PROJECT PROPOSAL

UIHelp

Software Engineering Class Group 7

Introduction

The University of Indonesia (UI) faces various safety challenges due to its large campus, which includes forested areas and extensive facilities. These challenges include traffic accidents, wild animals, fallen trees, fires, and floods. To address these issues, UI is committed to both environmental sustainability and safety through initiatives like UI GreenMetric.

To enhance campus safety and support sustainable practices, a disaster reporting application is being developed. This app will enable the UI community to quickly report incidents to Campus Security (PLK), ensuring a faster response to emergencies. The app also aligns with sustainability goals by reducing unnecessary travel and enabling efficient resource management.

Inspired by PetaBencana.id, this application simplifies the reporting process and directs reports to PLK for immediate handling, ensuring quicker responses and more coordinated disaster management.

I. Project Background

The University of Indonesia (UI) is one of the largest and most expansive campuses in Indonesia, covering areas that include forests and various educational and social facilities. This unique environment makes UI vulnerable to numerous incidents that can threaten the safety of the campus community, such as traffic accidents, fallen trees, encounters with wild animals, fires, and floods. The presence of forests and green spaces around the campus adds challenges in terms of monitoring and managing emergency situations.

At the same time, efforts to maintain environmental sustainability and ensure the safety of the campus population are top priorities for UI, as reflected in its commitment to the UI GreenMetric initiative. This program emphasizes the importance of managing campus resources, energy, and safety in a sustainable manner. However, these efforts require an effective system to detect and respond to incidents on the ground quickly and accurately.

One relevant solution to address these challenges is the development of a campus disaster reporting application. This application is expected to act as a bridge between UI's community and Campus Security (PLK) to facilitate reporting incidents such as accidents, fires, and natural disasters. With the app in place, reporting can be done quickly, emergency response becomes more efficient, and disaster management can be carried out in a more coordinated manner.

Moreover, the app supports campus sustainability initiatives by reducing the time and resources needed for reporting and handling incidents. Through the use of this technology, UI can manage disaster risks more effectively, while simultaneously maintaining campus sustainability and safety at an optimal level.

Meet the Team

Raja Yonandro Ruslito

Backend Dev & Dev-ops

Muhammad F. Haritsah

Frontend Dev & Design

Muhamad Fauzan

Project Manager & QA

Project Aim

We are working to develop an application that will assist the community within the University of Indonesia. Given the vast UI campus, which includes forested areas, it is prone to various types of incidents. This application is designed to connect the UI community with Campus Security (PLK) to enable quick responses to incidents such as traffic accidents, encounters with wild animals, fallen trees, fires, and floods.

The application aims to make life easier for UI residents by providing a fast and reliable way to report accidents or emergency situations. Through this app, UI residents can directly contact PLK, allowing for quicker responses to incidents like accidents, wild animals, fallen trees, fires, or floods.

Literature Review

Web-based emergency management applications are increasingly used in university environments to improve campus security and incident response. According to Lee et al. (2018), web-based platforms provide real-time communication between users and security personnel, facilitating faster incident reporting and response. One example is the Guardian by Rave system, which has been successfully implemented in several universities across the United States. This system enables users to report incidents via web platforms or mobile devices, receive alerts, and access emergency services instantly, significantly reducing response times.

Zhang et al. (2019) emphasize the importance of integrating web technologies with location-based services (LBS) and real-time notifications for better emergency coordination. Web applications allow for the centralization of information, providing campus security teams with a detailed view of ongoing incidents, enabling quicker and more targeted responses. This web-based approach enhances accessibility as it allows users to submit reports from any device with internet access, contributing to the efficiency of campus safety management.

Moreover, web-based disaster reporting applications align with sustainability goals by reducing the need for physical interventions. By utilizing real-time updates and remote monitoring, these systems minimize unnecessary energy consumption and resource use, supporting initiatives like UI GreenMetric. As Zhang et al. (2019) point out, web-based solutions can effectively balance environmental sustainability with campus safety, ensuring timely and accurate responses to emergencies while reducing the campus's ecological footprint.

Similar Product

This application was initially inspired by PetaBencana.id, a disaster reporting app that updates in real-time. The app utilizes social media as one of the reporting platforms, in addition to direct reporting within the app itself. It offers a comprehensive range of features, including reporting, legends, notifications, and a real-time map display with pins marking areas where disasters are occurring. One of the most interesting aspects of this app is that users can report incidents through popular social media platforms like Twitter and Facebook, making the reporting process more convenient as users don't need to open the app to submit a report.

Function / Module

To provide the most complete services for the community within UI, this application offers the following features:

1. **Disaster Report Form**: This form contains detailed information about the reporter and the disaster, including the reporter's name, student ID (for UI students) or national ID number (for non-students), disaster location, photo evidence of the disaster, and a description of the incident.
2. **UI Campus Map**: A real-time map equipped with pins marking the locations of ongoing or managed disasters within the campus.
3. **Notifications**: Alerts that provide updates on disasters, delivering real-time information to all users about the disaster and the current situation.

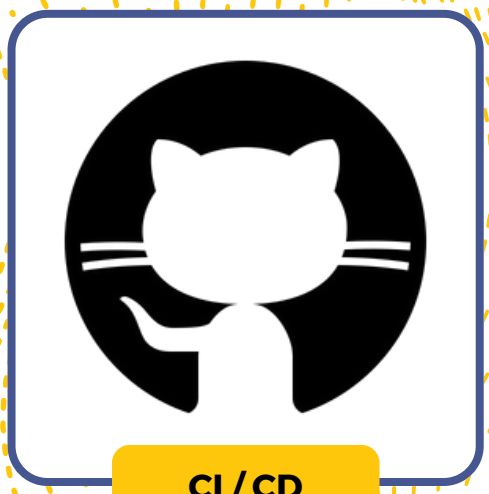
Risk Analysis

Potential Risk	Responsibility	Likelihood	Potential Impact	Contingency
Data Security and Privacy Breaches	Project Manager, Back-End Developer	High	Unauthorized access, user trust loss	Implement robust encryption, conduct regular security audits, comply with data protection regulations for incident reporting.
Inaccurate or Misleading Incident Reports	QA, Front-End Developer	Moderate	Delayed or inefficient emergency response, damage to credibility	Implement strict validation for reports, clear user instructions, and real-time GPS for accurate location tracking.
Technical Issues (App Downtime, Bugs)	DevOps, Back-End Developer	Moderate	Poor user experience, delayed response to emergencies	Implement thorough testing, automated monitoring, rapid bug fixes, and regular maintenance to ensure high uptime and reliability.
Competition from Other Reporting Channels	Project Manager, Back-End Developer	High	Users may prefer existing manual or phone-based systems	Highlight app advantages such as real-time updates, GPS tracking, and faster response times. Regularly update features to meet user needs.
Legal and Ethical Concerns (UI Policies)	Project Manager	Moderate	Violation of university regulations, legal disputes	Collaborate with legal experts to ensure compliance with UI's policies and data handling guidelines.
Limited App Functionality in Emergencies	Project Manager, Front-End Developer	High	Inability to report incidents during network outages or low internet connectivity	Incorporate offline reporting functionality, backup communication methods like SMS integration, and local server caching.

Tool Used

Frontend

Backend



CI / CD

Database

References

[1] YouTube, https://www.youtube.com/watch?v=B6tXP4wBoiI (accessed Sep. 23, 2024).
[2] YouTube, https://www.youtube.com/watch?v=9ivFYYNLT24 (accessed Sep. 23, 2024).
[3] PetaBencana.id, https://petabencana.id/ (accessed Sep. 23, 2024).

II. Project Management

System Review

The UI Help application is designed to enhance the safety and responsiveness of the University of Indonesia (UI) campus community through the following key features:

- User Access

All users has the right to post a report and view any disaster without the need of logging in. Reports made will immediately be forwarded to the PLK for further actions.

- Disaster Reporting

The core function of UI Help is its disaster reporting feature, allowing users to quickly report incidents such as traffic accidents, wild animal encounters, fallen trees, fires, and floods. Users provide details like location, incident description, and photo evidence, enabling Campus Security (PLK) to respond swiftly.

- Real-Time Map and Notifications

UI Help includes a real-time campus map with pins indicating reported incidents, allowing users to see ongoing and managed situations across the UI campus. Additionally, notifications are sent out to keep users updated on the status of incidents, promoting awareness and safety.

II. Project Management

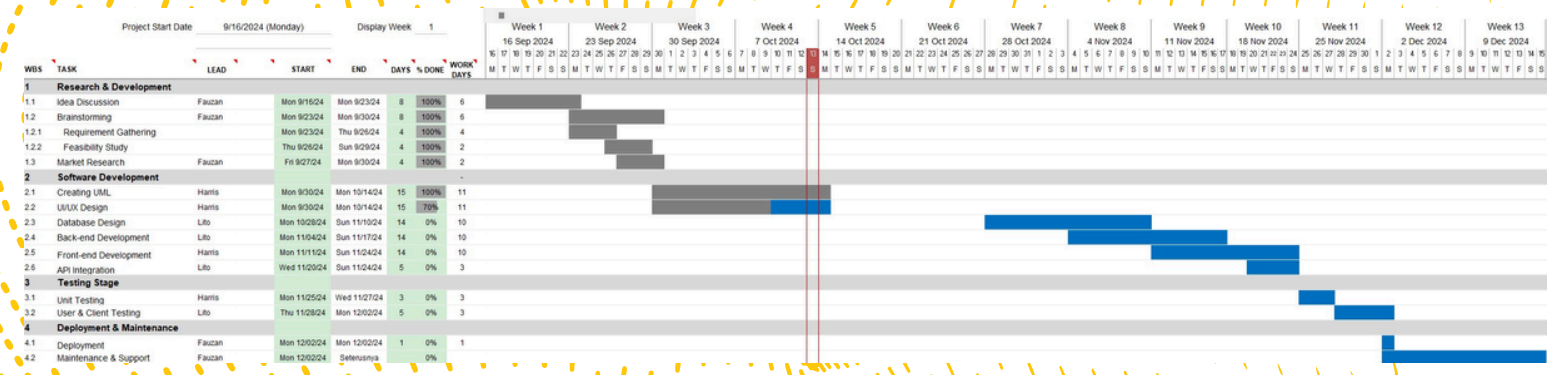
System Review

- Direct PLK Communication

The app allows for direct communication with Campus Security (PLK), ensuring that reports are handled professionally and quickly. This direct connection is designed to reduce response times and improve the coordination of emergency responses on campus.

UI Help aims to be an effective tool for creating a safer and more sustainable campus environment, allowing the UI community to act promptly during emergencies while supporting the university's broader goals.

Timeline



II. Project Management

Scope Planning

Scope Planning involves developing a detailed project scope statement that outlines the deliverables, objectives, and requirements of the project. It also includes creating a Work Breakdown Structure (WBS) that divides the project into smaller, manageable parts. For UI Help, the key activities in Scope Planning include:

- Defining the project scope and objectives, which include improving emergency response times, enhancing campus safety, and supporting UI's sustainability goals.
- Identifying stakeholders, such as students, staff, external visitors, and Campus Security (PLK).
- Creating a project scope statement that outlines the features and modules of the application, such as Disaster Report Form, Real-Time Map, Notifications, and Direct Communication with PLK.
- Developing a WBS that breaks down the project into components like System Review, Disaster Reporting Module, Map Integration, and Notification System.

II. Project Management

Scope Definition

Scope Definition involves further refining the project scope by identifying specific tasks, deliverables, and requirements. It also includes creating a scope baseline as a reference for measuring project performance. For UI Help, the key activities in Scope Definition include:

- Reviewing the project scope statement to ensure alignment with the project objectives.
- Identifying specific project deliverables, such as the Disaster Report Form module, Real-Time Map module, and Notification System module.
- Identifying project requirements, such as the need for real-time location tracking, data accuracy, and secure communication channels with PLK.
- Creating a scope baseline that serves as a reference for monitoring project progress and performance.

II. Project Management

Scope Verification

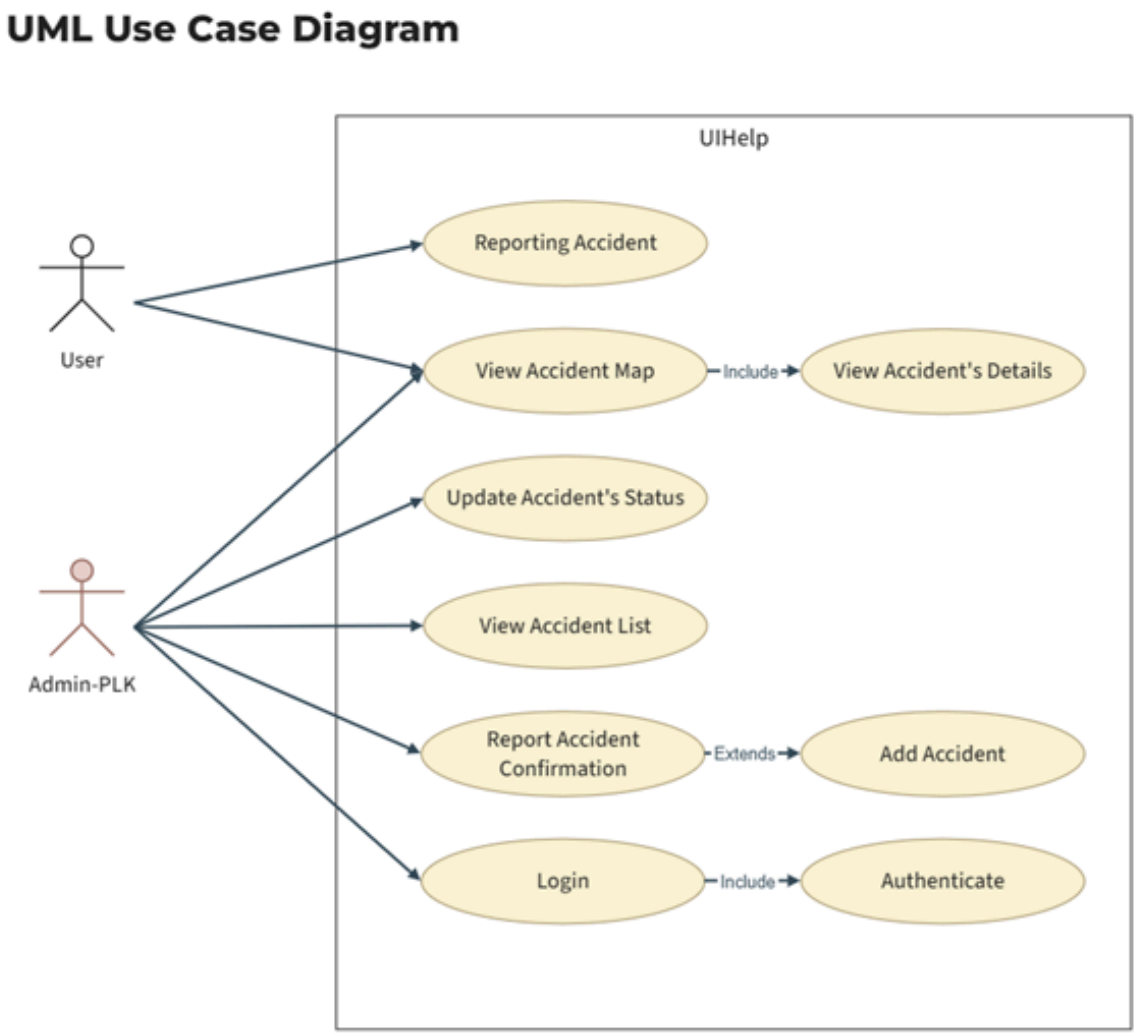
Scope Verification involves reviewing the project deliverables to ensure they meet the requirements outlined in the project scope statement. It also includes obtaining formal acceptance of the project deliverables from stakeholders. For UI Help, the key activities in Scope Verification include:

- Reviewing deliverables, such as the Disaster Report Form module, Real-Time Map module, and Notification System module.
- Comparing the deliverables to the project scope statement to ensure alignment with project objectives.
- Obtaining formal acceptance of the deliverables from stakeholders, including students, staff, and Campus Security (PLK).

III. Software Design

Use Case Diagram

The use case diagram here will provide an explanation of the relation between the user and admin

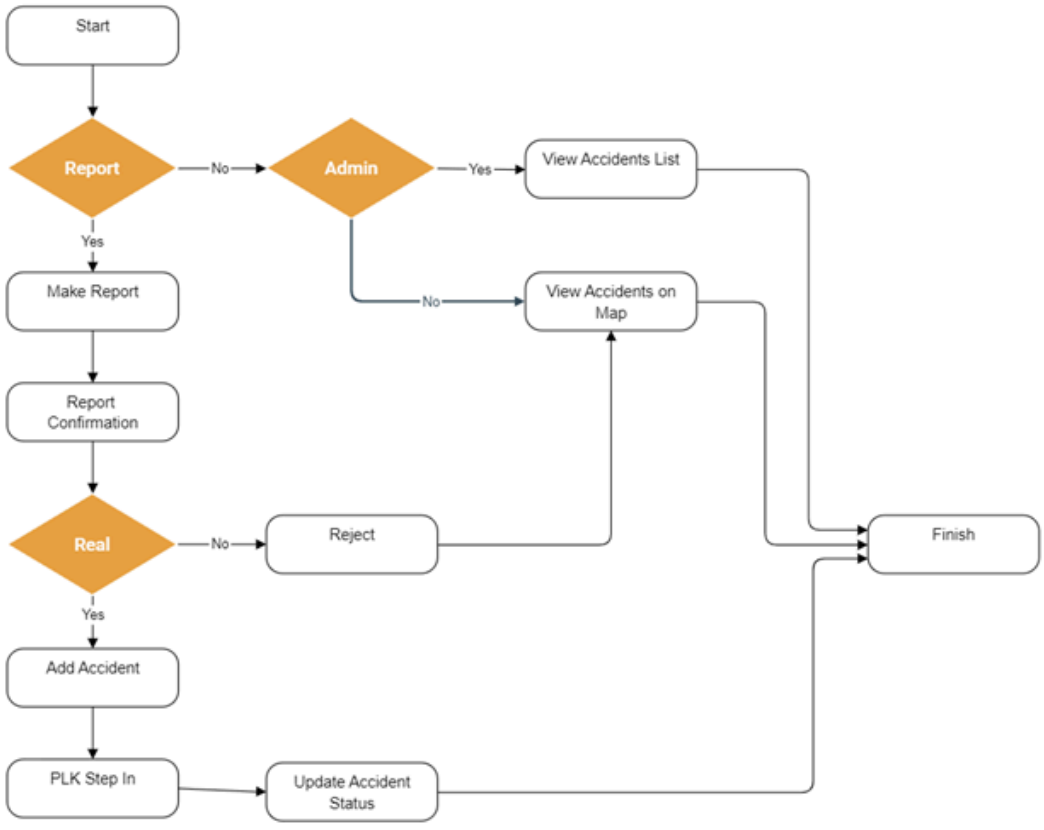


III. Software Design

Activity Diagram

This section focuses on the activity diagram, a visual representation of the workflow or processes within the system, displaying how activities and actions are interconnected

Activity Diagram

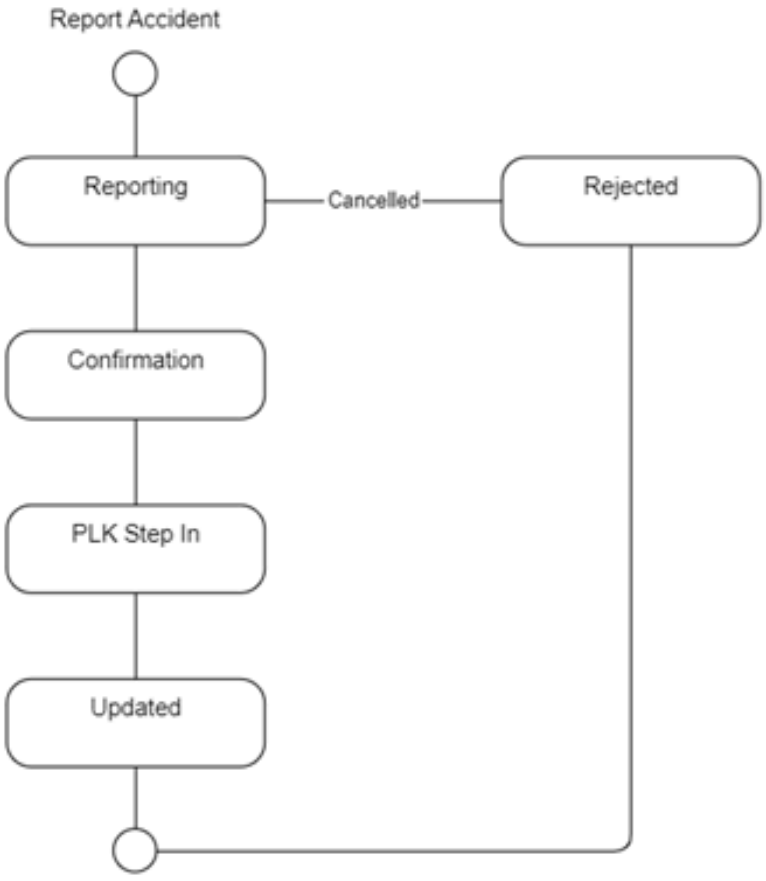


III. Software Design

State Diagram

The state diagram will explain all the different states of the application based on user interaction

State Diagram

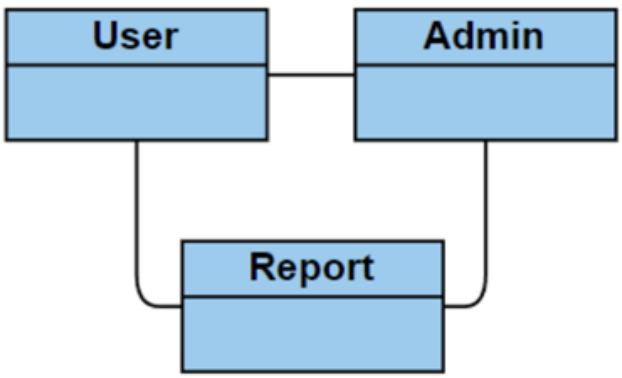


III. Software Design

Class Diagram

In class diagram, there is the class and the parameters that will be used in the application. There is also the relation between the class.

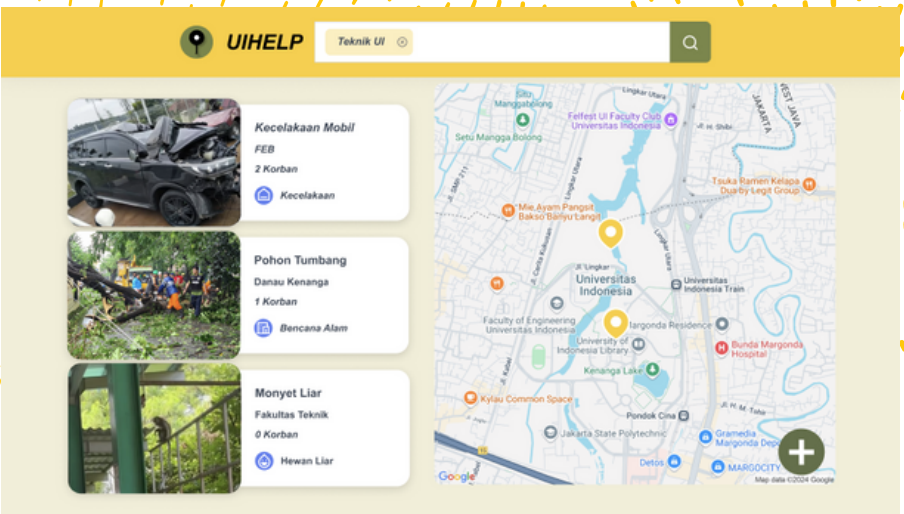
Class Diagram

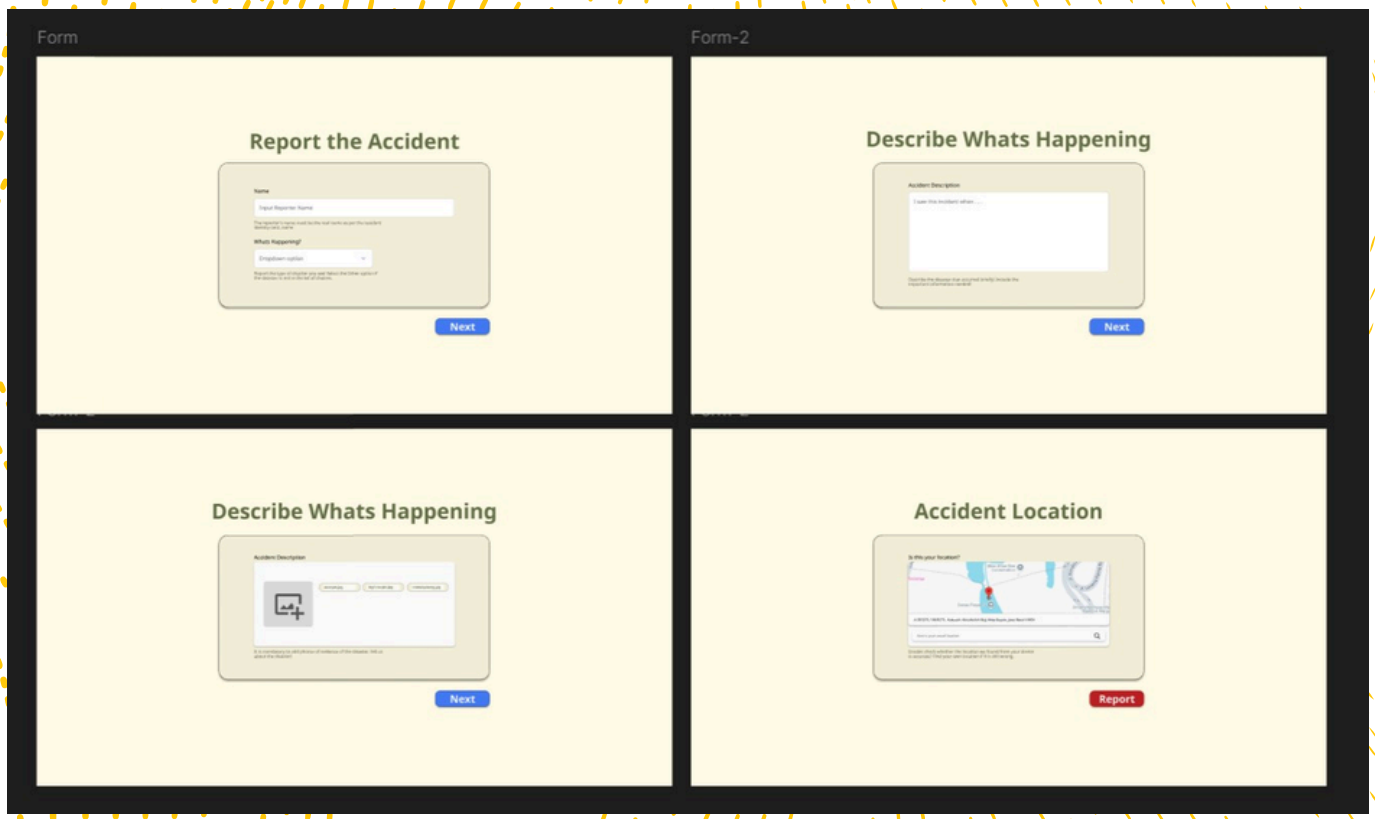


III. Software Design

Mock Up UI

In the mock-ups, there is Homepage and Report Page. These offer a visual preview of the system's user interface





IV. Implementation

Frontend Implementation

The frontend of UIHelp was developed using ReactJS to provide a user-friendly and responsive interface. The design process followed an iterative approach, focusing on functionality, usability, and compatibility across multiple devices.

Key Features

1. **Accident Reporting Form:**
   - Users can upload accident photos, fill in descriptions, and include their current location.
   - Validation ensures the file format (e.g., JPEG/PNG) and file size adhere to specific limits.

```
const handleSubmit = async () => {
  const reportData = {
    name: name,
    types: selectedOption === "Other" ? otherText : selectedOption,
    detail: description,
    picture: photo,
    location: userLocation.latitude + "," + userLocation.longitude,
  };

  console.log("Report Data: ", reportData);
  try {
    const response = await axios.post("http://localhost:5000/report/create", reportData);
    console.log("Response:", response.data);

    // Berikan notifikasi berhasil (opsional)
    alert("Report submitted successfully!");

    // Reset form setelah pengiriman
    setName("");
    setSelectedOption("Select an option");
    setOtherText("");
    setDescription("");
    setPhoto(null); // Reset foto

    navigate("/")
  } catch (error) {
    console.error("Error submitting report:", error);
    alert("Failed to submit report. Please try again.");

  }
};
```

IV. Implementation

Frontend Implementation

2. **Map View:**
   - Integrated with the leaflet API to display accident locations based on user coordinates.
   - A refresh button allows users to update their location if the data is inaccurate.

```
useEffect(() => {
  const fetchLocation = () => {
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(
        async (position) => {
          const { latitude, longitude } = position.coords;
          setUserLocation({ latitude, longitude });

          // Reverse geocoding untuk mendapatkan alamat
          try {
            const response = await axios.get(
              `https://nominatim.openstreetmap.org/reverse?format=json&lat=${latitude}&lon=${longitude}&addressdetails=1`
            );
            setAddress(response.data.address);
          } catch (err) {
            console.error("Error fetching address:", err);
            setError("Gagal mendapatkan detail lokasi.");
          }
        },
        (err) => {
          setError("Gagal mendapatkan lokasi. Pastikan izin lokasi aktif.");
          console.error(err);
        },
        {
          enableHighAccuracy: true,
          timeout: 5000,
          maximumAge: 0,
        }
      );
    } else {
      setError("Browser Anda tidak mendukung Geolocation API.");
    }
  };
```

IV. Implementation

Frontend Implementation

3. **User Dashboard:**
   - Displays recent accident reports to alert other users.
   - Reports are labeled based on their status (Pending, In Progress, Handled).

```
useEffect(() => {
  // Fetch data from the API when the component is mounted
  const fetchAccidents = async () => {
    try {
      const response = await axios.get("http://localhost:5000/report/");
      const data = response.data;
      const formattedData = data.map((item) => ({
        name: item.types,
        desc: item.detail,
        dateTime: dayjs(item.created_at).format("DD:MM:YYYY HH:mm"),
        status: item.status,
        lat: parseFloat(item.location.split(",")[0]), // Convert lat from string to float
        lng: parseFloat(item.location.split(",")[1]), // Convert lng from string to float
        img: item.picture, // Placeholder image (update as needed)
      }));
      setHappeningAccidents(formattedData);
    } catch (error) {
      console.error("Error fetching data:", error);
    }
  };

  fetchAccidents();
}, []);
```

IV. Implementation

Frontend Implementation

4. **Admin Dashboard:**
   - A dedicated interface for admins to review, verify, and manage user-submitted reports.
   - Key functionalities include:
     - **Report Management:** View a list of all submitted reports with filters for status (Pending, Verified).
     - **Photo and Details Review:** Inspect the uploaded images and accompanying details of each report.
     - **Verification Actions:** Update the status of reports (e.g., mark as Verified or Rejected).
     - **Notifications Management:** Trigger notifications to alert users of verified reports or resolved incidents.
   - The dashboard includes visual elements such as graphs and charts for statistical summaries, providing insights into incident types and trends over time.

```
useEffect(() => {
    const fetchAccidents = async () => {
        try {
            const response = await axios.get('http://localhost:5000/report/');
            const data = response.data;

            const formattedData = data.map((item) => ({
                id: item.id,
                name: item.types,
                desc: item.detail,
                dateTime: dayjs(item.created_at).toISOString(),
                status: item.status,
                lat: parseFloat(item.location.split(',')[0]),
                lng: parseFloat(item.location.split(',')[1]),
                img: item.picture || DEFAULT_IMAGE,
            }));

            setHappeningAccidents(formattedData);
        } catch (error) {
            console.error('Error fetching data:', error);
        }
    };

    fetchAccidents();
}, [isModalOpen]);
```

IV. Implementation

Frontend Implementation

Implementation Process

- **Axios** was employed to communicate with the backend via REST API.
- Testing was conducted using the **React Testing Library** to ensure the interface functions as intended.

IV. Implementation

Backend Implementation

The backend of UIHelp was developed using Node.js with Express.js as the primary framework. The backend handles business logic, data validation, and communication with the database.

Key Features

1. **Reporting Endpoint:**
   - Accepts user reports, validates data (description, photo, location), and stores it in the database.

```
async function createReport(req, res) {
    const { name, types, detail, picture, location } = req.body; // Ambil data dari req.body
    let fileUrl = null;

    try {
        // Validasi input
        if (!name || !detail || !types) {
            return res.status(400).json({
                success: false,
                message: 'Name, detail, and types are required.',
            });
        }

        // Jika gambar dalam format base64 diterima, unggah ke Cloudinary
        if (picture) {
            const result = await uploadToCloudinary(picture); // Mengirim gambar base64 ke Cloudinary
            fileUrl = result.secure_url; // Mendapatkan URL aman dari respon Cloudinary
        }

        // Menyimpan data laporan ke database
        const query = `
            INSERT INTO report (name, types, detail, picture, location)
            VALUES ($1, $2, $3, $4, $5)
            RETURNING *;
        `;
        const values = [name, types, detail, fileUrl, location]; // Pastikan urutan values sesuai query

        const { rows } = await pool.query(query, values);
        res.status(201).json({ success: true, data: rows[0] });
    } catch (error) {
        res.status(500).json({
            success: false,
            message: error.message,
        });
    }
}
```

IV. Implementation

Backend Implementation

2. **Admin Verification Endpoint:**
   - Allows admins to review reports, update their status, and provide feedback.

```
async function login(req, res) {
    const { email, password } = req.body;

    try {
        const result = await pool.query(
            'SELECT * FROM admin WHERE email = $1',
            [email]
        );

        if (result.rowCount === 0) {
            return res.status(401).json({ error: "Incorrect email or password"
});
        }

        const admin = result.rows[0];

        const isPasswordValid = await bcrypt.compare(password, admin.password);
        if (!isPasswordValid) {
            return res.status(401).json({ error: "Incorrect email or password"
});
        }

        // Generate token (example using JWT)
        const jwt = require('jsonwebtoken');
        const secret = process.env.JWT_SECRET || 'defaultSecretKey';
        const token = jwt.sign({ id: admin.id, email: admin.email }, secret, {
            expiresIn: '1d', // Token valid for 1 day
        });

        res.status(200).json({
            message: "Login Successful",
            token,
            account: {
                id: admin.id,
                email: admin.email,
                name: admin.name,
            },
        });
    } catch (error) {
        console.error('Error in login:', error);
        res.status(500).json({ error: "An error occurred" });
    }
}
```

IV. Implementation

Backend Implementation

Implementation Process

- API routes were structured into two main categories: User Routes and Admin Routes.
- Data validation logic utilized the Joi library to ensure completeness and proper formatting.
- Uploaded files (accident photos) are stored in cloud storage (Cloudinary), with references saved in the database.

IV. Implementation

Database Implementation

The UIHelp database uses PostgreSQL for storing report data and NeonDB for database replication to support high-performance access.

Table Design

1. **Admin Table:**
   - Stores user information such as name, email, and status (admin or user).

```
CREATE TABLE admin (
    id SERIAL PRIMARY KEY,
    name VARCHAR NOT NULL,
    email VARCHAR NOT NULL CHECK (email ~* '^[^@]+@[^@]+\.[^@]+$'),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    password VARCHAR NOT NULL
);
```

2. **Report Table:**
   - Contains details of accident reports, including descriptions, photos, location coordinates, verification status, and timestamps.

```
CREATE TABLE report (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    detail TEXT NOT NULL,
    types report_type,
    status report_status NOT NULL DEFAULT 'Pending',
    picture TEXT,
    location TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

IV. Implementation

Database Implementation

Implementation Process

- The database was initialized using **SQL** scripts that defined tables and relationships between users and reports.
- **NeonDB** replication was configured to provide real-time data access with optimized performance.

V. Test Plan

1. Testing Objectives

The objective of this testing is to ensure that the UIHelp application operates according to the specified requirements and user needs. The testing process will verify the reliability, functionality, and performance of the application in real-world usage scenarios, as well as the seamless integration of its features, including:

- Disaster Reporting Form: Ensures reports can be submitted with complete and accurate data.
- Real-Time Map: Displays the locations of ongoing incidents accurately.
- Notification System: Delivers real-time updates to users promptly.
- Direct Communication with PLK: Ensures messages are sent and responded to by Campus Security (PLK).

This testing also aims to identify potential bugs or errors that may impact the user experience and address them before the official release of the application.

V. Test Plan

2. Scope of Testing

The scope of testing includes all key features of the UIHelp application and its compatibility with supported platforms. The testing coverage is as follows:

2.1 Features to Be Tested

- Disaster Reporting Function:
  - Filling out the disaster reporting form.
  - Sending data to the server and storing it in the database.
- Real-Time Map:
  - Displaying disaster locations using pins.
  - Refreshing data in real-time when new reports are submitted.
- Admin Page :
  - Displaying disaster statisctics
  - Manage disaster status

2.2 Types of Testing Performed

- Functional Testing: Ensures all features operate as specified.
- Compatibility Testing: Verifies the app's performance on various devices and browsers.
- Performance Testing: Ensures the app performs well under high loads.

2.3 Testing Platforms

- Operating Systems: Android, iOS, Microsoft.
- Devices: Smartphones, Tablets, Laptops.
- Browsers: Google Chrome, Mozilla Firefox, Safari.

V. Test Plan

3. Testing Environment

The testing environment is designed to simulate real-world conditions under which the UIHelp application will be used. It includes relevant hardware, software, and data for thorough testing.

3.1 Software

- Frontend: ReactJS.
- Backend: NodeJS.
- Database: PostgreSQL, NeonDB.
- Testing Tools:
  - Postman (for API testing).

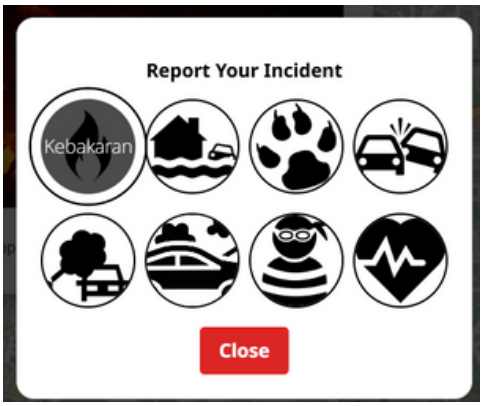
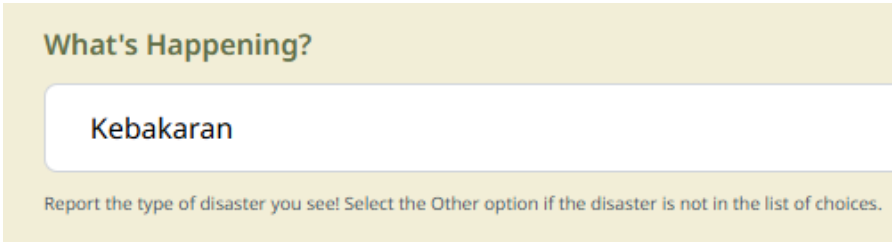
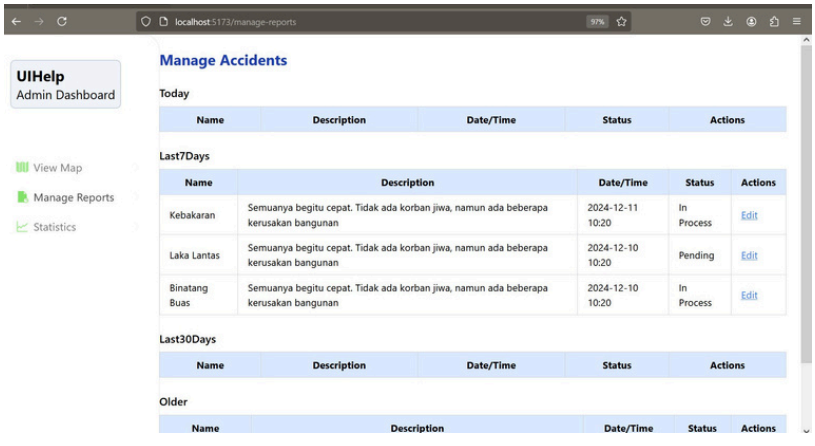
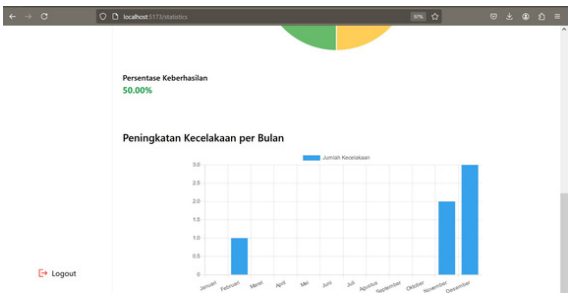
3.2 Test Data

- Dummy Data:
  - Disaster reports: 5 sample entries.
  - Locations: Campus UI coordinate data.
  - Users
- Real Data:
  - Disaster locations within the campus (with permission from PLK).

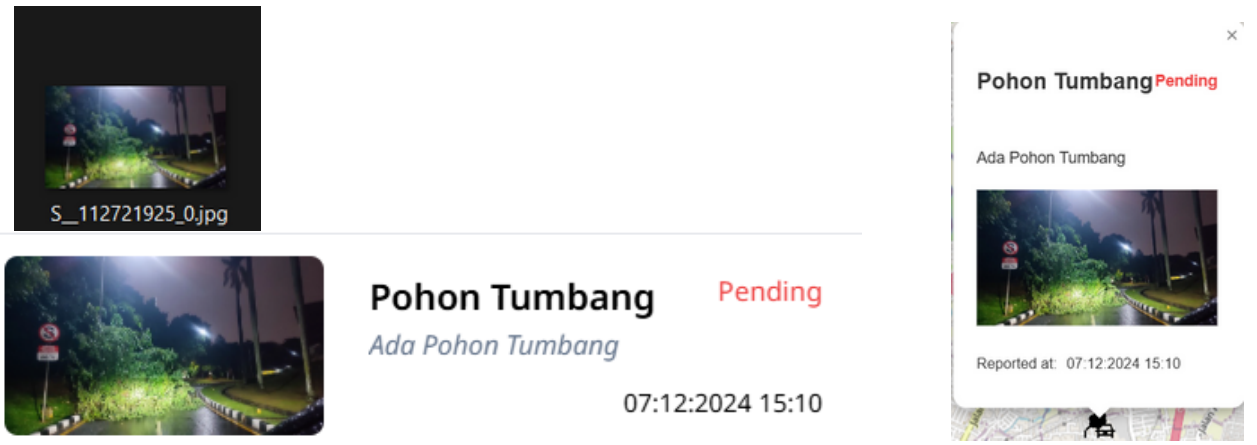
4. Test Cases

Test ID	Test Description	Test Steps	Expected Result	Actual Result	Status
TC01	Verify that users can create a disaster report	1. Open the disaster report form. 2. Fill in all required fields. 3. Click the 'Submit' button.	The report is successfully submitted, and a confirmation message is displayed.	The report has been successfully submitted and saved in the database.	Pass
					
TC02	Verify the real-time map displays disaster locations	1. Open the campus map in the app. 2. Observe the pins indicating disaster reports.	The map displays pins at disaster locations in real-time.	The map is able to display disaster location pins in real-time.	Pass
	 				

4. Test Cases

Test ID	Test Description	Test Steps	Expected Result	Actual Result	Status
TC03	Verify the disaster report form auto-fills the disaster type field based on the homepage selection	1. Open the homepage and try to create a report by selecting a disaster type. 2. Observe the disaster type field in the report form.	The disaster type field in the report form auto-fills based on the homepage selection.	The website successfully auto-fill the disaster type field in the report form.	Pass
					
TC04	Verify that data is saved for the admin when a new report is submitted	1. Submit a disaster report. 2. Check the admin database.	Admin receives complete information about the latest report in the database.	The website is able to save the data for the admin's report summary.	Pass
					

4. Test Cases

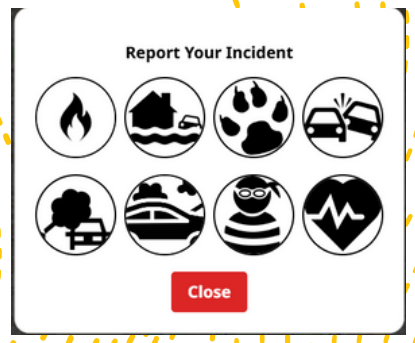
Test ID	Test Description	Test Steps	Expected Result	Actual Result	Status
TC05	Verify image can be uploaded properly to be checked by Admin	1. Select the image from local files or capture it from the camera and submit 2. Confirm the image appears in the admin's pending review section	The image is successfully uploaded and visible in the admin's review section.	The image is successfully uploaded and visible in the admin's review section.	Pass
					
TC06	Verify reporter's location (coordinates)	1. Access the disaster report form page. 2. Check if the location coordinates are accurate.	The coordinates and location match the reporter's current location.	The coordinates and location match the reporter's location when reporting the disaster.	Pass
					

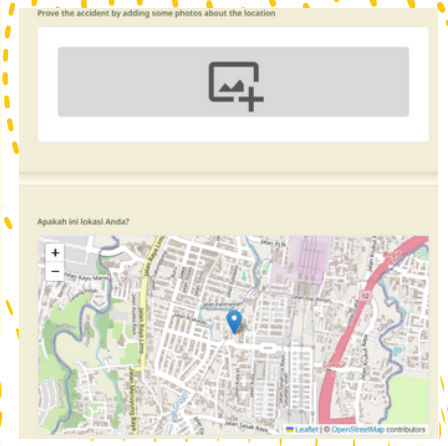
VI. User Manual

UIHelp is a disaster reporting platform for the Universitas Indonesia (UI) community, enabling users to report incidents, view real-time updates on a campus map, receive notifications, and communicate directly with Campus Security (PLK). This manual provides step-by-step guidance on how to use the features effectively.

1. Reporting a Disaster
   - Click "Report Disaster" on the homepage.

   - Fill in the form: Type of Incident, Location, Description and Photos.





   - Submit and get confirmation instantly!
2. Real-Time Map
   - Explore the map for incident updates.

   - Click pins for more details.

VII. User Testing Form

1. How well does the app function overall?
    - Extremely Well
    - Very Well
    - Well
    - Slightly Poorly
    - Poorly
2. How easy was it to navigate the app and use its features?
    - Extremely Easy
    - Very Easy
    - Easy
    - Slightly Difficult
    - Difficult
3. How useful did you find the disaster reporting feature?
    - Extremely Useful
    - Very Useful
    - Useful
    - Slightly Unhelpful
    - Unhelpful
4. How effective was the real-time map in providing disaster updates?
    - Very Effective
    - Effective
    - Neutral
    - Ineffective
    - Very Ineffective
5. How satisfied are you with the app's map system?
    - Very Satisfied
    - Satisfied
    - Neutral
    - Dissatisfied
    - Very Dissatisfied

VII. User Testing Form

6. How would you rate the app's overall performance (speed and responsiveness)?
    - Excellent
    - Good
    - Average
    - Below Average
    - Poor
7. How good is the integration system to the gallery/camera?
    - Excellent
    - Good
    - Average
    - Below Average
    - Poor
8. How accurate was the disaster location displayed on the real-time map?
    - Very Accurate
    - Accurate
    - Neutral
    - Inaccurate
    - Very Inaccurate
9. How helpful did you find the automatic disaster type selection feature?
    - Extremely Helpful
    - Very Helpful
    - Helpful
    - Slightly Unhelpful
    - Unhelpful
10. How well did the app meet your expectations overall?
    - Exceeded Expectations
    - Met Expectations
    - Neutral
    - Slightly Below Expectations
    - Did Not Meet Expectations

- What improvements would you like to see in the app?
- Do you have any additional comments or feedback about your experience with the app?

VII. User Testing Form

https://forms.gle/CX9u9BSnxkvPd8xC9

Referensi

- [1] "React Documentation," ReactJS. [Online]. Available: https://reactjs.org/docs/getting-started.html. [Accessed: Dec. 07, 2024].
- [2] "Express Documentation," ExpressJS. [Online]. Available: https://expressjs.com/en/starter/installing.html. [Accessed: Dec. 07, 2024].
- [3] "PostgreSQL Documentation," PostgreSQL Global Development Group. [Online]. Available: https://www.postgresql.org/docs/. [Accessed: Dec. 07, 2024].
- [4] J. Smith and M. Brown, "Real-time disaster management applications using cloud-based platforms," IEEE Access, vol. 7, pp. 12345-12357, 2022.
- [5] A. Kumar, "A study on geospatial mapping for emergency response," IEEE Transactions on Geoscience and Remote Sensing, vol. 59, no. 5, pp. 2334-2345, May 2021.
- [6] Universitas Indonesia, "Sustainability goals for campus safety," Internal Report, Universitas Indonesia, Depok, 2023.
- [7] "Flutter Documentation," Flutter. [Online]. Available: https://flutter.dev/docs. [Accessed: Dec. 07, 2024].
- [8] K. Patel, "The impact of notification systems on disaster response," in Proc. IEEE Int. Conf. on Humanitarian Technologies, New York, USA, 2020, pp. 125-130.