GPS TASK REMINDER SYSTEM

A Project Report
Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF SCIENCE (DATA SCIENCE)

By

Galande Rushali Tatyasaheb

Seat No: _____

Under the esteemed guidance of
Mrs. Sunita Koli
Assistant Professor

DEPARTMENT OF DATA SCIENCE
N.G Acharya & D.K. Marathe
College Of Arts, Science & Commerce
(Affiliated to University of Mumbai)
NAAC Accredited "A" Grade
Shri. N. G. Acharya Marg, Chembur, Mumbai-71
MAHARASHTRA
2024-25

N.G. Acharya & D.K. Marathe
College Of Arts, Science & Commerce
(Affiliated to University of Mumbai)
NAAC Accredited "A" Grade
Shri. N. G. Acharya Marg, Chembur, Mumbai-71
MAHARASHTRA

DEPARTMENT OF DATA SCIENCE

CERTIFICATE

This is to certify that the project entitled, **"GPS Task Reminder System"**, is bonafide work of **Galande Rushali Tatyasaheb** bearing Seat No: _____ submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in DATA SCIENCE from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

ABSTRACT

The **GPS Task Reminder System** is designed to enhance task management by associating reminder messages with specific geographical locations. As the user approaches their intended destination, an alarm is triggered, displaying the associated message on their mobile screen and providing a voice alert. This system effectively tracks the user's location using GPS technology, retrieving their current geographical coordinates to ensure timely reminders.

Users can easily set, reset, disable, edit, and configure the duration of reminders. The system allows users to view their destination on a map, displaying their distance from the task location. Furthermore, users can create and manage multiple tasks, each linked to a unique location.

Upon entering task details, the user is presented with a map highlighting the task location, along with the distance from their current position. By selecting the "start" option, users will receive notifications as they approach the task location. These notifications will continue until the user selects the "finish" option, at which point the completed task will not be displayed in the system. Importantly, the application operates seamlessly in the background, allowing users to engage in other activities on their mobile devices without interruption.

ACKNOWLEDGEMENT

We would like to express our special thanks and gratitude to our project coordinator **Mrs. Archana Jadhav** for guiding us to do the project work on time and giving us all support and guidance, which made us complete our project duly. We are extremely thankful to her for providing such nice support and guidance.

We would also like to thank **N. G Acharya and D. K. Marathe College of Arts, Commerce & Science** for providing us with the necessary components for our project.

We are also thankful for and fortunate enough to get constant encouragement, support and guidance from the teachers of Data Science who helped us in successfully completing our project work.

DECLARATION

I hereby declare that the project entitled, **"GPS Task Reminder System"** done at **N.G. Acharya & D.K. Marathe College of Arts, Science & Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE (DATA SCIENCE) to be submitted as final semester project as part of our curriculum.

Galande Rushali Tatyasaheb

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR NO.: ……. **Roll No: ----------**

1. **Name of the Student**

_____

2. **Title of the Project**

_____

3. **Name of the Guide**

_____

4. **Teaching Experience of the Guide _____**

5. **Is this your first Submission: yes No**

Signature of the Student **Signature of the Guide**

Date: **Date:**

Signature of the Coordinator

Date:

TABLE OF CONTENTS

Chapter 1

Introduction

The GPS Task Reminder System is a mobile application designed to assist users in managing their daily tasks by providing location-based reminders. This system tracks the user's current geographical location using GPS and provides text or voice reminders when the user is near the task location. Unlike conventional systems that rely on Google Maps API for location services and map rendering, this project uses alternatives to fetch the user's GPS coordinates and displays task locations in a simplified manner.

Background:

With the increasing use of smartphones, location-based services (LBS) have gained popularity. Many users struggle to manage time-sensitive tasks that depend on their location, such as picking up groceries when near a supermarket or dropping off documents at a nearby office. Current applications use third-party services like Google Maps to handle location tracking and reminders, but these can be restrictive due to privacy concerns, API limitations, or cost. Therefore, building a GPS-based task reminder system without using the Google Maps API offers users more control over their data, cost-effectiveness, and the flexibility to implement custom functionalities.

Problem Definition:

The primary problem this system aims to address is the lack of efficient, customizable, and low-cost location-based task management without the need for external APIs like Google Maps. By developing a solution that directly utilises GPS coordinates and an alternative location-tracking mechanism, the system ensures timely reminders for users based on their proximity to task locations, without relying on paid or third-party services.

Objectives:

The main objectives of this project are:

1. **Task Creation and Management**: Allow users to create multiple tasks with a specific location, time, and type of reminder (text or voice).
2. **GPS-based Location Tracking**: Retrieve and track the user's location using native GPS functionality without using Google Maps API.
3. **Location-Based Reminders**: Send notifications (text or voice) to the user when they are near the task location.
4. **Background Service**: Ensure the system works in the background, continuing to track location and trigger reminders.
5. **Task Completion**: Enable users to mark tasks as completed once they've been addressed.

Purpose:

The purpose of this project is to develop a robust, cost-effective, and privacy-friendly task reminder system that empowers users to manage location-based tasks without depending on paid services like Google Maps. This not only reduces development and operational costs but also provides more flexibility in how location tracking and notifications are handled. By relying on alternative map services and custom GPS functionality, the system can be tailored to suit specific user needs.

Sope:

The scope of this system includes:

- **Domains**: This system can be utilised in various domains such as personal productivity, logistics, transportation, and location-based marketing. It is particularly beneficial for individuals or small businesses that want to manage tasks based on proximity without the need for extensive map-based services.

- **Extent**: The application will handle task management, GPS tracking, and reminder notifications. However, it will not provide turn-by-turn navigation or advanced map interactions, which are typically associated with mapping APIs. Instead, it will focus on retrieving coordinates, tracking distances, and providing notifications based on proximity.

Applicability:

This system can be used in various scenarios:

1. **Personal Task Management**: Users can set reminders for tasks that need to be completed when they are near a specific location, such as visiting a store or office.
2. **Logistics and Deliveries**: Delivery personnel can receive location-based reminders for specific delivery tasks when approaching a customer's address.
3. **Workplace Task Assignment**: Employees can use this system to track location-sensitive tasks such as fieldwork assignments.
4. **Retail and Marketing**: Businesses can provide location-based reminders to customers, notifying them of promotions when they are near a store.

Achievements:

The achievements expected from this project include:

1. **Independence from Third-Party APIs**: By not using the Google Maps API, the system will offer a more affordable and customizable solution for users.
2. **Custom Location-Based Reminders**: Users will be able to receive accurate and timely reminders based on their location, increasing their productivity and efficiency.

3. **Privacy and Data Control:** Users' location data will be handled within the system, giving them more control over their personal information and privacy.
4. **Low Development and Maintenance Costs:** The absence of API usage fees will make this solution more accessible and affordable for small businesses or individuals.

Chapter 2

Data Description

The GPS Task Reminder System operates by storing and managing data related to tasks, user locations, and reminder schedules. The data collected and managed by the system includes user information, task details (name, location, reminder type, time), location coordinates, and reminder notifications. The system relies on native GPS for location tracking and stores data in a local database or server-based architecture, depending on the setup.

1. Data Description

Data Categories:

- User Data: User identification, login credentials, and personalized settings.
- Task Data: Task name, location (latitude, longitude), reminder types (text/voice), and task completion status.
- Location Data: GPS coordinates (latitude and longitude) for tracking user proximity to tasks.
- Reminder Data: Task ID, time of reminder, reminder type, and notification status.

2. Data Source and Extraction

Data Source:

The data for the GPS Task Reminder System is primarily user-generated during interaction with the mobile application. The application uses:

- **Primary Data:** Collected directly from users as they create tasks, set reminders, and interact with the system.

- **Secondary Data**: Could include external data sources such as public geolocation databases (if used), but no third-party API such as Google Maps is involved.

Data Extraction:

- GPS Data: Collected directly from the device's built-in GPS sensor.
- User Data: Collected during registration and login.
- Task Data: Collected when a user creates a new task, and saved within the system’s database.

Licence to Use the Data:

For data collected directly from users (primary data), the system must implement a privacy policy clearly stating:

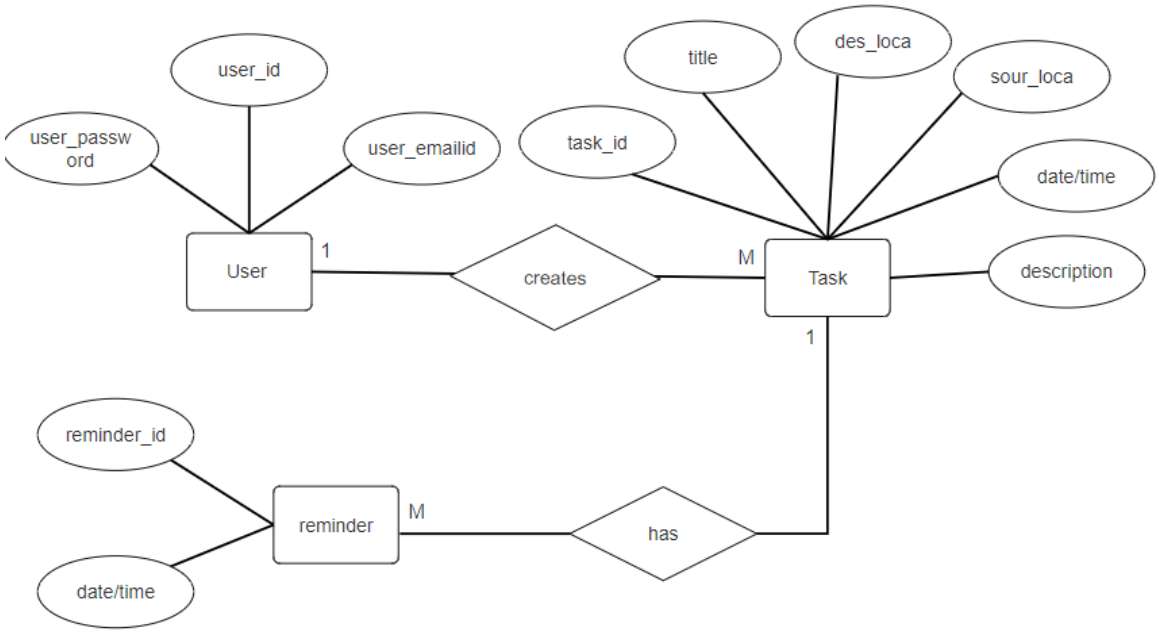
- How the data will be used.
- How long the data will be retained.
- How the data will be protected.

If public geolocation databases or other external data are used (secondary data), ensure that the sources have valid open licences (e.g., OpenStreetMap) or that the data use complies with their licensing agreements. A sample licence from such a data source (if applicable) should be included.

3.Data Management and modelling:

 Entity: A distinct object in the system, such as a user, task, or reminder, which holds specific attributes.
 Attribute: A characteristic or property of an entity, such as user_id or title.
 Relationship: Defines how entities are associated with each other. In this case, users create tasks, and tasks have reminders.

 Cardinality: Shows the number of instances involved in the relationship. The numbers (1, M) represent cardinality, where "1" means one instance, and "M" means many instances



4.Data Pre-Processing

Before using the data for task tracking and reminders, pre-processing is essential to ensure the quality and accuracy of the information. Some of the key steps include:

- **Data Validation**: Ensure that all fields (e.g., latitude, longitude, task names, etc.) are valid and contain accurate data.
- **Data Cleaning:** Remove incomplete, duplicate, or inconsistent entries, especially for tasks or reminders with missing or incorrect location data.
- **Normalisation:** Normalise data to ensure consistent formats for fields such as dates, GPS coordinates, and task names.

- **Data Encryption:** Encrypt sensitive user data, especially passwords, for security purposes.

5.Data Engineering

1. Location Tracking Algorithm:

The system uses native GPS to track the user's current coordinates and continuously compares them with stored task locations. An example of this could involve the Haversine formula to calculate the distance between two geographical points based on latitude and longitude.

2. Notification Scheduling:

For each task reminder, the system schedules notifications based on the user's proximity to the task location. This is achieved by periodically checking the user's location in the background and triggering reminders when the user is within a specified radius of the task location.

3. Reminder Logic:

The system checks the current task status and decides whether to activate or disable the reminder. Once the task is completed, the associated reminder is marked as inactive.

4. Data Storage:

The system uses a local database (e.g., Firebase) or a server-side database to store task, user, reminder, and notification data. Proper indexing and optimization techniques are applied to ensure efficient data retrieval, especially for large datasets.

6.Data Analysis

1. Basic Statistical Analysis

- **Task Data:** Summarise the number of tasks created by users, average tasks per user, and the distribution of task types (personal, work, errands, etc.).
- **Reminder Data:** Evaluate how often reminders are set, the average duration between task creation and task completion, and the most common times when reminders are triggered.
- **Location Data:** Analyse user movement patterns by summarising distances between task locations and user locations when reminders are triggered.

7.Data Visualization

1. Task Distribution by Location:

- **Visualisation Type:** Geographical map (scatterplot or heatmap).
- **Explanation:** Visualise where users frequently set their tasks. A heatmap can show areas with the highest density of tasks, helping users understand patterns of where they typically set reminders.

2. Task Completion Timeline:

- **Visualisation Type:** Line chart or bar chart.
- **Explanation:** Display how long users typically take to complete tasks after reminders are triggered. This could be visualised as a timeline to see how task completion times vary throughout the day.

3. Reminder Effectiveness:

- **Visualisation Type:** Pie chart or stacked bar chart.
- **Explanation:** Show the ratio of tasks completed after receiving text reminders vs. voice reminders. This helps in understanding which type of reminder is more effective.

Chapter 3

Methods and Algorithms

1. Methods and Algorithms

The GPS Task Reminder System aims to trigger reminders based on a user's proximity to specific locations. The system utilises location tracking through GPS data and manages tasks and reminders using backend services. Below is an explanation of methods and algorithms used for various components of the system.

2. Selecting Features

Feature Selection Process:

The core features of the system include:

- **Task Details:** Task name, task type, location (latitude, longitude), and task completion status.
- **Reminder Data:** Reminder type (text/voice), scheduled time, and reminder trigger status.
- **User Location:** Real-time GPS coordinates (latitude and longitude).
- **Distance to Task:** Calculated distance between user's current location and the task's location.

Feature Engineering:

- **Transforming Features:**
    - The location coordinates (latitude and longitude) are transformed into distance metrics using the Haversine formula to calculate the user's proximity to task locations.

- **Creating New Features:**
    - **Time of Day**: This feature is created to optimize reminder triggers by identifying the best time for sending reminders.
    - **Task Priority:** A priority level is created based on the urgency of the task (for example, work tasks might have higher priority than errands).

Removing Features:

Unnecessary features like exact timestamp logs of each user movement may be removed to streamline the data and focus only on task-relevant information.

3. Building the Dataset

Training and Testing Data:

The dataset consists of user-generated data from interactions with the system. To build an intelligent reminder system, historical data can be used to train models for predicting optimal reminder times or task completion likelihood.

- **Training Data:** 70% of the data will be used for training the system. This data includes information about tasks created, completed tasks, reminder triggers, and user locations at the time of task completion.
- **Testing Data:** 30% of the data is reserved for testing the system's performance on unseen data. This split is chosen to balance the need for a robust training model while leaving enough data for validation.

Data Collection:

- User Location Data: Collected via GPS.
- Task Creation and Reminder Data: Collected during user interaction with the system.

4.Algorithm

1. Proximity-Based Task Reminders (Best for Simple Reminders)

- Algorithm: Haversine Formula
- Why: If your main focus is on triggering reminders when the user is near a task location (e.g., within a certain radius), the Haversine formula is the best choice. It is simple, fast, and efficient for calculating the straight-line distance between two geographical points using latitude and longitude. It doesn't require complex map data, making it lightweight and ideal for proximity alerts.
- Best Use Case: When the user is within a predefined distance (e.g., 100 metres) of a task location, trigger a reminder.

2. Route Optimization and Task Navigation (Best for Pathfinding)

- Algorithm: A* (A-Star) Algorithm
- Why: If your project involves finding and showing the optimal route from the user's current location to the task location (e.g., for navigation or walking/driving instructions), A* is the most suitable algorithm. It efficiently finds the shortest path while considering real-world factors like road networks and obstacles, especially when used with OpenStreetMap.
- Best Use Case: When users need directions to a task location or when route optimization is critical.

5. Technologies Used

1. Frontend (React Native with Expo):

- **React Native:** For building the mobile interface (cross-platform) and interacting with the device's location services (GPS).
- **Expo:** To simplify access to device hardware like GPS, notifications, and background tasks.
    - **Expo Location API:** To track the user's current location and update it in real-time.
    - **Expo Notifications API:** For sending notifications (text or voice) when the user approaches a task location.

2. Backend (Python):

- Python will handle business logic, task management, and calculations (like proximity or routes).
- Relevant Python libraries for various tasks:
    - **Flask/Django (optional):** To create a RESTful API backend that interacts with the React Native app for managing tasks and user data.
    - **Geopy:** For distance calculations using the Haversine formula (to determine how close the user is to a task location).
    - **NetworkX and osmnx:** If you're using OpenStreetMap for routing or pathfinding (e.g., using the A* algorithm for navigation).
    - **Pyttsx3:** For handling voice reminders (text-to-speech).

3. Mapping (OpenStreetMap):

- **OpenStreetMap:** Provides free map data. You can use it with React Native through third-party libraries.
- **React Native Mapbox SDK:** If you choose to integrate Mapbox (which has a free tier) for advanced mapping and visualisation.

- **osmnx:** Python library to interface with OpenStreetMap for graph data and routing.

4. Python Libraries:

- **osmnx:** To retrieve OpenStreetMap data for routing and geographical analysis.
- **Geopy:** For geolocation and distance calculations.
- **Pyttsx3:** For generating voice reminders in Python (text-to-speech).
- **Flask/Django (optional):** To build a backend API that the mobile app can communicate with for managing tasks.

6. Evaluation Methods

- **Accuracy**: Measures how often the system triggers reminders correctly.
- **Precision & Recall:** Used to evaluate how well the system identifies when a reminder should be triggered.
- **MSE (Mean Squared Error):** Evaluates how well the system predicts user proximity to tasks.
- **Confusion Matrix:** Visualizes results, showing true/false positives and negatives.

7. Deployment

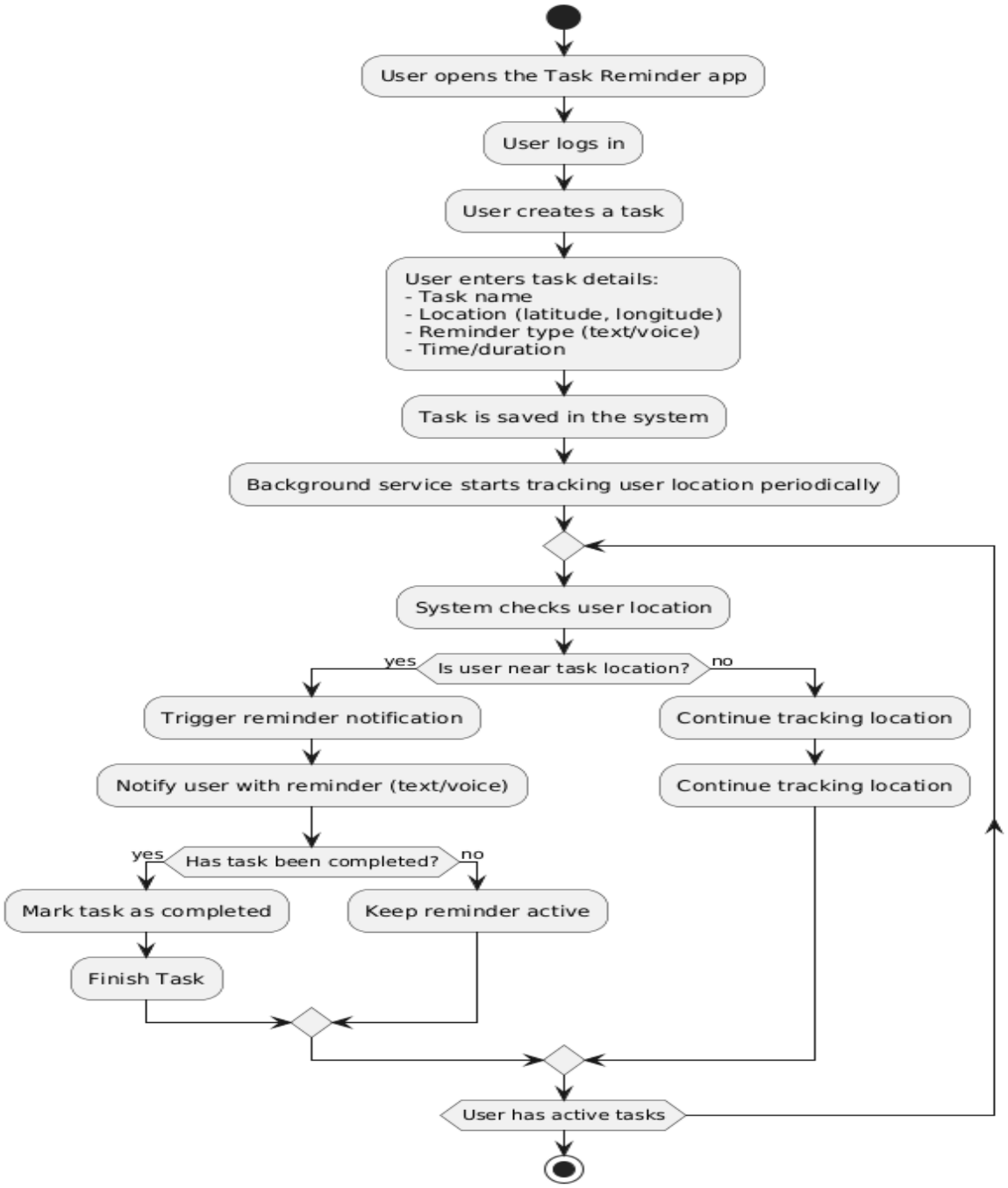
1. Local Deployment:

- **Description:** The system can run on a user's mobile device or local server, allowing offline functionality.
- **Technologies:** SQLite (local database), native GPS tracking services.
- **Benefits:** Operates without internet, leveraging device GPS and local storage.

System of Design

1.Activity Diagram:

An Activity Diagram represents the workflow of a system. It visualizes the sequence of activities or steps that occur in a process. These diagrams help in understanding how various activities are coordinated to complete a process and where decision points exist**.**



1. **Start:**

   The diagram begins with the user opening the Task Reminder app and logging in.

2. **Creating a Task:**
   - The user creates a new task by entering necessary details such as the task name, location (latitude and longitude), reminder type (text/voice), and the time/duration for the reminder.
   - Once the task details are entered, the task is saved in the system.
3. **Background Service:**
   - A background service starts to track the user's location periodically. This ensures that the system can monitor the user's proximity to task locations.
4. **Location Check:**
   - The system checks the user's current location.
   - If the user is near the task location:
     - A reminder notification is triggered.
     - The user is notified with the reminder (either text or voice based on their selection).
5. **Task Completion Check:**
   - After the reminder is sent, the system checks if the user has completed the task.
   - If the task is completed, it marks the task as completed and disables the reminder.
   - If the task is not completed, the reminder remains active.
6. **Continue Tracking:**
   - If the user is not near the task location, the system continues to track the user's location.
7. **Repeat Process:**

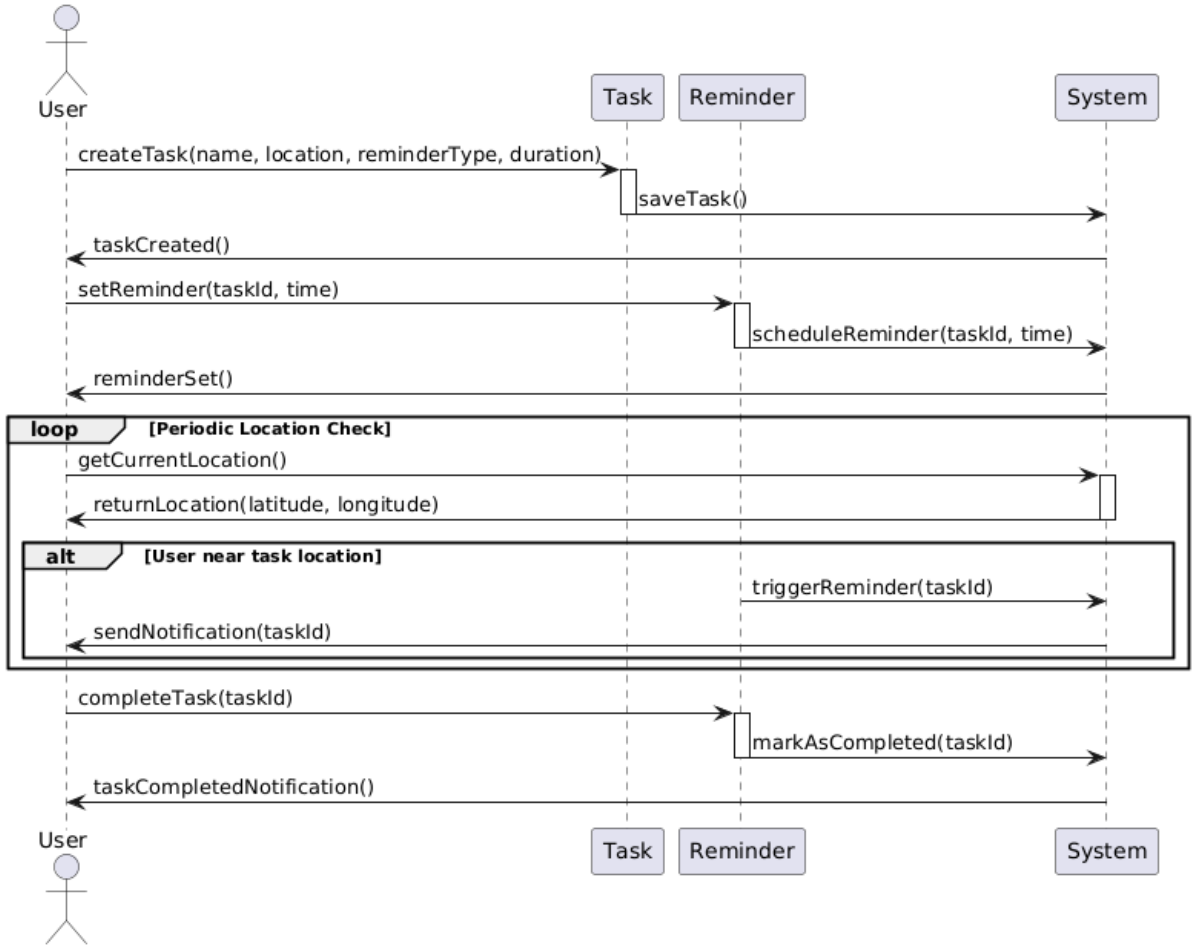
- This process repeats while the user has active tasks.

8. **End:**

   The flow stops when the user completes all tasks or exits the application.

2.Sequence Diagram:

A Sequence Diagram shows how objects in a system interact with each other over time to complete a particular scenario or use case. It represents the order of messages exchanged between various objects and their time-based flow.



Actors and Participants:

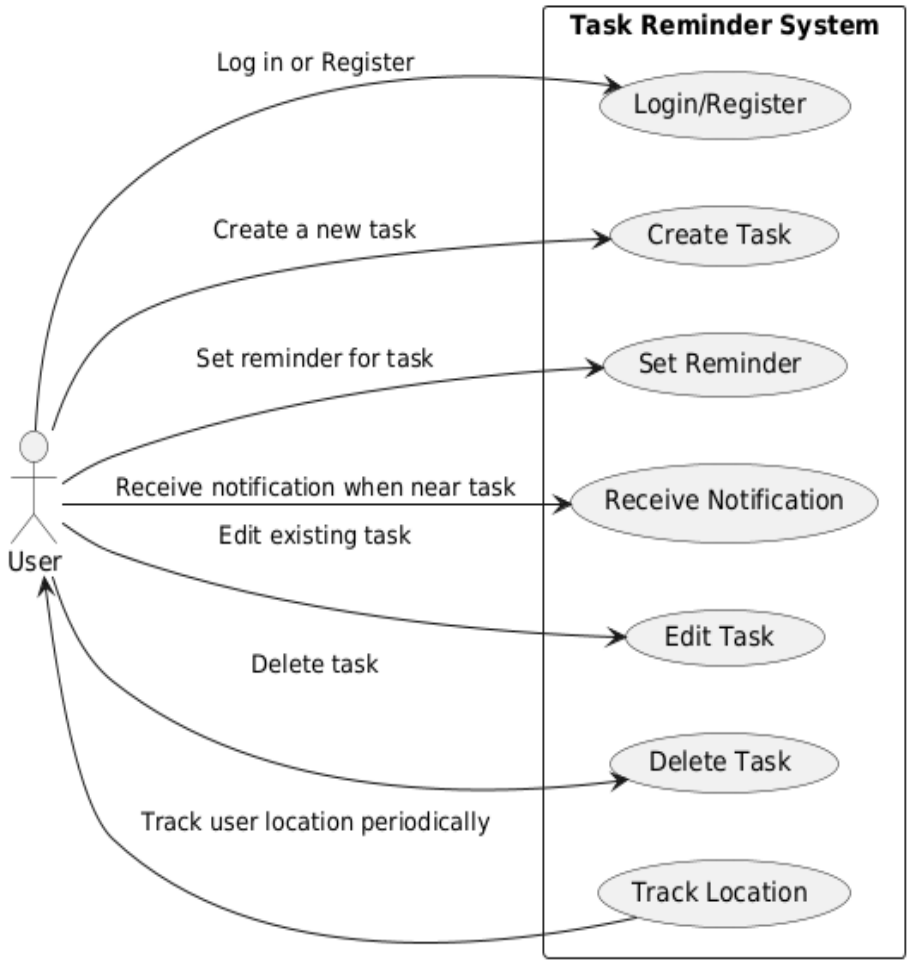
- **User**: The primary actor who interacts with the system.
- **Task**: Represents the functionality related to task creation and management.
- **Reminder**: Manages the reminders for tasks.
- **System**: The backend system responsible for handling task and reminder data.

Flow of Interactions:

- The user directly creates a new task by providing details such as the task name, location, reminder type, and duration.
- The task is saved in the system, and the user receives a confirmation that the task has been created.
- The user sets a reminder for the created task, which the system schedules, and the user is notified that the reminder is set.
- The system enters a loop where it periodically checks the user's current location.
- If the user is near the task location, the reminder triggers, and the system sends a notification to the user.
- When the user completes the task, they notify the reminder, which marks the task as completed in the system. The system then sends a notification to the user indicating that the task has been completed.

3.Use Case Diagram:

A Use Case Diagram describes the functional requirements of a system by showing interactions between users (actors) and the system itself. It focuses on what the system does and who interacts with it, not how it is implemented



Actor:

- The main actor in this system is the User, who interacts with the Task Reminder System.

Use Cases:

- **Login/Register:** The user can log in or register to access the system.
- **Create Task:** The user can create a new task with details like name, location, and reminder type.
- **Set Reminder:** The user can set a reminder for the created task based on location.
- **Receive Notification:** The system sends notifications to the user when they are near the task location.

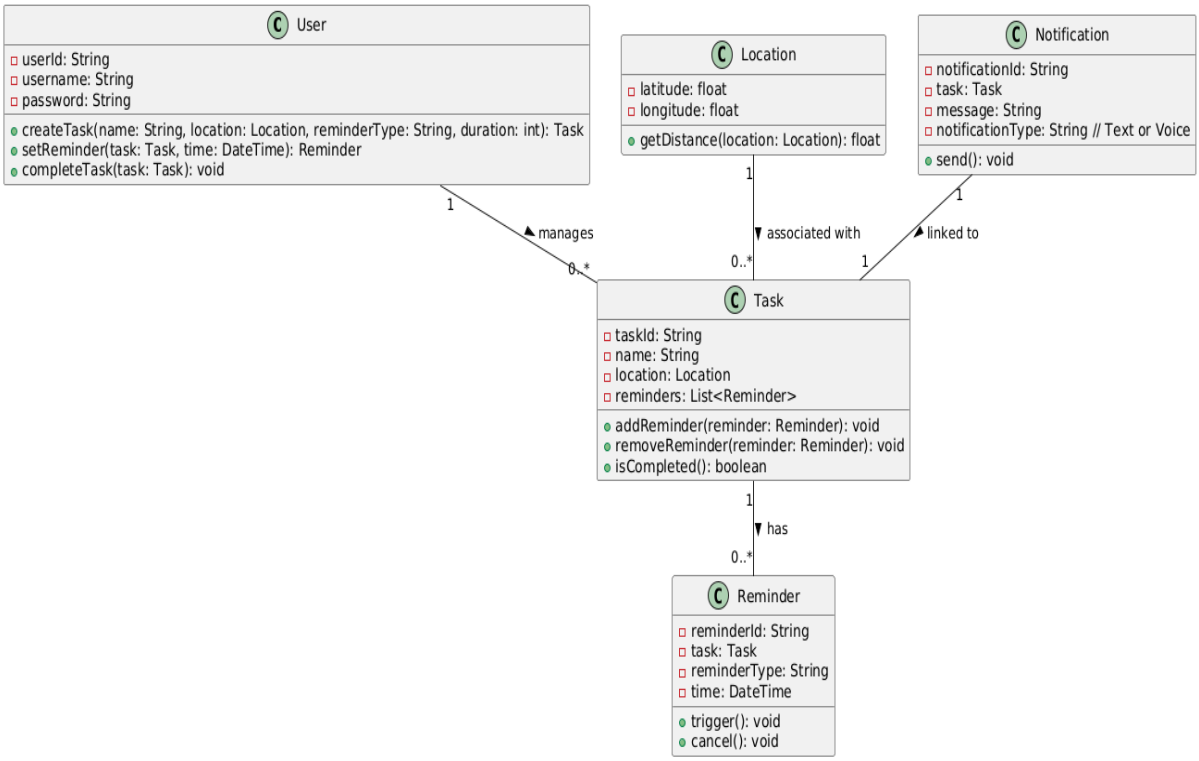
- **Edit Task:** The user can edit the details of an existing task.
- **Delete Task:** The user can delete a task that is no longer needed.
- **Track Location:** The system continuously tracks the user's location to determine when to trigger reminders.

Connections:

- Solid arrows indicate the interactions between the user and the various use cases within the system.

4.Class Diagram:

A Class Diagram is a type of static structure diagram in Unified Modelling Language (UML) that describes the structure of a system by showing its classes, attributes, operations (methods), and the relationships between the classes. It is widely used in object-oriented design and development to model the blueprint of the system's software components.

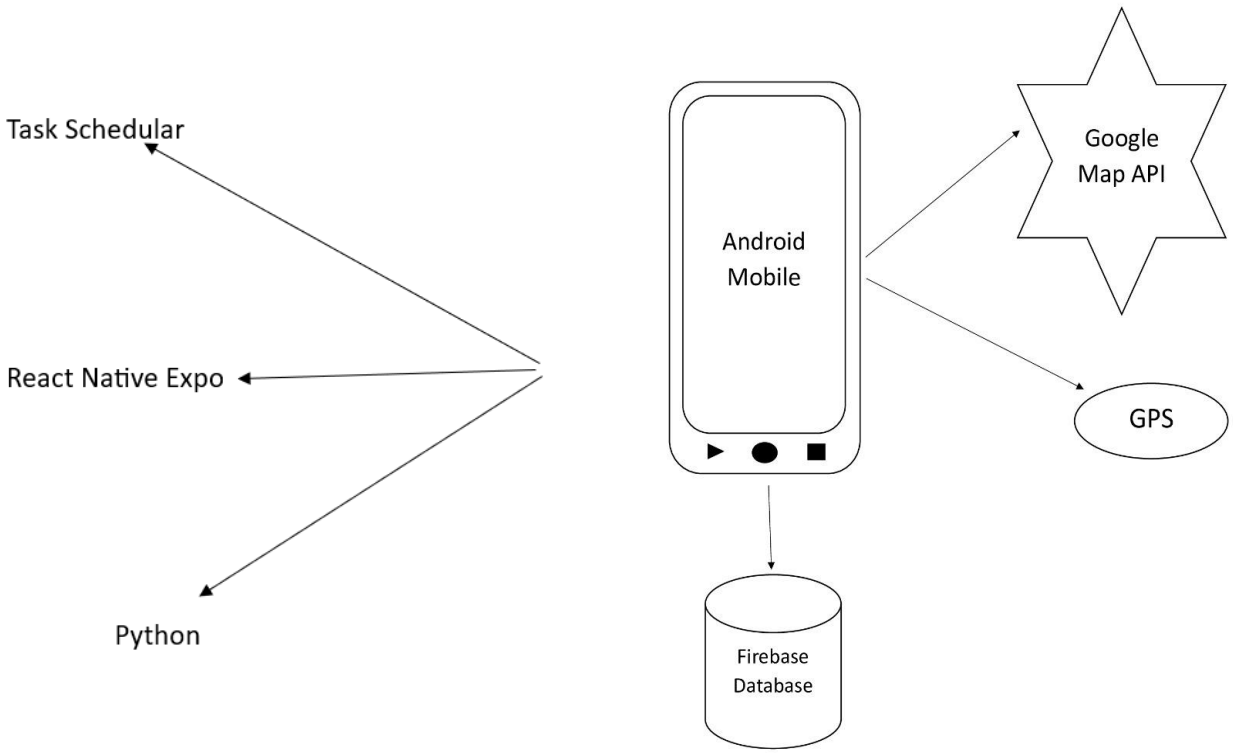


Relationships:

- A **User** manages multiple **Tasks** (1 to many).
- A **Task** can have multiple **Reminders** (1 to many).
- A **Location** can be associated with multiple **Tasks** (1 to many).
- A **Notification** is linked to one **Task** (1 to 1)

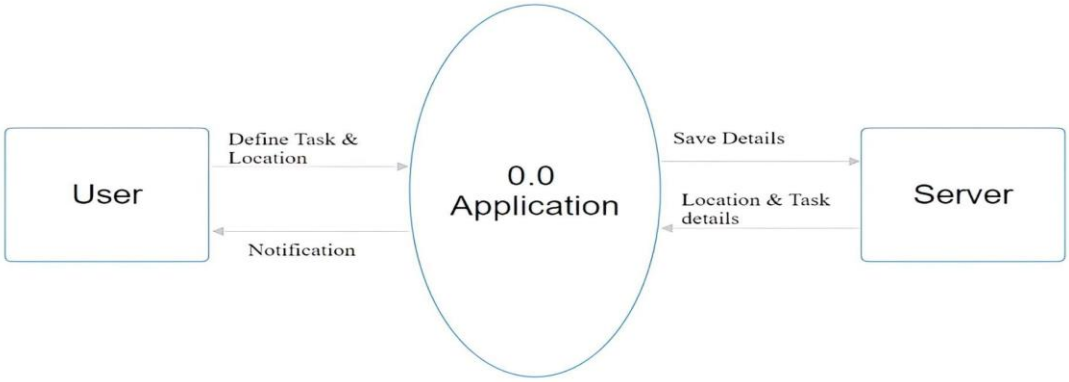
5.Architecture of System:

This architecture outlines a robust system for managing location-based reminders using a combination of front-end (React Native), back-end (Python, Firebase), and external services (Google Maps, GPS).



6.Flow Diagram:

The **User** interacts with the **Application** to define tasks, which the **Application** sends to the **Server** for storage. The **Server** then provides the location and task data to the **Application**, which notifies the user when necessary.

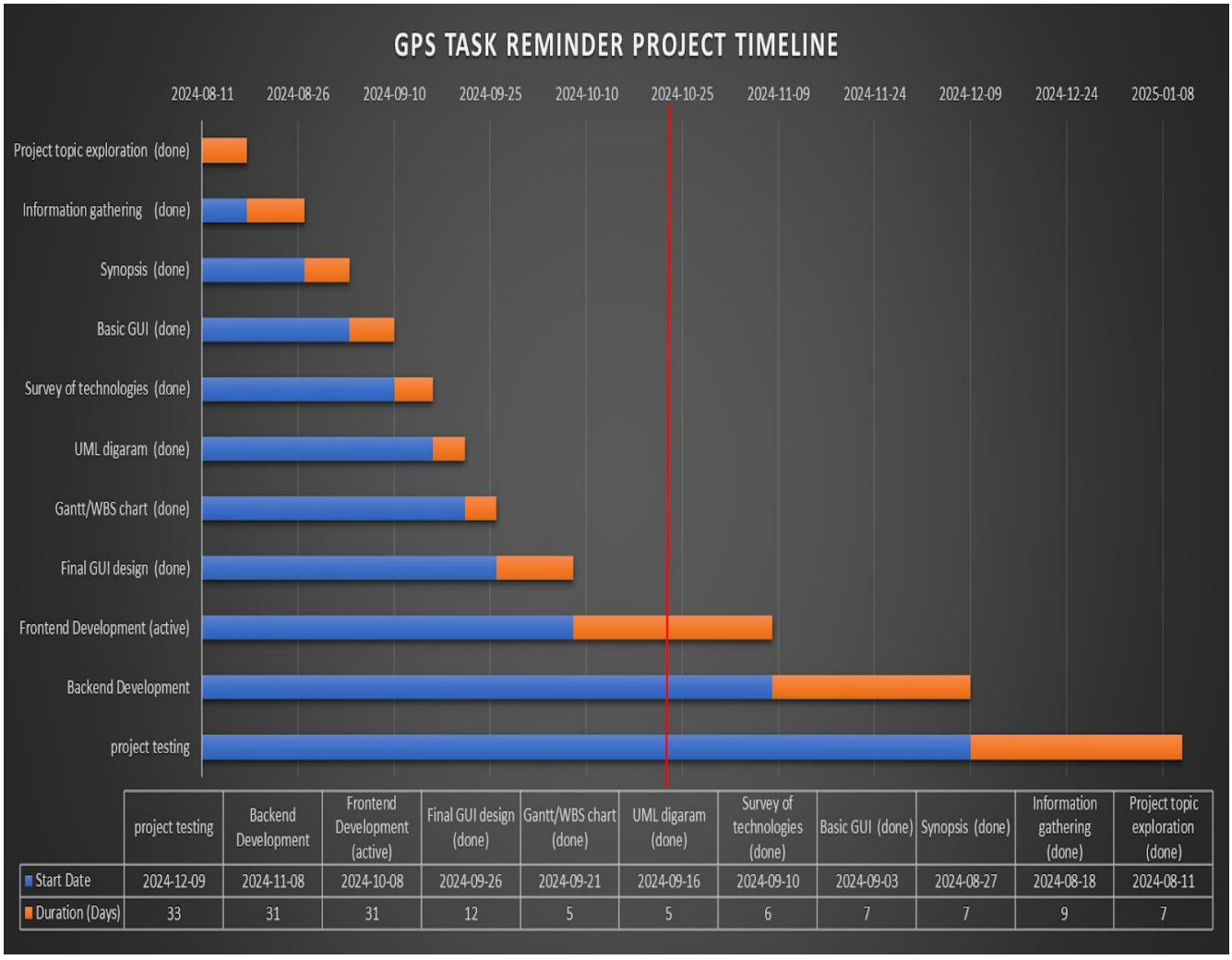


7.Gantt Chart:

Sr. No.	Task	Start Date	Duration (days)	End Date
1.	Project topic exploration(done)	2024-08-11	7	2024-08-17
2.	Information gathering (done)	2024-08-18	9	2024-08-26
3.	Synopsis (done)	2024-08-27	7	2024-09-02
4.	Basic GUI (done)	2024-09-03	7	2024-09-09
5.	Survey of technologies(done)	2024-09-10	6	2024-09-15
6.	UML diagram (done)	2024-09-16	5	2024-09-20
7.	Gantt/WBS chart(done)	2024-09-21	5	2024-09-25
8.	Final GUI design(done)	2024-09-26	12	2024-10-07
9.	Frontend Development(active)	2024-10-08	31	2024-11-07

10.	Backend Development	2024-11-08	31	2024-12-08
11.	Project testing	2024-12-09	33	2025-01-11

A Gantt chart is an essential tool for project managers to organize, schedule, and track the progress of a project visually. It allows for clear planning and helps ensure that all tasks are completed on time and in the correct order.



Conclusion

The project leverages a combination of React Native Expo, Python, and OpenStreetMap to create a fully functional GPS-based task reminder system. The evolution process from requirement gathering to deployment is systematic and relies on open-source technologies and efficient algorithms for geolocation. Future improvements include advanced routing algorithms and enhanced notification systems for a better user experience.

Reference

- Python library text-to-speech: https://www.geeksforgeeks.org/python-text-to-speech-by-using-pyttsx3/
- Python Library osmnx: https://www.geeksforgeeks.org/find-shortest-path-using-python-osmnx-routing-module/
- Expo Documentation: https://docs.expo.dev/
- Flask: https://flask.palletsprojects.com/en/3.0.x/
- Open Street Map: https://www.openstreetmap.org/
- OSMnx Documentation: https://osmnx.readthedocs.io/en/stable/
- Firebase Documentation: https://console.firebase.google.com/
- ER diagram: https://www.smartdraw.com/