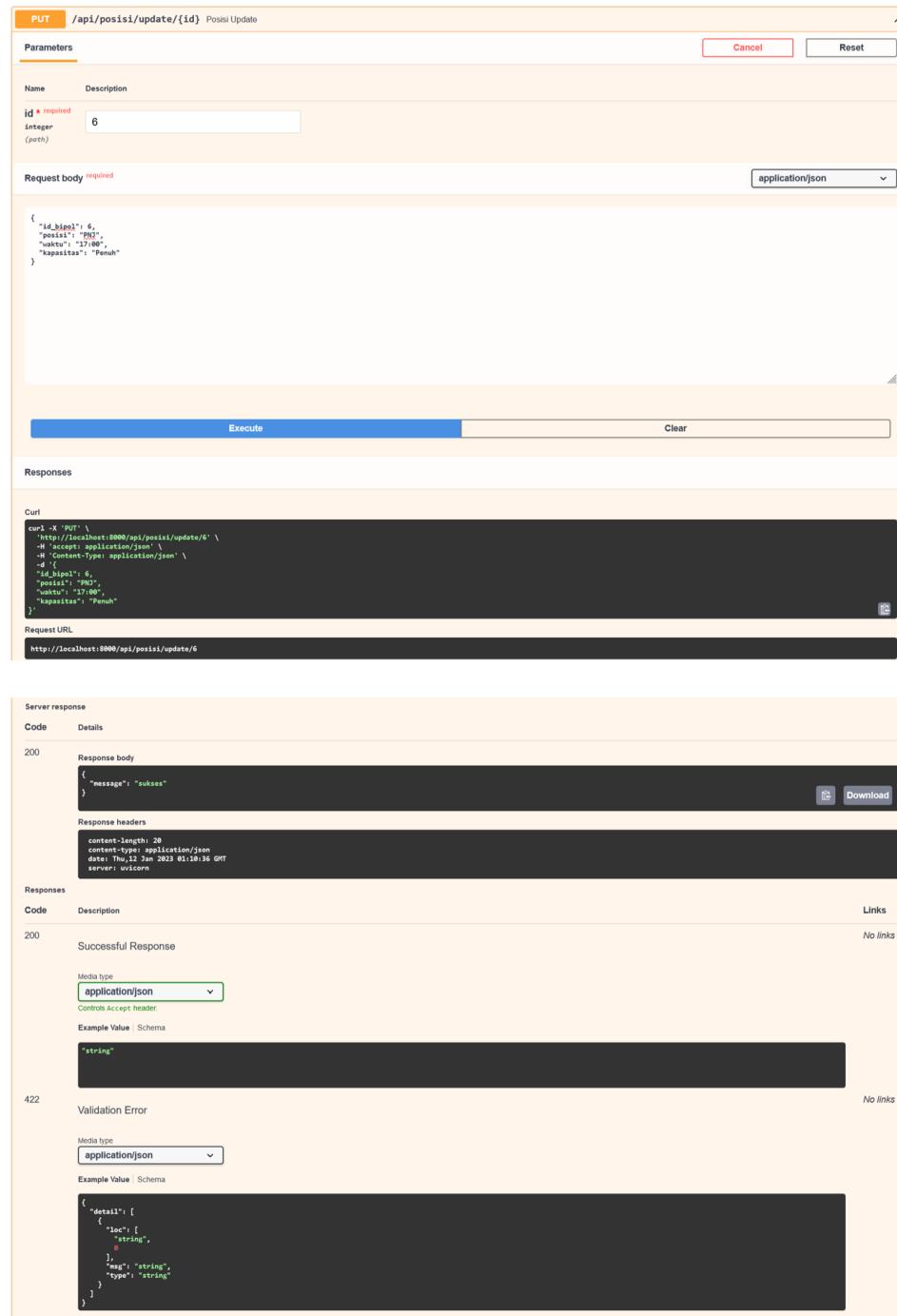
```
Server response
Code Details
200 Response body
{
  "message": "sukses"
}
Response headers
content-length: 28
content-type: application/json
date: Thu, 12 Jan 2023 01:10:16 GMT
server: Unicorn
Responses
Code Description Links
200 Successful Response
Media type
application/json
Controls Accept header.
Example Value | Schema
"string"

422 Validation Error
Media type
application/json
Example Value | Schema
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Gambar 3. 50 Posisi Read by ID (/api/posisi/{id})

3. 10. 21 Posisi Update (/api/posisi/update/{id})

Merupakan endpoint yang digunakan untuk mengubah data posisi bipol berdasarkan id bipol yang ada di database.



The screenshot shows a REST API documentation interface for the `PUT /api/posisi/update/{id}` endpoint. The top section is titled "Parameters" with a single parameter `id` (integer, path) set to 6. The "Request body" field contains the following JSON:

```
{
  "id_bipol": 6,
  "posisi": "P02",
  "waktu": "17:00",
  "kapasitas": "Penuh"
}
```

The "Responses" section includes a "curl" command and a "Request URL".

Server response (200):

```
{
  "message": "sukses"
}
```

Responses (422):

```
{
  "detail": [
    {
      "loc": [
        "string"
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Gambar 3. 51 Posisi Update (/api/posisi/update/{id})

3. 10. 22 Posisi Delete (/api/posisi/delete/{id})

Merupakan *endpoint* yang digunakan untuk menghapus data posisi bipol berdasarkan id bipol yang ada di *database*.

The screenshot shows a REST API testing interface for the `/api/posisi/delete/{id}` endpoint. At the top, there's a red **DELETE** button and the endpoint URL `/api/posisi/delete/{id}`. Below this is a **Parameters** section with a single parameter `id` (required, integer, path) set to `6`. There are **Execute** and **Clear** buttons. The **Responses** section contains two entries:

- Code 200**: Response body:

```
{ "message": "sukses" }
```

 (with **Download** link)
- Code 422**: Response body:

```
{ "detail": [ { "loc": [ "msg", "type" ], "msg": "string", "type": "string" } ] }
```

Each response entry has a **Media type** dropdown set to `application/json`, a **Controls Accept header** link, and a **Example Value** / **Schema** link.

Gambar 3. 52 Posisi Delete (/api/posisi/delete/{id})

3. 11 Program Flask

Flask merupakan suatu framework sekaligus library yang digunakan sebagai alat bantu pengembangan web. Dengan menggunakan library ini pengembang dapat membuat sebuah web yang terstruktur dan juga dapat mengatur behaviour suatu web dengan lebih mudah. Berikut dokumentasi flask python pada website bipol tracker.

3. 11. 1 Directory

File utama dari flask ini bernama app.py yang terletak pada folder HTML > App. Pada folder ini terdiri dari 2 folder dan satu file python app.py. Folder templates berisi file-file html yang akan ditampilkan pada website, sedangkan folder static berisi file-file pendukung seperti css, gambar, javascript, dan asset-asset lainnya.

3. 11. 2 Import library dan inisiasi flask

```
from flask import Flask, render_template, url_for, request, redirect, flash
from datetime import date, datetime
import requests, json

application = Flask(__name__)
```

3. 11. 3 get_day() function

Merupakan fungsi untuk mendapatkan nama hari untuk hari ini

```
today = date.today()
def get_day():
    intDay = today.weekday()
    day = ""
    intDay = 0
    if intDay == 0: day = 'Senin'
    elif intDay == 1: day = 'Selasa'
    elif intDay == 2: day = 'Rabu'
    elif intDay == 3: day = 'Kamis'
    elif intDay == 4: day = 'Jumat'
    elif intDay == 5: day = 'Sabtu'
    elif intDay == 6: day = 'Minggu'
    return day
```

3. 11. 4format_time(waktu) function

Merupakan fungsi untuk merapikan format time

```
def format_time(waktu):
    if waktu[1] == ':':
        return '0' + waktu[0] + ':' + waktu[2] + waktu[3]
    else:
        return waktu[0] + waktu[1] + ':' + waktu[3] + waktu[4]
```

3. 11. 5Rute ‘/’ atau ‘/posisi’ dan index() function

Merupakan *endpoint* dari halaman utama website

```
@application.route('/')
@application.route('/posisi')
def index():
    # send get request
    all_data_posisi_str = (requests.get('http://localhost:8000/api/posisi/')).text
    all_data_posisi = json.loads(all_data_posisi_str)["results"]
    # send get request
    all_data_bipol_str = (requests.get('http://localhost:8000/api/bipol/')).text
    # ambil data dari api
    all_data_bipol = json.loads(all_data_bipol_str)["results"]
    # add plat nomor
    for i in all_data_posisi:
        for j in all_data_bipol:
            if i["id_bipol"] == j["id_bipol"]:
                data = {'plat_nomor':j["plat_nomor"]}
                i.update(data)
    print(all_data_posisi)
    # display posisi page for user
    return render_template(
        'indexPosisi.html',
        data_posisi=all_data_posisi
    )
```

3. 11. 6Rute '/jadwal' dan jadwal() function

Merupakan *endpoint* dari halaman jadwal bipol

```
@application.route('/jadwal')
def jadwal():
    # send get request
    all_data_jadwal_str =
(requests.get('http://localhost:8000/api/jadwal')).text
    all_data_jadwal = json.loads(all_data_jadwal_str)[ "results" ]
    # send get request
    all_data_bipol_str = (requests.get('http://localhost:8000/api/bipol/')).text
    all_data_bipol = json.loads(all_data_bipol_str)[ "results" ]
    # fix format time
    for i in all_data_jadwal:
        i[ "waktu" ] = format_time(i[ "waktu" ])
    # get data jadwal for today
    data_jadwal = []
    for i in all_data_jadwal:
        try:
            if i[ "hari" ] == get_day():
                data_jadwal.append(i)
        except:
            break
    # fix format time
    for i in all_data_jadwal:
        i[ "waktu" ] = format_time(i[ "waktu" ])
    # get format for sort data by time
    strTime = []
    for i in all_data_jadwal:
        strTime.append(i[ "waktu" ])
    strTime.sort()
    strTime = dict.fromkeys(strTime)
    # get data by time
    data_jadwal = []
    for item in strTime:
        for i in all_data_jadwal:
            if i[ "waktu" ] == item:
                data_jadwal.append(i)
    # add plat nomor
    for i in data_jadwal:
```

```

for j in all_data_bipol:
    if i["id_bipol"] == j["id_bipol"]:
        data = {'plat_nomor':j["plat_nomor"]}
        i.update(data)
    # display jadwal page for user
return render_template('index.html', data_jadwal=data_jadwal)

```

3. 11. 7Rute '/driverRead' dan driverRead() function

Merupakan *endpoint* dari halaman dashboard admin yang berisi data driver

```

@application.route('/driverRead')
def driverRead():

    # request data dari API
    response = requests.get("http://localhost:8000/api/driver")

    if response.status_code == 200:
        flash('Berhasil menambahkan data!') # feedback ke user
    else:
        flash('Gagal menambahkan data!')
    return render_template(
        'driverRead.html',
        data=response.json()['results']
    )

```

3. 11. 8Rute '/driverInsert' dan driverInsert()

Merupakan *endpoint* dari halaman tambah driver

```

@application.route('/driverInsert', methods=['GET', 'POST'])
def driverInsert():
    if request.method =='GET':
        return render_template('driverInsert.html')
    if request.method =='POST':
        # request form yang ada di html

```

```

nama = request.form['nama']

username = request.form['username']
password = request.form['password']
# route API untuk create

url = "http://localhost:8000/api/driver/create/"

# ubah data jadi format JSON ({"key" : "value"}, . . .)
data_json=json.dumps({
    "id_driver": 0,
    "nama": nama,
    "username": username,
    "password": password
})

# cek datanya, udah jadi JSON belum
print("data JSON\n",data_json)

# request POST ke API dengan data JSON
response = requests.post(url, data=data_json)
# feedback ke user
if response.status_code == 200:
    flash('Berhasil menambahkan data!')
else:
    flash('Gagal menambahkan data!')
return redirect(url_for('driverRead'))

```

3. 11. 9Rute '/driverEdit/<int:id>' dan driverEdit(id)

Merupakan *endpoint* dari halaman edit driver

```

@application.route('/driverEdit/<int:id>', methods=['GET', 'POST'])
def driverEdit(id):
    if request.method =='GET':
        # request data yang lama dari API

        response = requests.get(
            "http://localhost:8000/api/driver/" + str(id)
        )

```

```

return render_template(
    'driverEdit.html',
    data=response.json()['results']
)
if request.method == 'POST':
    # request form yang ada di html
    nama = request.form['nama']
    username = request.form['username']
    password = request.form['password']

    # route API untuk create
    url = "http://localhost:8000/api/driver/update/" + str(id)
    # ubah data jadi format JSON ({"key": "value"}, . . .)
    data_json=json.dumps({
        "id_driver": id,
        "nama": nama,
        "username": username,
        "password": password
    })
    # cek datanya, udah jadi JSON belum
    print("data JSON\n",data_json)

    # request PUT ke API dengan data JSON
    response = requests.put(url, data=data_json)

    # feedback ke user
    if response.status_code == 200:
        flash('Berhasil mengupdate data!')
    else:
        flash('Gagal mengupdate data!')
return redirect(url_for('driverRead'))

```

3. 11. 10 Rute '/driverDelete/<int:id>' dan driverDelete(id)

Merupakan *endpoint* untuk menghapus data driver

```
@application.route('/driverDelete/<int:id>')
def driverDelete(id):
    # route API untuk delete + id terpilih
    url = "http://localhost:8000/api	driver/delete/" + str(id)
    # request DELETE ke API dengan id terpilih

    response = requests.delete(url)

    # feedback ke user
    if response.status_code == 200:
        flash('Berhasil menghapus data!')
    else:
        flash('Gagal menghapus data!')
    return redirect(url_for('driverRead'))
```

3. 11. 11 Rute '/bipolRead' dan bipolRead()

Merupakan *endpoint* dari halaman dashboard admin yang berisi data bipol

```
@application.route('/bipolRead')
def bipolRead():

    # request data dari API
    response = requests.get("http://localhost:8000/api/bipol")
    # feedback ke user

    if response.status_code == 200:
        flash('Berhasil menambahkan data!')
    else:
        flash('Gagal menambahkan data!')
    return render_template(
        'bipolRead.html',
        data=response.json()['results'])
```

)

3. 11. 12 Rute '/bipolInsert' dan bipolInsert()

Merupakan *endpoint* dari halaman untuk edit bipol

```
@application.route('/bipolInsert', methods=['GET', 'POST'])
def bipolInsert():
    if request.method =='GET':
        return render_template('bipolInsert.html')
    if request.method =='POST':

        # request form yang ada di html
        id = request.form['id']
        platnomor = request.form['platnomor']
        driver = request.form['driver']
        # route API untuk create
        url = "http://localhost:8000/api/bipol/create"
        # ubah data jadi format JSON ({"key" : "value"}, ...)
        data_json=json.dumps({
            "id_bipol": id,
            "plat_nomor": platnomor,
            "id_driver": driver
        })
        # cek datanya, udah jadi JSON belum
        print("data JSON\n",data_json)

        # request POST ke API dengan data JSON
        response = requests.post(url, data=data_json)
        # feedback ke user

        if response.status_code == 200:
            flash('Berhasil menambahkan data!')
        else:
            flash('Gagal menambahkan data!')
    return redirect(url_for('bipolRead'))
```

3. 11. 13 Rute '/bipolEdit/<int:id>' dan bipolEdit(id)

Merupakan *endpoint* dari halaman untuk edit bipol

```
@application.route('/bipolEdit/<int:id>', methods=['GET', 'POST'])
def bipolEdit(id):
    if request.method =='GET':
        # request data yang lama dari API
        response = requests.get(
            "http://localhost:8000/api/bipol/" +str(id)
        )
        return render_template(
            'bipolEdit.html',
            data=response.json()['results']
        )
    if request.method == 'POST':
        # request form yang ada di html
        id = request.form['id']
        platnomor = request.form['platnomor']
        driver = request.form['driver']
        # route API untuk update + id terpilih
        url = "http://localhost:8000/api/bipol/update/" + str(id)
        # ubah data jadi format JSON ({ "key" : "value"}, ...)
        data_json=json.dumps({
            "id_bipol": id,
            "plat_nomor": platnomor,
            "id_driver": driver
        })
        # cek datanya, udah jadi JSON belum
        print("data JSON\n",data_json)

        # request PUT ke API dengan data JSON
        response = requests.put(url, data=data_json)
        # feedback ke user
        if response.status_code == 200:
            flash('Berhasil mengupdate data!')
```

```

else:
    flash('Gagal mengupdate data!')
    return redirect(url_for('bipolRead'))

```

3. 11. 14 Rute '/bipolDelete/<int:id>' dan bipolDelete(id)

Merupakan *endpoint* untuk menghapus data bipol

```

@application.route('/bipolDelete/<int:id>')
def bipolDelete(id):
    # route API untuk delete + id terpilih
    url = "http://localhost:8000/api/bipol/delete/" + str(id)
    # request DELETE ke API dengan id terpilih
    response = requests.delete(url)
    # feedback ke user
    if response.status_code == 200:
        flash('Berhasil menghapus data!')
    else:
        flash('Gagal menghapus data!')
    return redirect(url_for('bipolRead'))

```

3. 11. 15 Rute '/jadwalRead' dan jadwalRead()

Merupakan *endpoint* dari halaman dashboard admin yang berisi data jadwal

```

@application.route('/jadwalRead')
def jadwalRead():
    # send get request
    all_data_jadwal_str = (
        requests.get('http://localhost:8000/api/jadwal/')
        ).text
    all_data_jadwal = json.loads(all_data_jadwal_str)[ "results" ]
    # send get request
    all_data_bipol_str = (
        requests.get('http://localhost:8000/api/bipol/'))

```

```

).text
all_data_bipol = json.loads(all_data_bipol_str)[ "results" ]
# fix format time
for i in all_data_jadwal:
    i["waktu"] = format_time(i["waktu"])
# add plat nomor
for i in all_data_jadwal:
    for j in all_data_bipol:
        if i["id_bipol"] == j["id_bipol"]:
            data = {'plat_nomor':j["plat_nomor"]}
            i.update(data)
# display jadwalRead page
return render_template(
    'jadwalRead.html',
    data_jadwal=all_data_jadwal
)

```

3. 11. 16 Rute '/jadwalInsert' dan jadwalInsert()

Merupakan *endpoint* dari halaman insert jadwal

```

@application.route('/jadwalInsert', methods=['GET', 'POST'])
def jadwalInsert():
    # send get request
    all_data_jadwal_str = (
        requests.get('http://localhost:8000/api/jadwal/')

        ).text
    all_data_jadwal = json.loads(all_data_jadwal_str)[ "results" ]
    # send get request
    all_data_bipol_str = (
        requests.get('http://localhost:8000/api/bipol/')

        ).text
    all_data_bipol = json.loads(all_data_bipol_str)[ "results" ]
    if request.method == 'GET':
        # display jadwalInsert page
        return render_template(

```

```

'jadwalInsert.html',
data_bipol=all_data_bipol

)
else :
    # get value
    bipol = request.form['bipol']
    hari = request.form['hari']
    halte = request.form['halte']
    waktu = request.form['waktu']
    # get new id_jadwal
    for i in all_data_jadwal:
        id_jadwal = i["id_jadwal"] + 1
    # send post request jadwal
    url = "http://localhost:8000/api/jadwal/create"
    payload = json.dumps ({
        "id_jadwal": id_jadwal,
        "id_bipol": bipol,
        "hari": hari,
        "waktu": waktu,
        "halte": halte
    })
    headers = {
        'Content-Type': 'application/json'
    }
    response = requests.request(
        "POST", url, headers=headers, data=payload
    )
print(response.text)
# display jadwalInsert page
return render_template(
    'jadwalInsert.html',
    data_bipol=all_data_bipol
)

```

3. 11. 17 Rute '/jadwalEdit/<int:id>' dan jadwalEdit(id)

Merupakan *endpoint* dari halaman untuk edit jadwal

```
@application.route('/jadwalEdit/<int:id>', methods=['GET', 'POST'])
def jadwalEdit(id):
    # send get request
    data_selected_str = (
        requests.get('http://localhost:8000/api/jadwal/' + str(id)))
    ).text
    data_selected = json.loads(data_selected_str)[ "results" ][0]
    # send get request
    all_data_bipol_str = (
        requests.get('http://localhost:8000/api/bipol/'))
    ).text
    all_data_bipol = json.loads(all_data_bipol_str)[ "results"]
    if request.method == 'GET':
        # convert pad 6 (00:00:00) to 4 (00:00)
        waktu = data_selected[ "waktu"]
        if waktu[1] == ':':
            waktu = '0' + waktu[0] + ':' + waktu[2] + waktu[3]
        else:
            waktu = waktu[0] + waktu[1] + ':' + waktu[3] + waktu[4]
        data_selected[ "waktu"] = waktu
        print(data_selected)
        # get plat nomor
        for i in all_data_bipol:
            if i[ "id_bipol"] == data_selected[ "id_bipol"]:
                data = { 'plat_nomor': i[ "plat_nomor"] }
                data_selected.update(data)
        # display jadwalEdit page
        return render_template(
            'jadwalEdit.html',
            data_selected=data_selected,
            data_bipol=all_data_bipol
```

```

        )
elif request.method == 'POST':
    # get value
    id_bipol = request.form['bipol']
    halte = request.form['halte']
    hari = request.form['hari']
    waktu = request.form['waktu']
    # send put request jadwal
    url = "http://localhost:8000/api/jadwal/update/" + str(id)
    payload = json.dumps ({
        "id_jadwal": id,
        "id_bipol": id_bipol,
        "hari": hari,
        "waktu": waktu+":00",
        "halte": halte
    })
    headers = { 'Content-Type': 'application/json' }
    response = requests.request(
        "PUT", url, headers=headers, data=payload
    )
    print(response.text)
    # display jadwalEdit page
    return redirect('..jadwalEdit/' + str(id))

```

3. 11. 18 Rute '/jadwalDelete/<int:id>' dan jadwalDelete(id)

Merupakan *endpoint* untuk menghapus data jadwal

```

@application.route('/jadwalDelete/<int:id>')
def jadwalDelete(id):
    # send delete request
    url = 'http://localhost:8000/api/jadwal/delete/' + str(id)
    response = requests.request("DELETE", url)
    print(response.text)
    # display jadwal
    return redirect('..jadwalRead')

```

3. 11. 19 Rute '/posisiRead' dan posisiRead()

Merupakan *endpoint* dari halaman dashboard admin yang berisi data posisi

```
@application.route('/posisiRead')
def posisiRead():
    # send get request
    all_data_posisi_str = (
        requests.get('http://localhost:8000/api/posisi/')
    ).text

    all_data_posisi = json.loads(all_data_posisi_str)[ "results" ]
    # send get request
    all_data_bipol_str = (
        requests.get('http://localhost:8000/api/bipol/')
    ).text
    all_data_bipol = json.loads(all_data_bipol_str)[ "results" ]
    # fix format time
    for i in all_data_posisi:
        i[ "waktu" ] = format_time(i[ "waktu" ])
    # add plat nomor
    for i in all_data_posisi:
        for j in all_data_bipol:
            if i[ "id_bipol" ] == j[ "id_bipol" ]:
                data = { 'plat_nomor': j[ "plat_nomor" ] }
                i.update(data)
    # display posisiRead page
    return render_template(
        'posisiRead.html',
        data_posisi=all_data_posisi
    )
```

3. 11. 20 Rute '/posisiInsert' dan posisiInsert()

Merupakan *endpoint* dari halaman untuk insert posisi

```
@application.route('/posisiInsert', methods=['GET', 'POST'])
def posisiInsert():
    # send get request
    all_data_bipol_str = (
        requests.get('http://localhost:8000/api/bipol/')

        ).text
    all_data_bipol = json.loads(all_data_bipol_str)[ "results" ]
    if request.method == 'GET':
        # display posisiInsert page
        return render_template(
            'posisiInsert.html',
            data_bipol=all_data_bipol
        )
    else :
        # get value
        bipol = request.form['bipol']
        posisi = request.form['posisi']
        kapasitas = request.form['kapasitas']
        current_datetime = datetime.now()
        waktu = str(current_datetime.strftime("%H:%M")) + ':00'
        # send post request jadwal
        url = "http://localhost:8000/api/posisi/create"
        payload = json.dumps ({
            "id_bipol": bipol,
            "posisi": posisi,
            "waktu": waktu,
            "kapasitas": kapasitas
        })
        headers = {
            'Content-Type': 'application/json'
        }
        response = requests.request(
            "POST", url, headers=headers, data=payload
```

```

        )
print(response.text)
# display jadwalInsert page
return render_template(
    'posisiInsert.html',
    data_bipol=all_data_bipol
)

```

3. 11. 21 Rute '/posisiEdit/<int:id>' dan posisiEdit(id)

Merupakan *endpoint* dari halaman untuk edit posisi

```

@application.route('/posisiEdit/<int:id>', methods=['GET', 'POST'])
def posisiEdit(id):
    # send get request
    data_selected_str = (
        requests.get('http://localhost:8000/api/posisi/' + str(id))
        ).text
    data_selected = json.loads(data_selected_str)[ "results" ][0]
    # send get request
    all_data_bipol_str = (
        requests.get('http://localhost:8000/api/bipol/')

        ).text
    all_data_bipol = json.loads(all_data_bipol_str)[ "results" ]
    if request.method == 'GET':
        # get plat nomor
        for i in all_data_bipol:
            if i[ "id_bipol" ] == data_selected[ "id_bipol" ]:
                data = { 'plat_nomor': i[ "plat_nomor" ] }
                data_selected.update(data)
        # display posisiEdit page
        return render_template(
            'posisiEdit.html',
            data_selected=data_selected,

```

```

        data_bipol=all_data_bipol

    )
elif request.method == 'POST':
    # get value
    id_bipol = request.form['bipol']
    posisi = request.form['posisi']
    kapasitas = request.form['kapasitas']
    current_datetime = datetime.now()
    waktu = str(current_datetime.strftime("%H:%M")) + ':00'
    # send put request posisi
    url = "http://localhost:8000/api/posisi/update/" + str(id)
    payload = json.dumps ({
        "id_bipol": id_bipol,
        "posisi": posisi,
        "waktu": waktu,
        "kapasitas": kapasitas
    })
    headers = {
        'Content-Type': 'application/json'
    }
    response = requests.request(
        "PUT", url, headers=headers, data=payload
    )
    print(response.text)
    # display posisiEdit page
    return redirect('../posisiEdit/' + str(id))

```

3. 11. 22 Rute '/posisiDelete/<int:id>' dan posisiDelete(id)

Merupakan *endpoint* untuk menghapus data posisi

```

@application.route('/posisiDelete/<int:id>')
def posisiDelete(id):
    # send delete request
    url = 'http://localhost:8000/api/posisi/delete/' + str(id)
    response = requests.request("DELETE", url)
    print(response.text)
    # display posisi

```

```
return redirect('../posisiRead')
```

3. 11. 23 Rute '/login' dan login()

Merupakan *endpoint* dari halaman login masuk admin dan driver

```
user = ""
bipolid = 0

@application.route('/login', methods=['GET', 'POST'])
def login():
    global user
    # send get request
    all_data_driver_str = (
        requests.get('http://localhost:8000/api/driver/')
        ).text
    all_data_driver = json.loads(all_data_driver_str)[ "results" ]
    if request.method == 'GET':
        if user == 'admin':
            return redirect('..//jadwalRead')
        elif user != "":
            return redirect('..//dashboardDriver')
        # display login page
        return render_template('login.html')
    else:
        # get value
        username = request.form['user']
        password = request.form['passwd']
        # authentication admin
        if username == 'admin' and password == '123':
            user = 'admin'
            # display dahsboardAdmin page
            return redirect('..//jadwalRead')
        # authentication driver
        for i in all_data_driver:
            if i["username"] == username:
                if i["password"] == password:
                    user = username
                    # display dahsboardDriver page
                    return redirect('..//dashboardDriver')
```

```
# display login page  
  
    return render_template('login.html')
```

3. 11. 24 Rute '/logout' dan logout()

Merupakan *endpoint* untuk logout keluar dari website

```
@application.route('/logout', methods=['GET', 'POST'])  
def logout():  
    global user, bipolid  
    user = ""  
    bipolid = 0  
    return redirect('/')
```

3. 11. 25 Rute '/dashboardDriver' dan dashboardDriver()

Merupakan *endpoint* dari halaman dashboard driver

```
@application.route('/dashboardDriver', methods=['GET', 'POST'])  
def dashboardDriver():  
    global user, bipolid  
    # send get request  
    all_data_driver_str = (  
  
        requests.get("http://localhost:8000/api/driver/"))  
  
        ).text  
    all_data_driver = json.loads(all_data_driver_str)[ "results" ]  
    # send get request  
    all_data_bipol_str = (  
  
        requests.get("http://localhost:8000/api/bipol/"))  
  
        ).text  
    all_data_bipol = json.loads(all_data_bipol_str)[ "results" ]  
    for i in all_data_driver:  
        if i[ "username" ] == user:  
            driverid = i[ "id_driver" ]  
            for j in all_data_bipol:
```

```

if j['id_driver'] == driverid:
    bipolid = j['id_bipol']
    active1 = 'btn-primary'
    active2 = 'btn-secondary'
    return render_template(
        'dashboardDriver.html',
        id=bipolid,
        active1=active1,
        active2=active2
    )
return redirect('..')

```

3. 11. 26 Rute '/dashboardDriverOff' dan dashboardDriver()

Merupakan endpoint dari halaman dashboard driver yang off/tidak aktif

```

@application.route('/dashboardDriverOff', methods=['GET', 'POST'])
def dashboardDriverOff():
    global bipolid
    id = bipolid
    payload = json.dumps ({
        "id_bipol": id,
        "posisi": '_',
        "waktu": '',
        "kapasitas": '-'
    })
    headers = {
        'Content-Type': 'application/json'
    }
    url = "http://localhost:8000/api/posisi/update/" + str(id)
    response = requests.request(
        "PUT", url, headers=headers, data=payload
    )
    print(response.text)
    active1 = 'btn-secondary'
    active2 = 'btn-primary'

```

```

able = 'disabled'
return render_template(\n
    'dashboardDriver.html',
    id=0,
    active1=active1,
    active2=active2,
    able=able
)

```

3. 11. 27 Rute '</<int:id>/<int:i>/<int:j>' dan fungsi updateDriver(id,i,j)

Merupakan *endpoint* untuk mengupdate posisi driver

```

@app.route('/<int:id>/<int:i>/<int:j>')
def updateDriver(id, i, j):
    data_posisi = (
        requests.get('http://localhost:8000/api/posisi/' + str(id))
        .text
    )
    if id == 0:
        return redirect('..../dashboardDriverOff')
    elif i != 0:
        if i == 1:
            posisi = 'Stasiun UI'
        elif i == 2:
            posisi = 'Pondok Cina'
        else:
            posisi = 'PNJ'
    kapasitas = json.loads(
        data_posisi
    )['results'][0]['kapasitas']
    else:
        if j == 1:
            kapasitas = 'Kosong'
        elif j == 2:

```

```

        kapasitas = 'Tersedia'
else:
    kapasitas = 'Penuh'
    posisi = json.loads(data_posisi)[ "results" ][0][ 'posisi' ]
current_datetime = datetime.now()
waktu = str(current_datetime.strftime("%H:%M")) + ':00'
payload = json.dumps ({
    "id_bipol": id,
    "posisi": posisi,
    "waktu": waktu,
    "kapasitas": kapasitas
})
headers = {
    'Content-Type': 'application/json'
}
url = "http://localhost:8000/api/posisi/update/" + str(id)
response = requests.request(
    "PUT", url, headers=headers, data=payload
)
print(response.text)
return redirect('..../dashboardDriver')

```

3. 11. 28 Run atau looping program

Merupakan run code untuk menjalankan flask web

```

if __name__ == '__main__':
    application.run(debug=True)

```

3.12 Pengujian

3.12.1 Pengujian *Black Box*

Teknik pengujian menggunakan Black Box digunakan untuk menjalankan program yang digunakan user dapat sesuai dengan harapan, hasilnya sebagai berikut:

a. Pengujian Halaman Admin

Tabel 3.3. Black Box halaman login admin

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Memasukan username dan password lalu klik tombol login.	Masuk ke halaman admin dan mengakses dashboard.	Masuk ke halaman admin dan mengakses dashboard.	[✓] Diterima [] Ditolak

Tabel 3.4. Black Box halaman login admin

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol logout.	Proses logout selesai admin kembali ke halaman login.	Proses logout berhasil dilakukan admin, admin kembali ke halaman login.	[✓] Diterima [] Ditolak

Tabel 3.5. Black Box halaman database jadwal

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol	Menampilkan halaman untuk	Admin berhasil menekan tombol tambah dan halaman berubah	[✓] Diterima [] Ditolak

	tambah jadwal.	menambahkan jadwal.	menjadi insert jadwal	
2	Menekan tombol Edit	Menampilkan halaman untuk mengedit data jadwal.	Admin berhasil menekan tombol edit dan halaman berubah menjadi update jadwal.	[✓] Diterima [] Ditolak
3	Menekan tombol hapus jadwal.	Menampilkan pop up untuk menghapus data jadwal.	Admin berhasil menekan tombol hapus lalu berdasarkan data yang dipilih tersebut terhapus.	[✓] Diterima [] Ditolak

Tabel 3.6. Black Box halaman database posisi

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol tambah posisi.	Menampilkan halaman untuk menambahkan posisi.	Admin berhasil menekan tombol tambah dan halaman berubah menjadi insert posisi	[✓] Diterima [] Ditolak
2	Menekan tombol Edit	Menampilkan halaman untuk mengedit data posisi.	Admin berhasil menekan tombol edit dan halaman berubah menjadi update posisi.	[✓] Diterima [] Ditolak

3	Menekan tombol hapus posisi.	Data posisi yang dipilih terhapus dan menampilkan data posisi.	Admin berhasil menekan tombol hapus dan data posisi yang dipilih terhapus dan menampilkan data posisi.	[✓] Diterima [] Ditolak
---	------------------------------	--	--	----------------------------

Tabel 3.7. Black Box halaman database bipol

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol tambah bipol.	Menampilkan halaman untuk menambahkan bipol.	Admin berhasil menekan tombol tambah dan halaman berubah menjadi insert bipol.	[✓] Diterima [] Ditolak
2	Menekan tombol Edit	Menampilkan halaman untuk mengedit data posisi.	Admin berhasil menekan tombol edit dan halaman berubah menjadi update bipol.	[✓] Diterima [] Ditolak
3	Menekan tombol hapus bipol.	Data bipol yang dipilih terhapus dan menampilkan data bipol.	Admin berhasil menekan tombol hapus dan data bipol yang dipilih terhapus dan menampilkan data bipol.	[✓] Diterima [] Ditolak

Tabel 3.8. Black Box halaman database driver

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol tambah driver.	Menampilkan halaman untuk menambahkan driver.	Admin berhasil menekan tombol tambah dan halaman berubah menjadi insert driver.	[✓] Diterima [] Ditolak
2	Menekan tombol Edit	Menampilkan halaman untuk mengedit data driver.	Admin berhasil menekan tombol edit dan halaman berubah menjadi update driver.	[✓] Diterima [] Ditolak
3	Menekan tombol hapus driver.	Data posisi yang dipilih terhapus dan menampilkan data driver.	Admin berhasil menekan tombol hapus dan data driver yang dipilih terhapus dan menampilkan data driver.	[✓] Diterima [] Ditolak

Tabel 3.9. Black Box halaman index utama

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol my website	Menampilkan halaman landing page posisi.	Admin berhasil menekan tombol my website dan halaman berubah menjadi landing page posisi.	[✓] Diterima [] Ditolak

b. Pengujian Halaman Driver

Tabel 3.10. Black Box halaman login driver

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Memasukan username dan password lalu klik tombol login.	Masuk ke halaman driver dan mengakses dashboard.	Masuk ke halaman driver dan mengakses dashboard.	[✓] Diterima [] Ditolak

Tabel 3.11. Black Box halaman login driver

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol logout.	Proses logout selesai driver kembali ke halaman login.	Proses logout berhasil dilakukan driver, driver kembali ke halaman login.	[✓] Diterima [] Ditolak

Tabel 3.12. Black Box halaman dashboard driver

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol tidak aktif.	Menonaktifkan status operasi driver.	Status operasi driver nonaktif.	[✓] Diterima [] Ditolak
2	Menekan tombol aktif	Mengaktifkan status operasi driver.	Status operasi driver aktif.	[✓] Diterima [] Ditolak
3	Memilih posisi bipol.	Driver berhasil menekan tombol posisi dan data posisi bipol terupdate.	Driver berhasil menekan tombol posisi dan data posisi bipol terupdate.	[✓] Diterima [] Ditolak
4	Memilih status kapasitas bipol.	Driver berhasil menekan tombol status dan data status kapasitas bipol terupdate.	Driver berhasil menekan tombol status dan data status kapasitas bipol terupdate.	[✓] Diterima [] Ditolak

a. Pengujian Halaman Landing Page

Tabel 3.13. Black Box halaman landing page

No.	Skenario	Harapan	Hasil Pengujian	Keterangan
1	Menekan tombol jadwal.	Masuk ke halaman jadwal bipol.	Masuk ke halaman jadwal bipol.	[✓] Diterima [] Ditolak
2	Menekan tombol posisi.	Masuk ke halaman posisi bipol.	Masuk ke halaman posisi bipol.	[✓] Diterima [] Ditolak

3. 12. 2 Pengujian *User Acceptance Test* (UAT)

Pengujian ini ditujukan untuk fokus terhadap jalur bisnis *end to end* tidak terpaku pada pencarian kesalahan didalam sistem seperti *bug*.

Tabel 3.14. UAT Testing

No.	Pertanyaan	Kode Pertanyaan
1	Apakah fungsi pada halaman utama berfungsi dengan baik?	P1
2.	Apakah fungsi login pada halaman login berfungsi dengan baik?	P2
3	Apakah fungsi logout pada halaman login berfungsi dengan baik?	P3
4	Apakah fungsi tambah jadwal pada halaman tambah jadwal berfungsi dengan baik?	P4
5	Apakah fungsi edit jadwal pada halaman edit jadwal berfungsi dengan baik?	P5

6	Apakah fungsi hapus jadwal pada halaman dashboard jadwal berfungsi dengan baik?	P6
7	Apakah fungsi tambah posisi pada halaman tambah posisi berfungsi dengan baik?	P7
8	Apakah fungsi edit posisi pada halaman edit posisi berfungsi dengan baik?	P8
9	Apakah fungsi hapus posisi pada halaman dashboard posisi berfungsi dengan baik?	P9
10	Apakah fungsi tambah bipol pada halaman tambah kegiatan berfungsi dengan baik?	P10
11	Apakah fungsi edit bipol pada halaman edit bipol berfungsi dengan baik?	P11
12	Apakah fungsi hapus bipol pada halaman dashboard bipol berfungsi dengan baik?	P12
13	Apakah fungsi tambah driver pada halaman tambah driver berfungsi dengan baik?	P13
14	Apakah fungsi edit driver pada halaman edit driver berfungsi dengan baik?	P14
15	Apakah fungsi hapus driver pada halaman dashboard driver berfungsi dengan baik?	P15
16	Apakah fungsi pada landing page jadwal berfungsi dengan baik?	P16
17	Apakah fungsi pada landing page posisi berfungsi dengan baik?	P17

18	Apakah fungsi tombol aktif status operasi driver berfungsi dengan baik?	P18
19	Apakah fungsi tombol pilih posisi berfungsi dengan baik?	P19
20	Apakah fungsi tombol pilih status bipol berfungsi dengan baik?	P20

BAB V

PENUTUP

5.1. Kesimpulan

Bis Politeknik atau Bipol adalah transportasi yang sering digunakan oleh mahasiswa PNJ karena memudahkan untuk berangkat dan pulang kuliah. Namun, yang menjadi permasalahan adalah mahasiswa PNJ tidak mengetahui informasi yang pasti mengenai jadwal dan posisi bipol sehingga membuat mahasiswa PNJ menunggu kedatangan bipol tanpa adanya kepastian. Oleh karena itu, kami menghadirkan sebuah solusi yang dapat menjawab permasalahan-permasalahan tersebut, yaitu **Bipol Tracker**. Bipol Tracker adalah sebuah aplikasi berbasis website untuk memantau jadwal dan posisi Bis Politeknik (bipol) yang digunakan oleh Mahasiswa PNJ dalam transportasi di lingkungan kampus.

5.2. Saran

Saran yang hendak kami sampaikan dari pembuatan website tersebut adalah sebagai berikut:

1. Mengembangkan website Bipol Tracker dari sisi Front-End, Back-End, dan Security.
2. Menambahkan fitur real-time tracking menggunakan perangkat IoT.
3. Mengimplementasikan Bipol Tracker pada bis kuning (bikun).

DAFTAR PUSTAKA

- https://repository.dinamika.ac.id/id/eprint/2233/5/BAB_III.pdf
https://elib.unikom.ac.id/files/disk1/491/jbptunikompp-gdl-indrawardh-24507-3-unikom_i-i.pdf
http://repository.uin-suska.ac.id/16958/7/7.%20BAB%20II_2018315SIF.pdf
https://elibrary.unikom.ac.id/id/eprint/5598/9/UNIKOM_Andryco%20Dwi%20Susanto%20Hutapea_BAB%202.pdf
<http://e-journal.uajy.ac.id/6353/4/TF306026.pdf>
<https://eprints.umm.ac.id/36144/3/jiptummpp-gdl-bagusprase-50390-3-2.babii.pdf>
<https://webdev-id.com/berita/apa-itu-github/>
<http://eprints.mercubuana-yogya.ac.id/id/eprint/2513/9/BAB%20II.pdf>
https://repository.bsi.ac.id/index.php/unduh/item/67418/File_10-Bab-II-Landasan-Teori.pdf
https://elibrary.unikom.ac.id/id/eprint/5283/8/UNIKOM_Muhammad%20Al%20Faruqi_Bab%20II.pdf
<http://e-journal.uajy.ac.id/9846/4/3TF07125.pdf>
<https://dspace.uji.ac.id/bitstream/handle/123456789/15755/05.2%20bab%202.pdf?sequence=6&isAllowed=y>