GUARDIAN TRACK

Smart Anti-Theft Alarm Mobile Application

Members:

Caballes, Almiko Jasper

Garciano, Steven

Lesion, Jhey Nan Jhon

Estandarte, Christian Jay

Gonzales, Lord Edmund

May 22, 2024

INTRODUCTION

Project Overview

Guardian Track mobile app is originally part of a capstone project, given that it is a capstone project it can't be use with just anyone, the app will require a system (hardware) installed in the vehicle to be able to use the application features. These features are Real-time Vehicle Tracking, Alarm status monitoring and vehicle status updates.

Real-time vehicle tracking: Through the integrated GPS module in the system hardware, users can locate their vehicle's location using the app. The app utilizes a map that allows users to easily navigate to their vehicle's location. Users will see two markers on the map: one for their vehicle and one for their current location.

Alarm status monitoring: Users can easily toggle the system functionalities on or off within the app. When turned on, the app will receive all data transmitted from the hardware, enabling users to monitor its functionalities in real-time.

Vehicle Status Updates: Users will know what really happen to their vehicle, because the hardware can detect four level of severities, level 1 will be considered as someone bumped to your vehicle or something hit your vehicle and level 4 is the highest and will be alarming the user of theft.

Objectives

1. **Develop a Reliable Mobile Application:**
    - Design and implement the Guardian Track mobile app with a user-friendly interface for seamless navigation and interaction.
2. **Integrate Real-Time Vehicle Tracking:**
    - Develop map integration within the app to display both the vehicle's and user's current locations with precise markers.
3. **Provide Comprehensive Vehicle Status Updates:**
    - Develop a system within the app that categorizes and communicates the severity of events detected by the hardware.

Scope

The Guardian Track mobile application will provide a robust vehicle security solution through an easy-to-use app compatible with Android and iOS. It will connect to hardware installed in the vehicle to offer real-time GPS tracking, displaying the vehicle's and user's locations on an interactive map. Users can manage the vehicle’s alarm system, turning it on and off and receiving real-time updates about their vehicle's status. These updates will include alerts for incidents ranging from minor bumps to potential theft. The project ensures secure communication between the hardware and app, protecting user data. Extensive testing will ensure the app works effectively in various conditions, and user feedback will help refine its features. Additionally, the project will offer clear user guides and support to help users make the most of the app.

The goal is to deliver a reliable and efficient anti-theft solution that enhances vehicle security and user confidence.

Target Audience

The Guardian Track mobile app is designed for vehicle owners who want to ensure their vehicles are secure. This includes regular people who care about their car's safety. Basically, anyone who wants peace of mind about their vehicle's safety can benefit from using Guardian Track.

PROJECT PROPOSAL

Background

The Guardian Track project aims to address the growing concern among vehicle owners regarding the security of their vehicles. With increasing instances of vehicle theft, there is a pressing need for a reliable and user-friendly solution to monitor and protect vehicles in real-time. Traditional security measures may fall short in providing comprehensive coverage, hence the development of the Guardian Track mobile application.

Problem Statement

Vehicle owners often face challenges in effectively monitoring and safeguarding their vehicles against theft and damage. Existing security systems may lack the ability to provide real-time tracking and alerts, leaving vehicles vulnerable to theft and unauthorized access. Additionally, the complexity and cost associated with implementing such systems can be prohibitive for many users. There is a need for a more accessible and efficient solution that combines real-time tracking, alarm monitoring, and status updates in a single, user-friendly platform.

Goals and Deliverables

1. GuardianTrack mobile application for Android platforms.

2. Integrated hardware system compatible with vehicles.

3. Real-time tracking functionality within the mobile app.

4. User interface for easy toggling of system functionalities.

5. Documentation and user manual for installation and usage

Timeline

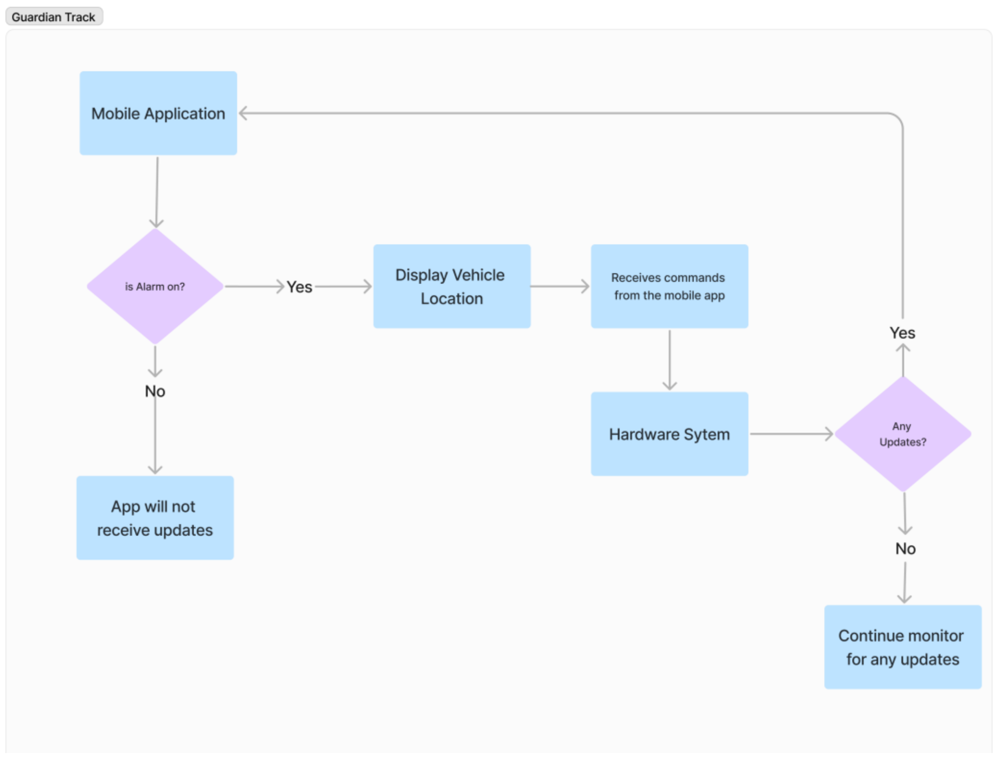
TASK	**START DATE**	**END DATE**	**DURATION**
Planning Phase	02/15/2024	03/01/2024	**15 days**
Design Phase	03/02/2024	03/10/2024	**9 days**
Development Phase	03/11/2024	04/01/2024	**20 days**
Testing Phase	04/4/2024	04/10/2024	**7 days**
Final Review & Refine	04/11/2024	05/09/2024	**40 days**

SYSTEM ARCHITECTURE

System Components

1. **Mobile Application:**
   - User Interface: Provides a user-friendly interface for vehicle tracking, alarm monitoring, and status updates.
   - GPS Module Integration: Displays real-time location data on an interactive map.
   - Notification System: Sends alerts to users about vehicle status and incidents.
   - Control Panel: Allows users to toggle alarm system functionalities.
2. **Hardware System:**
   - GPS Tracker: Installed in the vehicle to provide real-time location data.
   - Alarm System: Detects unauthorized access or impacts and sends alerts to the mobile app.
   - Communication Module: Transmits data between the hardware and the mobile application
3. **Backend Server:**
   - Database: Stores user information, vehicle data, and logs of all events and alerts.
   - Notification Service: Manages the sending of real-time alerts and updates to users.

Data Flow Diagram



1. **Mobile Application:**
   - Sends user commands (e.g., alarm on/off) to the backend server.
   - Displays real-time vehicle location and status updates.
2. **Backend Server:**
   - Receives commands from the mobile app and forwards them to the hardware system.
   - Receives data from the hardware system and updates the mobile app.
   - Manages user data and event logs.
3. **Hardware System:**
   - Sends real-time location data and alerts to the backend server.
   - Executes commands received from the backend server.

Technology Stack

For the development of the Guardian Track system, we will utilize the following technologies, frameworks, and tools:

Android Studio: We chose Android Studio for app development due to its robust features, extensive documentation, and strong community support. It provides a comprehensive integrated development environment (IDE) specifically tailored for Android app development, allowing us to efficiently create a user-friendly and responsive mobile application for Android devices.

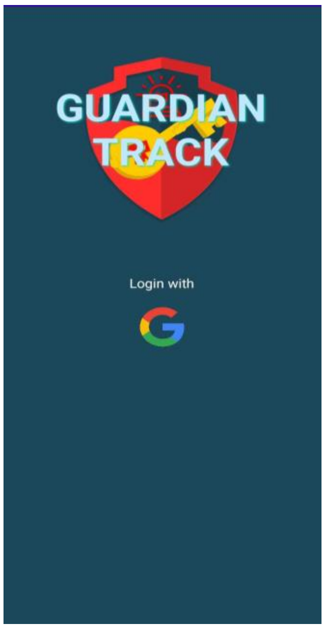
MongoDB Atlas: MongoDB Atlas will serve as our database solution. We selected MongoDB for its flexibility, scalability, and ease of integration with our chosen technology stack. MongoDB Atlas, being a fully managed cloud database service, offers high availability, automatic backups, and built-in security features, ensuring reliable and secure storage of our application data.

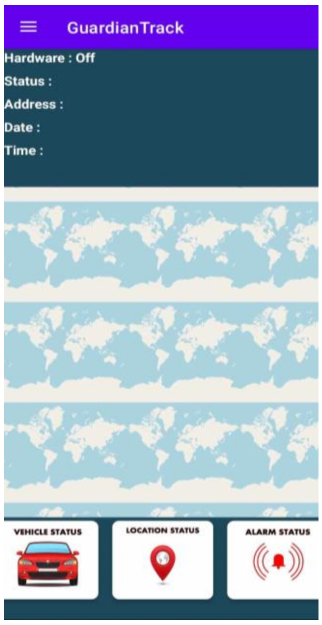
Google API: We will leverage Google API for user authentication and login functionality. Google API provides a seamless and secure way for users to authenticate using their Google accounts, offering convenience and familiarity to users while ensuring authentication security for our application.

Vercel: Vercel will be used for server-side deployment and hosting. We opted for Vercel due to its simplicity, scalability, and support for serverless functions. Vercel's seamless integration with popular frontend frameworks and its automatic deployment capabilities will streamline our development process and ensure efficient deployment of updates and changes to our application.

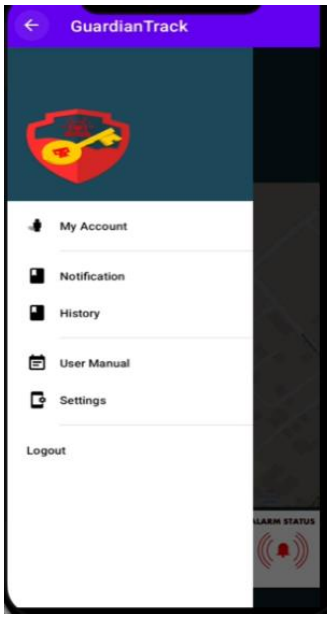
By leveraging these technologies, frameworks, and tools, we aim to develop a robust, scalable, and user-friendly Guardian Track system that effectively addresses the needs of our users while ensuring seamless integration, security, and reliability throughout the development and deployment process.

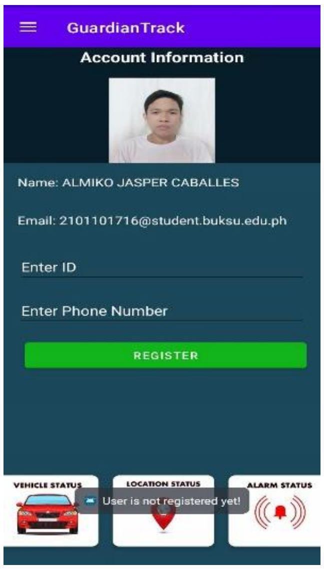
DESIGN





Hardware : Off
Status :
Address :
Date :
Time :

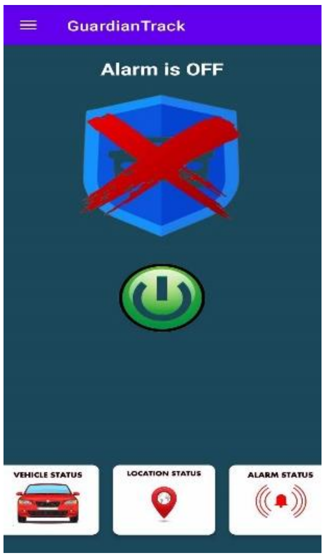




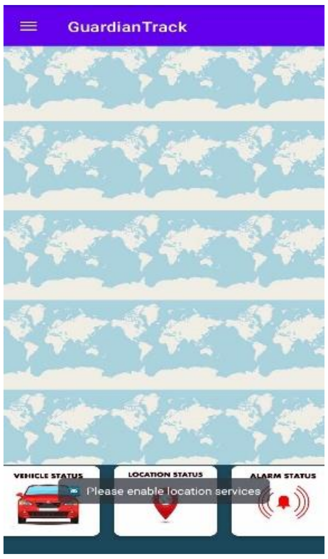
Name: ALMIKO JASPER CABALLES

Email: 2101101716@student.buksu.edu.ph

User is not registered yet!



Alarm is OFF



Please enable location services

DEVELOPMENT

1. **Android Studio:**

   - **Installation:** Download and install Android Studio from the official website.
   - **SDK Setup:** Ensure the Android SDK is installed and configured within Android Studio.
   - **Emulator Configuration:** Set up an Android emulator for testing or connect a physical device via USB for live testing.

2. **OpenStreetMap:**

   - **Library Integration:** Add the OpenStreetMap library to the project. This can be done by including the relevant dependency in the build.gradle file.

3. **MongoDB:**

   - **Database Setup:** Set up a MongoDB instance using MongoDB Atlas.

4. **Vercel:**

   - **Deployment Setup:** Set up a Vercel account and link the project repository.
   - **Configuration:** Configure Vercel for automatic deployments from the main branch of the code repository.

5. **Google API for Login:**

   - **API Console Setup**: Create a project in the Google API Console and enable the Google Sign-In API.
   - **Credentials:** Generate OAuth 2.0 credentials and configure them in the Android project.

Code Repository

- **GitHub:**
  - Repository Creation: Create a new repository on GitHub for the project.
  - Repository Structure: Organize the repository with separate folders for the Android app, backend server, and documentation.
  - Version Control: Use Git for version control, committing changes regularly and pushing them to the GitHub repository.

API Integration

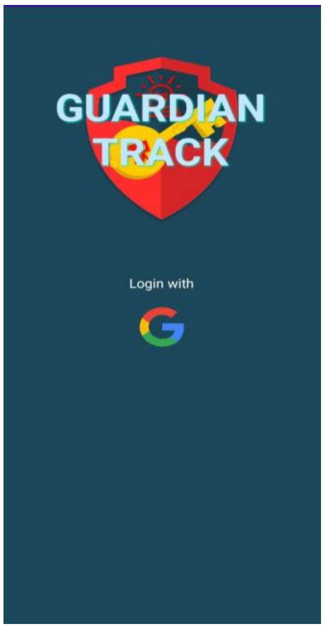
1. **MongoDB Integration:**
   - **Database Operations:** Implement CRUD operations for managing user data, vehicle data, and event logs.
   - **Connection Handling:** Ensure efficient and secure handling of database connections within the backend server.
2. **Google Login Integration:**
   - **Authentication: I**mplement Google Sign-In in the Android app using the Google Sign-In API.
   - **Token Handling:** Securely handle authentication tokens and verify them on the backend server to authenticate users.
3. **OpenStreetMap Integration:**
   - **Map Display:** Integrate OpenStreetMap into the Android app to display real-time vehicle locations.
   - **Marker Management:** Implement functionality to add, update, and remove markers on the map based on real-time data.

DOCUMENTATION

User Manual

Login:

To access GuardianTrack you will need to log in using your registered credentials (Google Account). Follow these steps to log in:



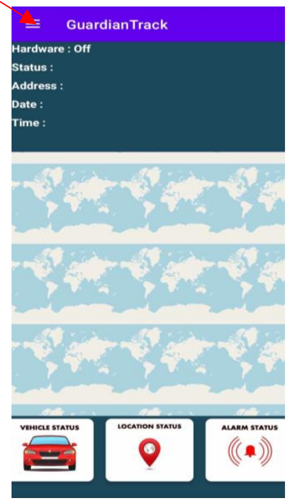
1. Open the GuardianTrack app
2. On the "Login with" section click google icon
3. Enter your credentials

If you encounter any issues while logging in, please make sure that you have entered the correct username and password. If you've forgotten your password, you can click on the "Forgot Password" link in google to reset it.

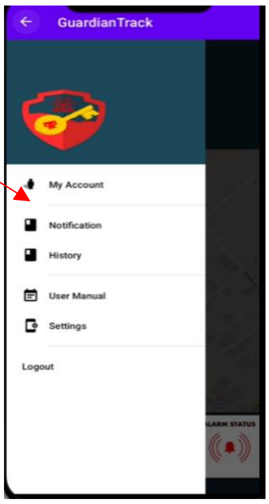
Register:

If you're new to GuardianTrack you'll need to create an account. Here's how to register:

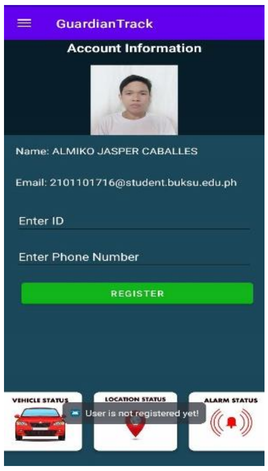
Step 3



Step 4



1. Open the GuardianTrack app
2. Login using google credentials (if already logged in skip step 2)
3. In the top-right corner click the three lines and a sidebar will appear
4. Click on the "Account"



Name: ALMIKO JASPER CABALLES

Email: 2101101716@student.buksu.edu.ph

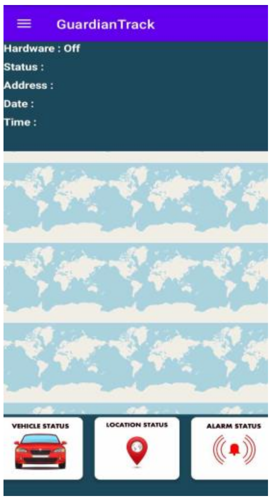
User is not registered yet!

5. Enter ID and Phone Number and click register button
6. Once registered you can now access the apps functionality

Note: The ID is unique and reguired. Each hardware consists with one ID registered in the DATABASE, if ID is not yet registered in the database you can not register in the app and cannot access its functionality

Navigations:

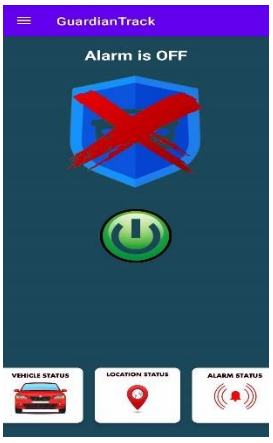
Navigating through GuardianTrack is intuitive and user-friendly. Here are some tips for efficient navigation:



Hardware : Off
Status :
Address :
Date :
Time :

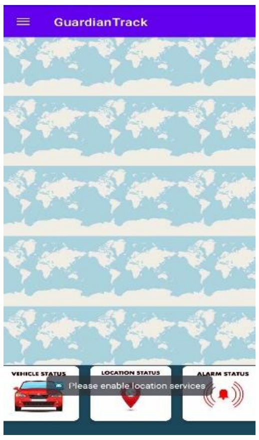
Dashboard: The home page is your central hub for accessing content and updates from hardware. You can also navigate to other two functionalities within the dashboard these functionalities are follows:

ALARM STATUS

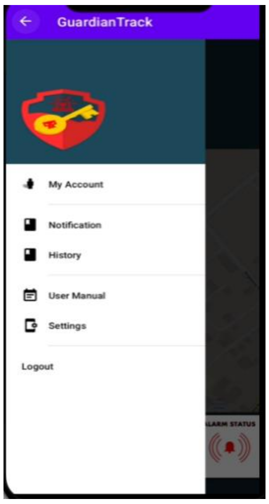


In Alarm Status, you can turn and off the functionality of the app, whether it will receive a data from hardware or not.

LOCATION STATUS



In Location Status, you will see the current location of the use and also the location of the vehicle, if alarm is on you will see here the pinned location of your vehicle.



Menu Bar: Use the menu bar located at the side of the screen to access different sections of the app such as your account, settings, and history, and notifications.

Technical Documentation

1. **Overview:**

   The Guardian Track mobile application is designed to enhance vehicle security through real-time tracking, alarm monitoring, and status updates. It integrates various components, including a mobile app, backend server, hardware system, and multiple third-party services.

2. **System Components:**

- Mobile Application:
  - Developed using Android Studio.
  - Utilizes OpenStreetMap for displaying real-time vehicle locations.
  - Integrated with Google API for user authentication.
- Backend Server:
  - Hosted on Vercel.
  - Uses Node.js and Express.js for API endpoints.
  - Connects to MongoDB for data storage.
- Hardware System:
  - Includes a GPS module for tracking.
  - Features an alarm system for security alerts.
  - Uses a communication module for data transmission.

3. Development Environment Setup:

- Android Studio:
  - Install Android Studio and configure the Android SDK.
  - Set up an emulator or physical device for testing.
- OpenStreetMap:
  - Add OpenStreetMap library to the Android project.
  - Configure API keys if required**.**
- MongoDB:
  - Set up MongoDB Atlas or a local MongoDB server.
  - Use MongoDB driver for backend integration.
- Vercel:
  - Create an account on Vercel and link the GitHub repository.
  - Configure for automatic deployments from the main branch.
- Google API for Login:
  - Create a project in the Google API Console and enable Google Sign-In.
  - Generate OAuth 2.0 credentials and configure them in the Android project.

4. **Code Repository:**

- Hosted on GitHub.
- Organized with folders for the Android app, backend server, and documentation.
- Utilizes Git for version control with regular commits and pull requests for collaboration.

5. **Technology Stack:**

- Frontend: Android Studio, OpenStreetMap, Google API.
- Backend: MongoDB.
- Hosting: Vercel.
- Version Control: GitHub.

API Documentation

1. Overview:

The API acts as a bridge connecting various components, including the Guardian Track mobile application, Hardware, the backend server, user authentication, vehicle tracking, alarm management, and status updates.

Endpoints

1. Check Pin Location

- URL: /checkpinlocation

- Method: POST
- Description: Retrieves the latest pin location for a given uniqueId.
- Request Body:{ "uniqueId": "string"}
- Response:{ "latitude": "number", "longitude": "number", "time": "string", "address": "string"}

2. Get Location

- URL: /getlocation
- Method: GET
- Description: Retrieves all location data for a given uniqueId.
- Query Parameters: uniqueId: string
- Response: { "latitude": "number", "longitude": "number", "time": "string", "address": "string"}

3. Save data from hardware

- URL: /data
- Method: POST
- Description: Saves vibration and location data.
- Request Body:{ "vibrationDuration": "number", "latitude": "number", "longitude": "number", "uniqueId": "string", "level": "string"}
- Response:
    - 200 OK: Data saved successfully.
    - 500 Internal Server Error: Failed to save data.

4. Update Current Location

- URL: /currentlocation
- Method: POST
- Description: Updates the pin location for a given user.

- Request Body:{ "name": "string", "uniqueId": "string", "email": "string", "cellphonenumber": "string", "pinlocation": "string"}
- Response:
    - 200 OK: Pin Successfully.
    - 400 Bad Request: User or hardware not found.
    - 500 Internal Server Error: Internal server error.

6. Stop Current Location

- URL: /stopcurrentlocation
- Method: POST
- Description: Stops the current location tracking and saves the pin location.
- Request Body:{ "uniqueId": "string", "pinlocation": "string", "currentlatitude": "number", "currentlongitude": "number", "statusPin": "boolean"}
- Response:
    - 200 OK: Location updated successfully.
    - 400 Bad Request: Invalid location.
    - 500 Internal Server Error: Internal server error.

7. Check User Registration

URL: /checkuserregister

Method: POST

Description: Checks if a user is registered.

Request Body: {"userName": "string", "email": "string"}

Response: { "uniqueId": "string", "name": "string", "email": "string", "cellphonenumber": "string"}

- 404 Not Found: User not registered yet.

        - 500 Internal Server Error: Internal server error.

8. Get Current Location Status

URL: /getcurrentlocation

Method: GET

Description: Retrieves the current location status of the hardware.

Query Parameters: uniqueId: string,status:boolean

Response: { "status": "boolean"}

9. Send Theft Details

- URL: /sendtheftdetails
- Method: POST
- Description: Sends theft details and saves them to the database.
- Request Body:{ "uniqueId": "string", "currentlatitude": "number", "currentlongitude": "number", "description": "string", "level": "string"}
- Response:
    - 200 OK: Theft details saved successfully.
    - 500 Internal Server Error: Internal server error.

10. Get Theft Details

- URL: /gettheftdetails
- Method: POST
- Description: Retrieves the latest theft details for a given uniqueId.
- Request Body: {"uniqueId": "string"}

- Response: {"latitude": "number", "longitude": "number", "time": "string", "description": "string", "level": "string", "address": "string"}

11. User Registration

- URL: /userregister
- Method: POST
- Description: Registers a new user.
- Request Body: { "name": "string", "uniqueId": "string", "email": "string", "cellphonenumber": "string"}
- Response:
    - 200 OK: User registered successfully.
    - 400 Bad Request: User is already registered or hardware ID not found.
    - 500 Internal Server Error: Internal server error.

12. Delete User

- URL: /deleteuser
- Method: POST
- Description: Deletes a user.
- Request Body: {"name": "string", "uniqueId": "string", "email": "string"}
- Response:
    - 200 OK: Successfully deleted.
    - 400 Bad Request: Unable to delete.
    - 500 Internal Server Error: Internal server error.

13. Hardware Registration

- URL: /hardwareregister
- Method: POST
- Description: Registers new hardware.
- Request Body: { "uniqueId": "string"}
- Response:
    - 200 OK: Hardware registered successfully.
    - 400 Bad Request: Hardware already registered.
    - 500 Internal Server Error: Internal server error.

14. Get User Number

- URL: /usernumber
- Method: POST
- Description: Retrieves the user's cellphone number for a given uniqueId.
- Request Body: { "uniqueId": "string"}
- Response: {"cellphonenumber": "string"}

15. Get Hardware Status

- URL: /hardwarestatus
- Method: GET
- Description: Retrieves the status of the hardware for a given uniqueId.
- Query Parameters:
- uniqueId: string
- Response: {"status": "boolean"}

16. Change Hardware Status

- URL: /changestatus
- Method: POST
- Description: Changes the status of the hardware.
- Request Body: { "uniqueId": "string", "status": "boolean" }
- Response: {"status": "boolean"}

17. Get History

- URL: /gethistory
- Method: POST
- Description: Retrieves the combined history from all collections for a given uniqueId.
- Request Body: { "uniqueId": "string"}
- Response: { "data": [{"collection": "string","latitude": "number", "longitude":"number","time": "string","level": "string","description": "string"   }]}

Project Recap

Project Overview

The Guardian Track mobile application was developed as part of a capstone project aimed at providing a reliable vehicle security solution. The application works in conjunction with hardware installed in the vehicle to offer features such as real-time vehicle tracking, alarm status monitoring, and vehicle status updates.

Objectives

Develop a Reliable Mobile Application: Create a user-friendly interface for seamless navigation and interaction.

Integrate Real-Time Vehicle Tracking: Implement map integration to display both the vehicle's and user's current locations with precise markers.

Provide Comprehensive Vehicle Status Updates: Develop a system to categorize and communicate the severity of events detected by the hardware.

System Components

1. Mobile Application:

User Interface: Offers a user-friendly platform for vehicle tracking, alarm monitoring, and status updates.

GPS Module Integration: Displays real-time location data on an interactive map.

Notification System: Sends alerts to users about vehicle status and incidents.

Control Panel: Allows users to toggle alarm system functionalities.

2. Hardware System:

GPS Tracker: Installed in the vehicle to provide real-time location data.

Alarm System: Detects unauthorized access or impacts and sends alerts to the mobile app.

Communication Module: Transmits data between the hardware and the mobile application.

3. Backend Server:

Database: Stores user information, vehicle data, and logs of all events and alerts.

Development Phases

Planning Phase: The project was carefully planned over a period of 15 days to outline the scope, objectives, and timeline.

Design Phase: This phase lasted 9 days and involved designing the user interface and system architecture.

Development Phase: Over a span of 20 days, the mobile application, backend server, and hardware system were developed and integrated.

Testing Phase: A 7-day period was dedicated to testing the application under various **conditions to ensure functionality and reliability.**

Final Review & Refine: This phase involved a thorough review and refinement of the application over 40 days based on user feedback and additional testing.

Technology Stack

Android Studio: For app development, offering a comprehensive IDE tailored for Android.

MongoDB Atlas: As the database solution, providing flexibility, scalability, and secure data storage.

Google API: For user authentication and login functionality.

Vercel: For server-side deployment and hosting, supporting serverless functions and automatic deployment capabilities.

Lessons Learned

Project Management: Effective planning and time management were crucial in keeping the project on track. Using project management tools helped in coordinating tasks among team members.

Team Collaboration: Regular communication and collaboration were key to addressing challenges and ensuring everyone was aligned with the project goals.

Technical Skills: Working with a diverse technology stack improved our proficiency in Android development, database management, and backend integration.

User-Centered Design: Gathering and incorporating user feedback helped us create a more intuitive and functional application.

Problem-Solving: We encountered various technical challenges, such as integrating the GPS module and ensuring real-time data transmission, which enhanced our problem-solving skills.

Acknowledgements

We would like to acknowledge several individuals and organizations that have significantly contributed to the successful completion of the Guardian Track project:

Online Communities and Forums: Various online communities and forums provided invaluable resources, solutions to technical problems, and inspiration from similar projects. The shared knowledge and collaborative spirit in these communities greatly enhanced our development process.

Open-Source Contributors: Developers and contributors of the open-source libraries and tools that we integrated into our project provided robust and well-documented software that was crucial for our development.

Capstone Project Mentors: Our mentors provided critical guidance and support throughout the project. Their expertise and feedback were instrumental in shaping the project and overcoming challenges.

Appendix

A. Additional References

GitHub Documentation: Comprehensive guides and documentation available on GitHub's official site helped us effectively use version control and collaborate seamlessly.

Stack Overflow: The community-driven Q&A platform provided solutions to various technical challenges we encountered during development.

MongoDB Documentation: MongoDB's official documentation helped us effectively utilize MongoDB Atlas for our database needs, ensuring secure and efficient data management.

Google API Documentation: The official Google API documentation provided guidelines and examples for integrating Google Sign-In and other Google services into our app.

B. References

1. MongoDB Documentation (https://www.mongodb.com/docs/)

2. Google API Documentation (https://developers.google.com/docs/api/reference/rest)

3. Android Developer Documentation (https://developer.android.com/develop)

4. GitHub Documentation (https://docs.github.com/en)

5. Stack Overflow (https://stackoverflow.com/)