T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CAR SHARING MANAGEMENT SYSTEM

AHMET FURKAN EKINCI, MAHMUT ESAT AKSU

SUPERVISOR
ASST. PROF. DR. BURCU YILMAZ

GEBZE
2025

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

CAR SHARING MANAGEMENT SYSTEM

AHMET FURKAN EKINCI, MAHMUT ESAT AKSU

SUPERVISOR
ASST. PROF. DR. BURCU YILMAZ

2025
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 02/10/2024 by the following jury.

JURY

Member
(Supervisor) : Asst. Prof. Dr. Burcu Yılmaz

Member : Assoc. Prof. Dr. Habil Kalkan

Member : PhD Başak Buluz

ABSTRACT

This study presents the development of a car-sharing mobile application designed for Gebze Technical University, leveraging modern technologies such as ASP.NET Core for backend services and Flutter for the mobile interface. The application aims to facilitate efficient and sustainable transportation among university members by enabling users to share rides seamlessly. Users can create posts indicating their starting point and destination on an interactive map, whether they have a vehicle or not. They can also browse posts from others, send join requests to car owners, and communicate through an integrated messaging system to ask questions or coordinate details.

To enhance user experience, the application incorporates a notification mechanism that alerts users when they receive messages or ride requests. Furthermore, a secure login process is implemented using Gebze Technical University's OAuth2 authentication system, allowing users to log in with their institutional email and credentials. By combining innovative features with a user-friendly design, this project provides a convenient and secure platform for ride-sharing, promoting collaboration and reducing environmental impact within the university community.

ÖZET

Bu çalışma, Gebze Teknik Üniversitesi için tasarlanmış, arka uç hizmetler için ASP.NET Core ve mobil kullanıcı arayüzü için Flutter gibi modern teknolojilerden faydalanılarak geliştirilmiş bir araç paylaşım uygulamasını sunmaktadır. Kullanıcılar, araçları olsun veya olmasın, başlangıç noktalarını ve varış noktalarını etkileşimli bir haritada belirterek gönderiler oluşturabilirler. Ayrıca, başkalarının gönderilerine göz atabilir, araç sahiplerine katılma istekleri gönderebilir ve soru sormak veya ayrıntıları koordine etmek için entegre bir mesajlaşma sistemi aracılığıyla iletişim kurabilirler.

Kullanıcı deneyimini geliştirmek için uygulama, kullanıcılar mesaj veya yolculuk isteği aldıklarında onları uyaran bir bildirim mekanizması içerir. Ayrıca, kullanıcıların kurumsal e-postaları ve kimlik bilgileriyle oturum açmalarına olanak tanıyan Gebze Teknik Üniversitesi'nin OAuth2 kimlik doğrulama sistemi kullanılarak güvenli bir oturum açma süreci uygulanır. Yenilikçi özellikleri kullanıcı dostu bir tasarımla birleştirerek, bu proje, üniversite topluluğu içinde iş birliğini teşvik eden ve çevresel etkiyi azaltan kullanışlı ve güvenli bir yolculuk paylaşımı platformu sağlar.

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our supervisor, Asst. Prof. Dr. Burcu Yılmaz, for his invaluable guidance, patience, and encouragement throughout this project.

Ahmet Furkan Ekinci and Mahmut Esat Aksu

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Transportation plays a vital role in the daily lives of students and staff at Gebze Technical University (GTU). However, the lack of a structured car-sharing system has resulted in inefficiencies such as increased individual car usage, traffic congestion, high transportation costs, and environmental pollution. This report outlines the development of a car-sharing mobile application tailored specifically for the GTU community.

The application combines the following features to create a seamless user experience:

1. **User Authentication:** Users log in securely using GTU's OAuth2 authentication system, ensuring that only verified university members can access the platform. This creates a trusted environment where users can confidently interact with others.

2. **Post Creation for Ride-Sharing:** Users can create ride-sharing posts by selecting their starting and destination points on an interactive map powered by Google Maps API. They can also specify details such as the number of available seats, time, and whether they are offering or seeking a ride.

3. **Browsing and Filtering Posts:** Users can browse ride-sharing posts created by others and apply filters based on criteria such as departure time, location, or vehicle availability. This allows users to quickly find rides that meet their needs.

4. **Real-Time Notifications:** The application uses Firebase notifications to inform users about:

   - New ride requests.
   - Messages from other users.
   - Updates on their posts or requests.

   These notifications ensure that users stay engaged and up-to-date without manually checking the application.

5. **Messaging System:** A built-in messaging system allows users to communicate directly to coordinate ride details, ask questions, or establish trust with potential ride partners.

6. **Request Management:** Users can send ride requests to post creators and receive responses. Post creators can accept or reject requests, and the status is updated in real time. Both parties are notified of any updates through the app.

7. **Sustainability Points System:** The application tracks users' participation in ride-sharing activities and rewards them with sustainability points. These points encourage eco-friendly behavior and may be utilized in future university reward systems, such as discounts or priority access to campus services.

8. **Cross-Platform Availability:** Built with Flutter, the mobile application supports both Android and iOS devices, ensuring accessibility for the entire university community.

The Purpose of the Project section discusses the motivations behind the project and the goals it aims to achieve. The Scope of the Project defines the technical and functional boundaries, detailing the key features and limitations of the system. Finally, the Problem Definition highlights the current transportation challenges faced by the GTU community and explains how this project addresses these issues to create a sustainable and collaborative solution.

1.1. Purpose of the Project

The primary motivation behind this project is to address the lack of an efficient car-sharing system within the Gebze Technical University (GTU) community. Rising individual car usage has led to issues such as traffic congestion, high transportation costs, and increased environmental pollution. This project aims to provide a solution by developing a user-friendly mobile application that facilitates seamless ride-sharing among students and staff. Beyond solving transportation challenges, the system introduces a Sustainability Point mechanism, which rewards users for participating in environmentally friendly actions, such as sharing rides. These points could form the foundation for future systems within GTU, enabling users to redeem them for campus services or benefits. The ultimate goal is to promote sustainable transportation practices, reduce the environmental impact of commuting, and foster collaboration within the university community.

1.1.1. Advantages for Users

This car-sharing project offers significant benefits to GTU students and staff by addressing their transportation needs in a cost-effective and efficient manner. Key user benefits include

- **Cost savings:** By sharing rides, users can significantly reduce their daily transportation expenses.

- **Convenience:** The platform provides an easy-to-use system for creating and joining ride posts, streamlining the process compared to informal arrangements through social networks.
- **Incentives through sustainability points:** Users earn points for participating in ride-sharing activities, encouraging continuous engagement with the platform. These points can potentially be used for services within the GTU in the future, such as discounts on campus facilities, priority access to resources, or other community benefits.
- **Safety and reliability:** With features such as real-time notifications, secure login through the GTU OAuth2 system, and a verified user base, the application ensures a safe and reliable environment for ridesharing.

1.1.2. Environmental Impacts

The project plays a critical role in promoting environmental sustainability by reducing reliance on individual vehicles and encouraging shared mobility. The application helps lower fuel consumption and reduce carbon emissions, directly contributing to a cleaner, healthier environment. Furthermore, the Sustainability Point System incentivizes users to adopt eco-friendly practices, reinforcing a culture of environmental responsibility within the GTU community. By making sustainable actions measurable and rewarding, the platform not only addresses immediate transportation issues but also instills long-term environmentally conscious habits among its users.

1.1.3. Community Collaboration

The application strengthens community bonds by fostering collaboration and interaction between GTU students and staff. Key aspects of this collaboration include the following.

- **Enhanced Social Bonds:** By coordinating shared rides, users interact with individuals from different departments and backgrounds, creating new opportunities for social connections.
- **Incentivized Engagement:** The Sustainability Point System encourages users to participate more actively in ride-sharing, creating a shared sense of purpose within the community.
- **Messaging and Trust:** The built-in messaging system allows users to communicate ride details and establish trust, improving the overall user experience and

ensuring smooth coordination.

This collaborative approach not only addresses transportation challenges but also enhances the university's social and environmental culture. In the future, the points earned through the system could be extended to encourage broader community participation, such as earning discounts or privileges in campus services.

1.2. Scope of the Project

The scope of this project defines the key features included in the car-sharing mobile application designed for Gebze Technical University (GTU). These features focus on providing a functional, user-friendly, and efficient system tailored to the transportation needs of GTU students and staff. By outlining the core components, this section highlights how the project aims to address the identified problems and achieve its objectives.

1.2.1. Features Included in the Scope

The car-sharing mobile application includes a range of features that ensure a seamless and efficient user experience. These features are designed to meet the transportation needs of GTU students and staff while promoting sustainability and community engagement. The primary features are:

- **Post Creation:** Users can easily create ride-sharing posts by selecting their starting and destination points on an interactive map.
- **Map Integration:** The application is powered by Google Maps services, enabling users to visually select routes and destinations. This feature ensures intuitive navigation and helps users find or offer rides efficiently.
- **Notification System:** Real-time notifications powered by Firebase alert users about ride requests and messages. This ensures users stay informed and engaged with the platform.
- **Messaging System:** The application includes a secure, built-in messaging system that allows users to communicate with one another about ride details. This fosters trust and coordination between users.
- **Secure Login:** GTU's OAuth2 authentication mechanism is integrated to ensure that only verified university members can access the platform. This guarantees a trusted and safe user environment.

- **Sustainability Points System:** To encourage environmentally friendly transportation habits, the application tracks user participation in ride-sharing and rewards them with sustainability points. These points could be utilized in future GTU systems, such as discounts or campus services.

These features form the backbone of the car-sharing application and address the primary goals of providing cost-effective, eco-friendly, and community-focused transportation solutions for GTU.

1.3. Problem Definition

The car-sharing mobile application project was conceived to address several pressing challenges faced by the Gebze Technical University (GTU) community. This section provides a detailed analysis of the current problems, the specific issues targeted by the project, and the solutions proposed to mitigate these challenges. By resolving these problems, the project aims to create a sustainable, cost-effective, and collaborative transportation system for GTU.

1.3.1. Current Shortcomings

The current transportation practices within GTU highlight significant inefficiencies due to the absence of a structured car-sharing system. These shortcomings include:

- **Lack of a Dedicated Platform:** At present, GTU students and staff rely on informal arrangements (e.g., social media groups) to coordinate shared rides. This leads to difficulty in finding available rides or passengers, lack of reliability and trust, and inefficient coordination.
- **Individual Car Dependency:** The absence of a structured ride-sharing system has resulted in increased dependency on personal vehicles, leading to higher financial costs and traffic congestion.
- **Missed Opportunities for Community Interaction:** Without a platform to encourage collaboration, there is limited interaction between students and staff from different departments, reducing the sense of community within GTU.

1.3.2. Targeted Issues

The project specifically aims to address the following problems:

- **Traffic Congestion:** Heavy traffic during peak hours caused by individual car usage.
- **High Transportation Costs:** Significant financial burden on students and staff due to personal vehicle usage.
- **Environmental Pollution:** Increased carbon emissions and air pollution due to personal vehicle reliance.
- **Inefficient Communication:** Informal ride-sharing arrangements lead to miscommunication and missed opportunities.

1.3.3. Proposed Solution

To address these challenges, the car-sharing mobile application introduces an innovative and user-friendly solution that integrates technology to promote ride-sharing. The proposed features and functionalities aim to resolve the identified issues in the following ways:

1. **Centralized Platform for Ride-Sharing:** The application provides a dedicated platform where users can create and join ride-sharing posts. By integrating Google Maps, users can visually select routes, making the process intuitive and efficient.
2. **Collaborative Ride-Sharing Experience:** The system facilitates ride-sharing among users, encouraging collaboration and community interaction. This enables students and staff to share rides based on mutual convenience, reducing the need for individual car usage.
3. **Environmental Benefits through Sustainability Points:** The application tracks user participation in ride-sharing and rewards them with sustainability points, incentivizing eco-friendly behavior. Reducing individual car usage directly lowers carbon emissions and contributes to GTU's environmental goals.
4. **Enhanced Communication and Coordination:** A built-in messaging system ensures smooth communication between users, eliminating the inefficiencies associated with informal arrangements. Real-time notifications keep users informed about ride requests and updates.
5. **Secure and Reliable System:** By integrating GTU's OAuth2 authentication system, the application guarantees that only verified university members can access the platform. This builds trust and ensures a safe user environment.

2. SYSTEM DESIGN AND DEVELOPMENT

The System Design and Development section explains the technical foundation of the car-sharing application, including the system architecture, functional diagrams, and structural design. This section highlights how the application components interact and how they were designed to meet the project's objectives.

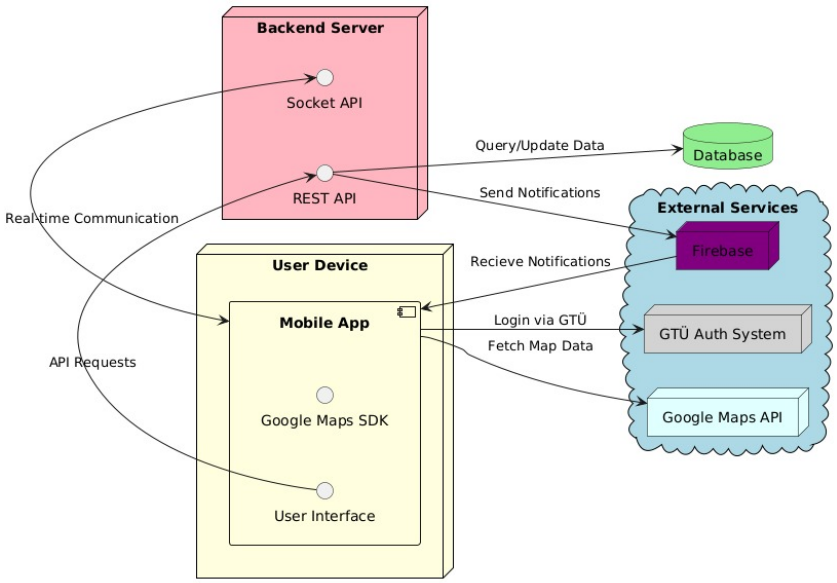

Figure 2.1: System Design Diagram

2.1. System Architecture

The system architecture is based on a client-server model that separates the backend and frontend functionalities, ensuring scalability, security, and maintainability. The application integrates various technologies and services to deliver a seamless and efficient experience. The architecture consists of the following components:

2.1.1. Backend (Server-Side)

- **Technology Used:** Developed using **ASP.NET Core**, the backend handles critical functionalities such as:
  - User authentication through **Gebze Technical University's OAuth2 sys-**

tem, ensuring only verified university members can access the platform.

  - Management of ride posts, messaging, and tracking of sustainability points.
  - Secure and efficient communication with the database to store and retrieve user data, ride details, and message histories.

- **APIs:** The backend exposes **RESTful APIs** to the mobile application, enabling seamless communication between the server and the client.

2.1.2. Frontend (Client-Side)

- **Technology Used:** Built using **Flutter**, the mobile application provides a user-friendly and responsive interface for:
  - Creating and browsing ride posts.
  - Messaging other users and receiving notifications.
  - Viewing real-time updates on ride-related activities.
- **Cross-Platform Compatibility:** Flutter's cross-platform capability ensures the application runs smoothly on both Android and iOS devices.

2.1.3. Database

- **Database Design:** A relational database is implemented to store critical data securely, including:
  - User profiles.
  - Ride posts.
  - Messages and notifications.
  - Sustainability point records.
- **Efficiency:** The database is designed for optimized query performance, ensuring data integrity and seamless operation.

2.1.4. Third-Party Services

- **GTU OAuth2 System:** Integrated for secure user authentication, allowing students and staff to log in with their university credentials. This ensures that only verified members of GTU can access the application, creating a safe and trusted user environment.

- **Google Maps Services:** Used for interactive map functionality, allowing users to visually select starting and destination points, view routes, and identify nearby ride opportunities. This feature enhances usability and simplifies ride coordination.
- **Firebase Notification Service:** Provides real-time notifications to keep users informed about:
  - New ride requests.
  - Updates on their posts.
  - Incoming messages.

  This service ensures users stay engaged and do not miss critical updates.

2.2. Use Case Diagram

The Use Case Diagram provides a high-level overview of how users interact with the system and the primary functionalities offered by the application. The main actors in the system are:

- **User (Student/Staff):** Responsible for creating posts, sending ride requests, and messaging other users.

2.2.1. Use Cases

The key use cases of the system include:

- **Create a ride post:** Users can create posts by selecting their starting and destination points and providing ride details.
- **Browse and search ride posts:** Users can view available ride posts and search for rides based on specific criteria.
- **Send and receive ride requests:** Users can request to join a ride or respond to requests from others.
- **Communicate with other users via messaging:** A built-in messaging system allows users to coordinate ride details and establish trust.

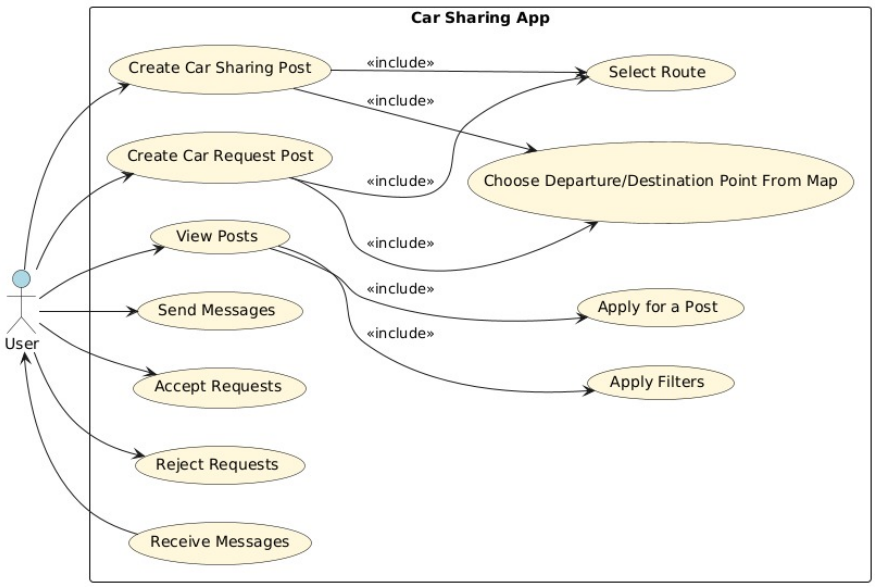

Figure 2.2: Use Case Diagram

2.3. Database Design

The database for the car-sharing application is designed as a **relational database** to ensure the efficient and secure storage of critical data. The design supports the core functionalities of the application, such as user authentication, ride posts, messaging, and sustainability points tracking. It is optimized for query performance, ensuring that the system operates seamlessly under various workloads.

2.3.1. Key Tables and Their Roles

- **User Table:** Stores information about the users of the system, including:
  - **user_id:** Primary key for uniquely identifying users.
  - **unique_id:** A unique identifier linked to GTU's OAuth2 system.
  - **username, email, name, surname:** Basic user information.
  - **sustainability_point:** Tracks points earned for eco-friendly activities.
  - Flags (**is_student, is_academic_personal, is_administrative_staff**) indicate the user type.
- **Journey Table:** Represents ride posts created by users, with columns such as:
  - **journey_id:** Primary key.
  - **has_vehicle:** Indicates whether the user owns a vehicle.

  - **map_id:** Foreign key linking to the **Map** table for route information.
  - **time:** Scheduled time for the journey.
  - **is_one_time:** Indicates if the journey is a one-time or recurring trip.

- **Map Table:** Stores detailed route information:
  - Starting and destination points (**departure_latitude, destination_latitude**).
  - District-level details (**current_district, destination_district**).
  - **map_route:** Optional data for representing the entire route.

- **Request Table:** Tracks ride requests between users:
  - **request_id:** Primary key.
  - **journey_id:** Foreign key linking to the **Journey** table.
  - **sender_id, receiver_id:** IDs of the sender and receiver of the request.
  - **status_id:** Links to the **Status** table to represent request states.

- **Message Table:** Facilitates communication between users:
  - **message_id:** Primary key.
  - **sender_id, receiver_id:** IDs of the participants.
  - **message_text:** Content of the message.
  - **is_read:** Boolean indicating if the message is read.
  - **time:** Timestamp of the message.

- **UserDeviceTokens Table:** Stores device-specific tokens for real-time notifications:
  - **user_device_id:** Primary key.
  - **user_id:** Foreign key linking to the **User** table.
  - **token:** Device token for push notifications.
  - **time:** Timestamp of the token creation.

2.3.2. Design Goals

- **Data Integrity:** Use of foreign keys ensures relationships between tables are consistently maintained.
- **Scalability and Performance:** Indexed columns (e.g., **user_id, journey_id**) ensure fast query performance.

- **Flexibility:** The schema is designed to support future expansions, such as ride ratings or analytics.
- **Real-Time Functionality:** The **UserDeviceTokens** table supports Firebase notifications for real-time updates.

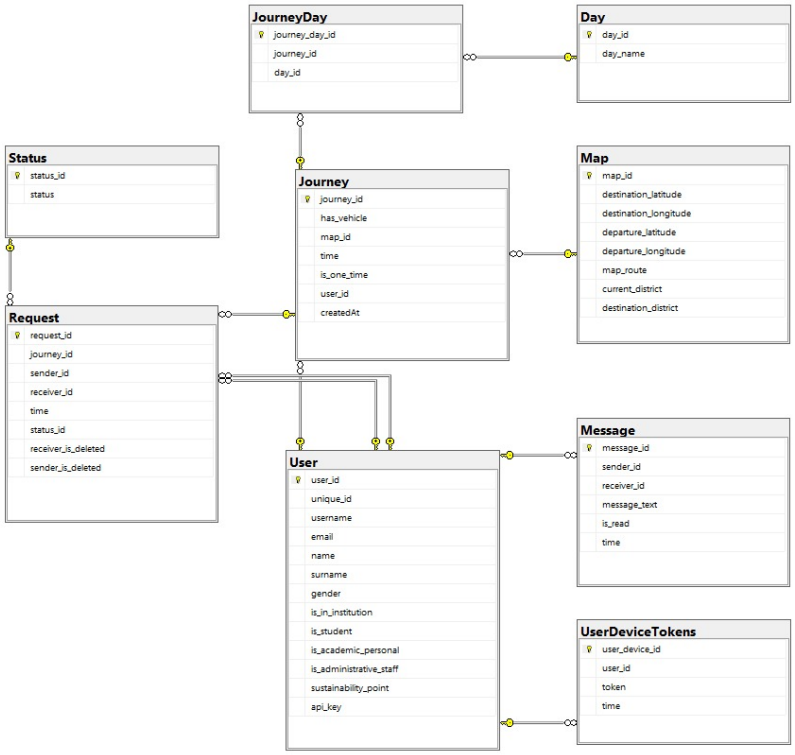

Figure 2.3: Database Design Diagram

2.4. Sequence Diagram

The **Sequence Diagram** illustrates the flow of interactions between the system components during specific use cases. It shows the order of events and how data flows between the client, server, and database.

2.4.1. Key Components

- **User:** Represents the student or staff interacting with the mobile application.
- **Mobile App (UI):** The client-side application built with Flutter, providing the interface for creating posts, browsing rides, and messaging.

- **Backend:** The server-side logic implemented using ASP.NET Core that handles API requests, business logic, and database operations.[1]
- **GTU Auth System:** Verifies user credentials during login to ensure that only university-affiliated users can access the platform..[2]
- **Google Maps API:** Provides location-based services such as route planning and map data retrieval.
- **Firebase:** Enables real-time push notifications to keep users informed about messages and ride requests.[3].[4]
- **Database:** Stores and retrieves user, post, message, and request data.

2.4.2. Example Use Case: Creating a Ride Post

1. **User Action:** The user logs in using their GTU credentials and selects starting and destination points on the map. Additional details such as time and vehicle availability are filled in.
2. **Login Process:**
    - The mobile app sends the user's credentials to the backend.
    - The backend forwards these credentials to the GTU Auth System, which verifies them and returns user data (e.g., name, email).
    - The user data is stored in the database if valid, and the app proceeds to the dashboard.
3. **Post Creation:**
    - The mobile app sends the ride post details (route, time, vehicle availability) to the backend API.
    - The backend validates the data, retrieves the route details using Google Maps API, and saves the post data to the database.
    - Upon successful insertion, a confirmation message is sent back to the mobile app, and the post is displayed on the user's dashboard.

2.4.3. Other Key Interactions

- **Viewing and Filtering Posts:** The user can fetch posts from the backend using filters such as time and location. The backend queries the database for posts matching the filters and returns the results to the mobile app.

- **Sending a Ride Request:** The user selects a post and sends a request to the post owner. The backend validates the request, updates the Request table in the database, and notifies the recipient using Firebase.
- **Messaging:** The user sends a message to another user. The message is stored in the database, and the recipient is notified in real-time via Firebase.
- **Accepting/Rejecting Requests:** The post owner accepts or rejects a ride request. The backend updates the status in the Request table and notifies the sender of the updated status.

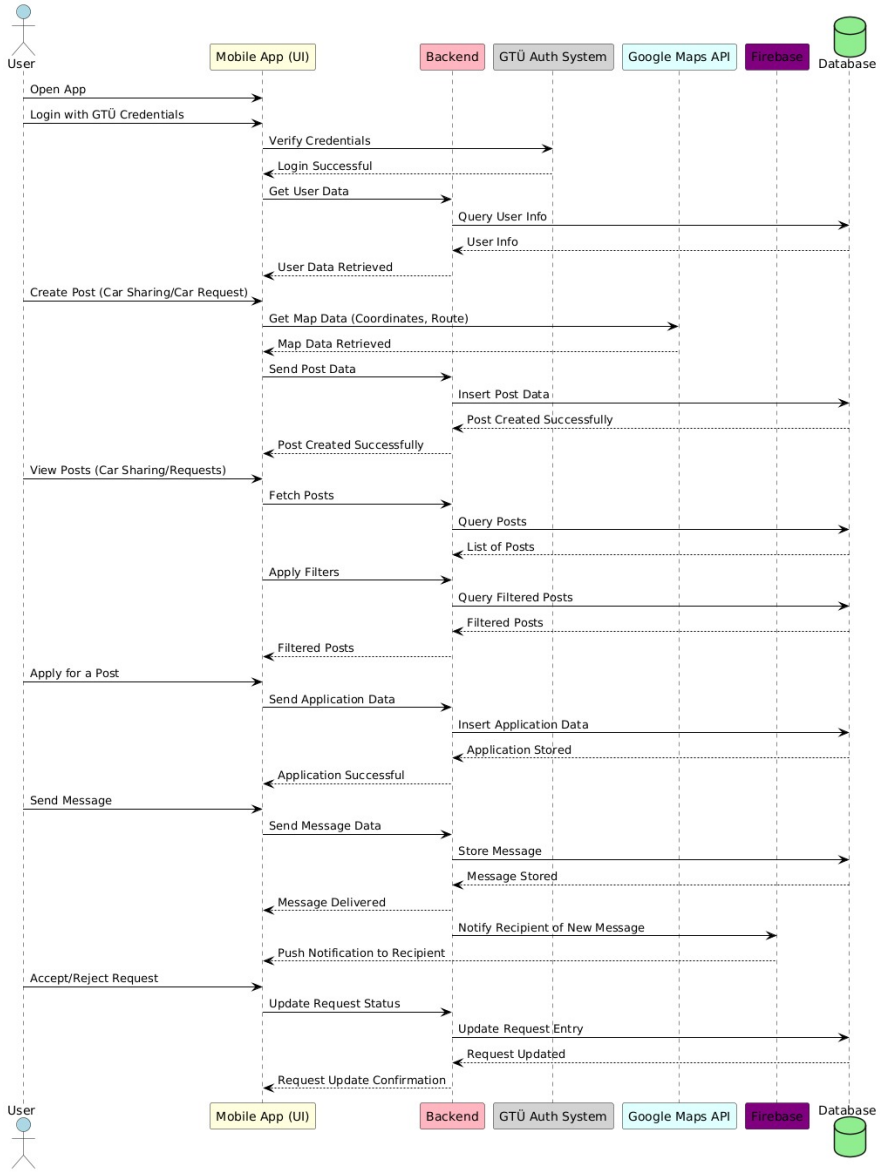

Figure 2.4: Sequence Diagram

2.5. Class Diagram

The Class Diagram defines the structural design of the system by illustrating the relationships between different classes used in both the backend and frontend. It helps visualize the organization of data, the flow of logic, and the interaction between various components of the application. The backend class diagram is particularly crucial as it outlines how the server-side logic interacts with the database and frontend to provide the necessary functionality.

2.5.1. Backend Architecture

The backend of the car-sharing application follows a layered architecture, ensuring modularity, scalability, and maintainability. Each layer has a well-defined responsibility and interacts with other layers through clearly established boundaries. This design promotes separation of concerns, making the system easier to test, extend, and maintain.[1]

The backend is organized into four primary layers: API, Business, Data Access, and Entities. These layers collaborate to handle user requests, process business logic, interact with the database, and provide a seamless experience for the client application.[5]

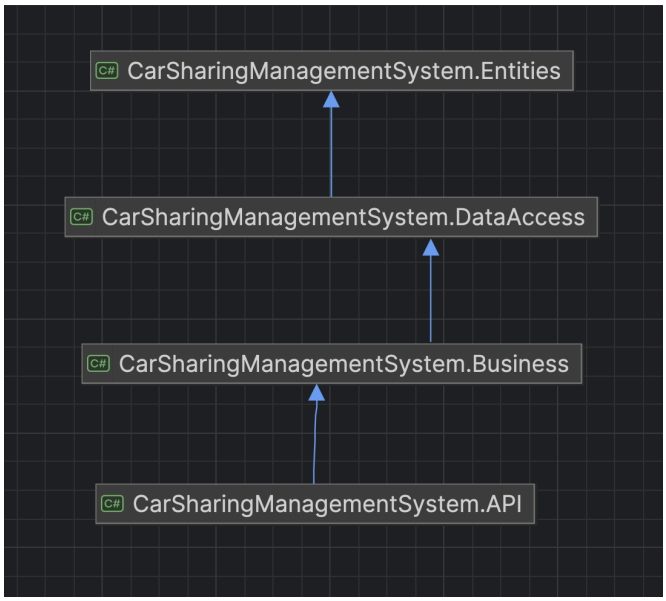

Figure 2.5: Back-end System General Design

2.5.1.1. Layers of the System

The system is designed using a layered architecture to ensure modularity, scalability, and maintainability. Each layer has a specific responsibility, enabling clear separation of concerns and efficient collaboration between components.

2.5.1.2. API Layer

The API layer serves as the entry point for all requests coming from the frontend. It exposes RESTful endpoints and acts as a mediator between the client and the backend logic.

- **Responsibilities:**
  - Handle HTTP requests and map them to appropriate service methods.
  - Validate incoming data to ensure that only valid requests are processed.
  - Return structured responses (e.g., success messages or error codes) to the client.

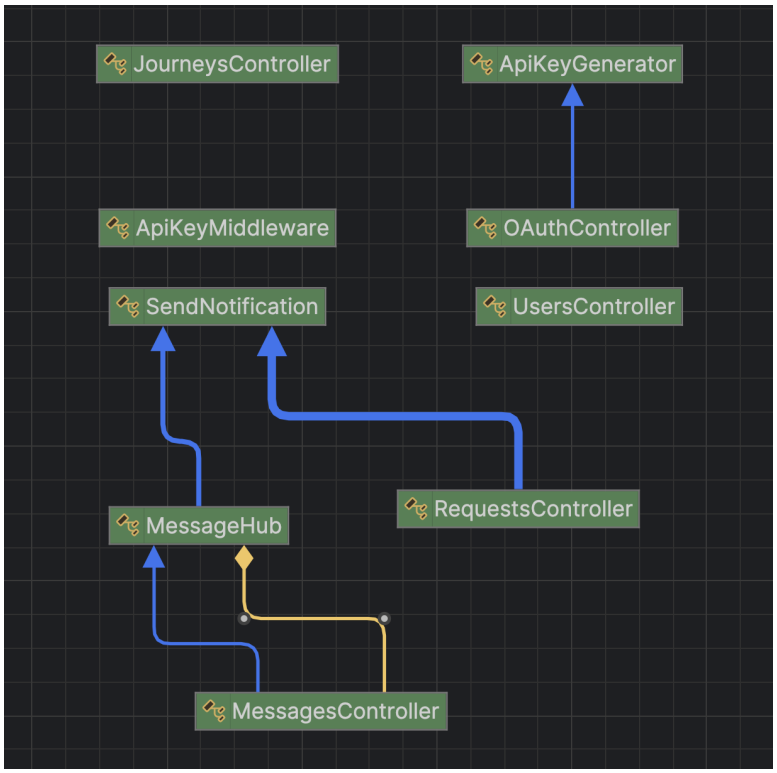

Figure 2.6: Back-end Api Layer Design

2.5.1.3. Business Layer

The business layer implements the core logic of the application. It processes the requests received from the API layer and applies the necessary business rules before interacting with the data layer.

- **Responsibilities:**
  - Encapsulate the business rules and application logic.
  - Coordinate between the API and Data Access layers to ensure consistent data processing.
  - Manage calculations, such as sustainability point allocation for ride-sharing activities.

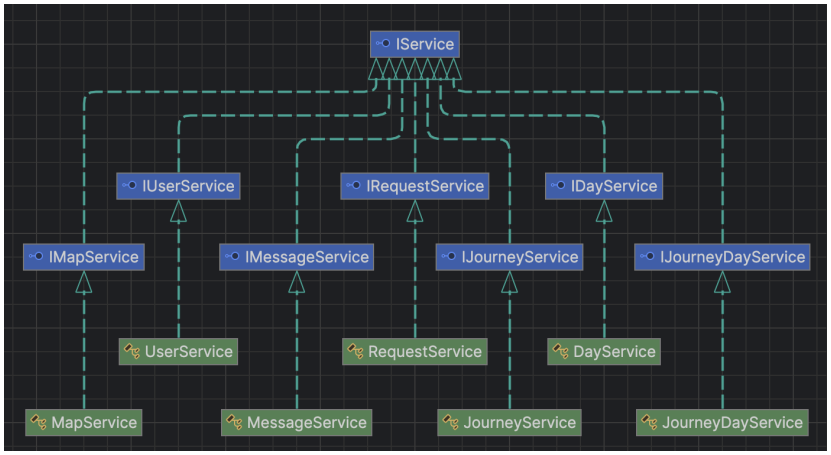

Figure 2.7: Back-end Business Layer Design

2.5.1.4. Data Access Layer

The data access layer serves as an abstraction over the database, handling all interactions with the data storage.

- **Responsibilities:**
  - Provide CRUD (Create, Read, Update, Delete) operations for the database.
  - Abstract database queries, ensuring that the business layer does not directly interact with the database.
  - Maintain data integrity by enforcing database constraints and relationships.

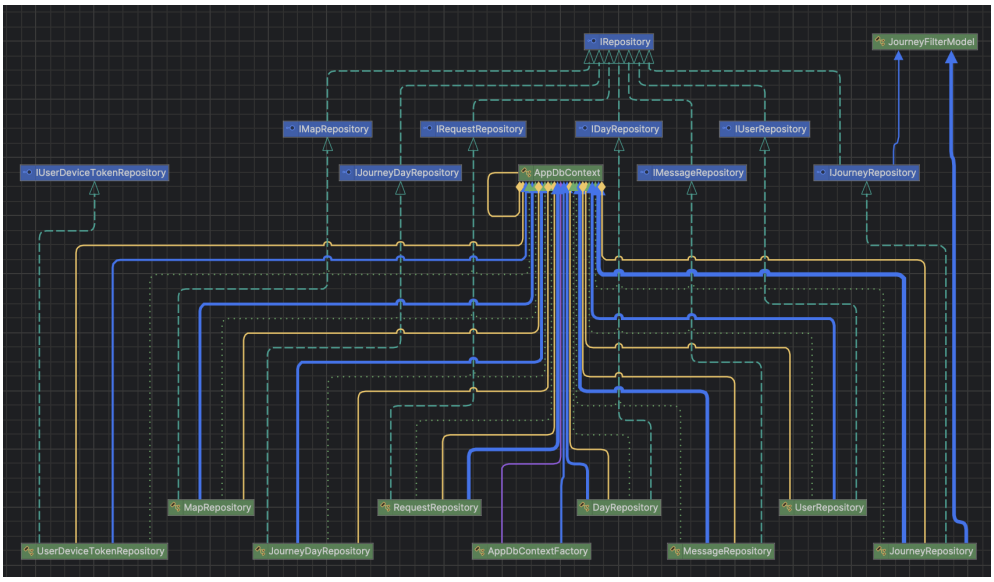

Figure 2.8: Back-end Repository Layer Design

2.5.1.5. Entities Layer

The entities layer defines the data models used throughout the backend.

- **Responsibilities:**
  - Define the structure of the application's core data, such as users, ride posts, and messages.
  - Act as the foundation for database schemas, ensuring consistency between the database and the backend logic.
  - Serve as the data transfer objects between the layers.

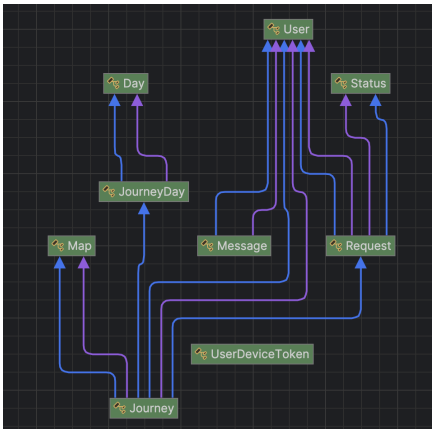

Figure 2.9: Back-end Entities Layer Design

2.5.1.6. NuGet Packages Used in the Backend

This section provides an overview of the NuGet packages used in the backend and their purposes.[6]

2.5.1.6.1 Microsoft Entity Framework Core

- **Microsoft.EntityFrameworkCore** (v8.0.10): Core ORM functionality for data modeling and database access.
- **Microsoft.EntityFrameworkCore.Design** (v8.0.10): Provides design-time tools for migrations and scaffolding.
- **Microsoft.EntityFrameworkCore.SqlServer** (v8.0.10): SQL Server provider for EF Core.
- **Microsoft.EntityFrameworkCore.Tools** (v8.0.10): CLI tools for migrations and database management.

Purpose: Enables database interaction, migrations, and querying in a strongly typed manner using LINQ.

2.5.1.6.2 Microsoft.Extensions.Configuration

- **Microsoft.Extensions.Configuration** (v8.0.0): Provides configuration abstraction for .NET applications.
- **Microsoft.Extensions.Configuration.Json** (v8.0.1): JSON configuration file support.

Purpose: Allows flexible configuration management for the application, including `appsettings.json` file usage.

2.5.1.6.3 AspNetCoreRateLimit

- **AspNetCoreRateLimit** (v5.0.0)

Purpose: Implements rate-limiting for ASP.NET Core APIs to prevent abuse and ensure fair usage of system resources.

2.5.1.6.4 FirebaseAdmin

- **FirebaseAdmin** (v3.1.0)

Purpose: Provides tools to integrate with Firebase services, such as real-time notifications and cloud messaging.

2.5.1.6.5 Google.Apis.Auth

- **Google.Apis.Auth** (v1.68.0)

Purpose: Facilitates Google OAuth2 authentication for integrating Google services or managing secure API access.

2.5.1.6.6 Microsoft.AspNetCore.SignalR

- **Microsoft.AspNetCore.SignalR** (v1.1.0)

Purpose: Enables real-time communication between the server and clients, useful for features like live notifications or messaging.

2.5.1.6.7 Swashbuckle.AspNetCore

- **Swashbuckle.AspNetCore** (v6.4.0)

Purpose: Adds Swagger/OpenAPI support to document and test your ASP.NET Core APIs interactively.

2.5.2. Flutter Class Diagram

The Flutter Class Diagram illustrates the structural design of the frontend application, highlighting the key components and their interactions. The frontend is organized into several layers to ensure a clean and maintainable architecture. Each layer is responsible for a specific aspect of the application, providing both functionality and a seamless user interface.

2.5.2.1. Key Components

2.5.2.1.1 Screens Screens represent the main user interface components where users interact with the application. Each screen corresponds to a specific functionality and is implemented as a Flutter widget.

- **Login Screen:** Manages the user authentication process using GTU's OAuth2 system.
- **Map Screen:** Displays the interactive map, allowing users to select starting and destination points.
- **Create Post Screen:** Allows users to create ride-sharing posts with relevant details such as route and time.

- **Chat Screen:** Facilitates user communication through a built-in messaging interface.
- **Edit Post Screen:** Enables users to update or modify their existing ride-sharing posts.

2.5.2.1.2 Services The service layer is responsible for interacting with the backend API, Firebase notifications, and real-time communication. It encapsulates the business logic for seamless data transfer between the backend and frontend.

- **MessageService:** Handles messaging functionality by sending and retrieving messages from the backend.
- **FirebaseService:** Manages push notifications to inform users about ride requests or messages.
- **AuthService:** Handles user authentication, ensuring secure login using GTU's OAuth2 system.
- **SignalRService:** Enables real-time communication, such as live message updates and ride request notifications, using SignalR technology.

2.5.2.1.3 Widgets Widgets in Flutter are reusable user interface components that simplify the creation of consistent and modular designs.[7]

- **CustomAppBar:** A reusable app bar widget used across different screens for consistent navigation and design.
- **Input Fields and Buttons:** Provide standardized components for user interaction, ensuring a consistent UI/UX.

2.5.2.1.4 Utils Utility classes contain helper methods and shared logic used across multiple parts of the application.[8]

- **JourneyUtils:** Provides helper functions, such as calculating the nearest available date for a ride.

2.5.2.1.5 Models The data models define the structure of the information used within the application, ensuring consistency between the frontend and backend.[9]

- **RidePostModel:** Represents the details of a ride-sharing post, such as starting point, destination, and available seats.
- **UserModel:** Represents user data, such as name, email, and sustainability points.

2.5.2.2. Diagram Explanation

The diagram visually represents the relationships and interactions between the classes:

- **Screens interact with Services** to fetch or send data to the backend via APIs.
- **Widgets** are used within Screens to create a dynamic and reusable interface.
- **Services like FirebaseService and SignalRService** ensure real-time updates and notifications, directly supporting the application's responsiveness.
- **Models** act as data containers for the information passed between layers.
- **Utils** provide reusable logic, improving code efficiency and reducing redundancy.

This modular structure ensures scalability, maintainability, and a clean separation of concerns, making the application efficient and easy to extend.

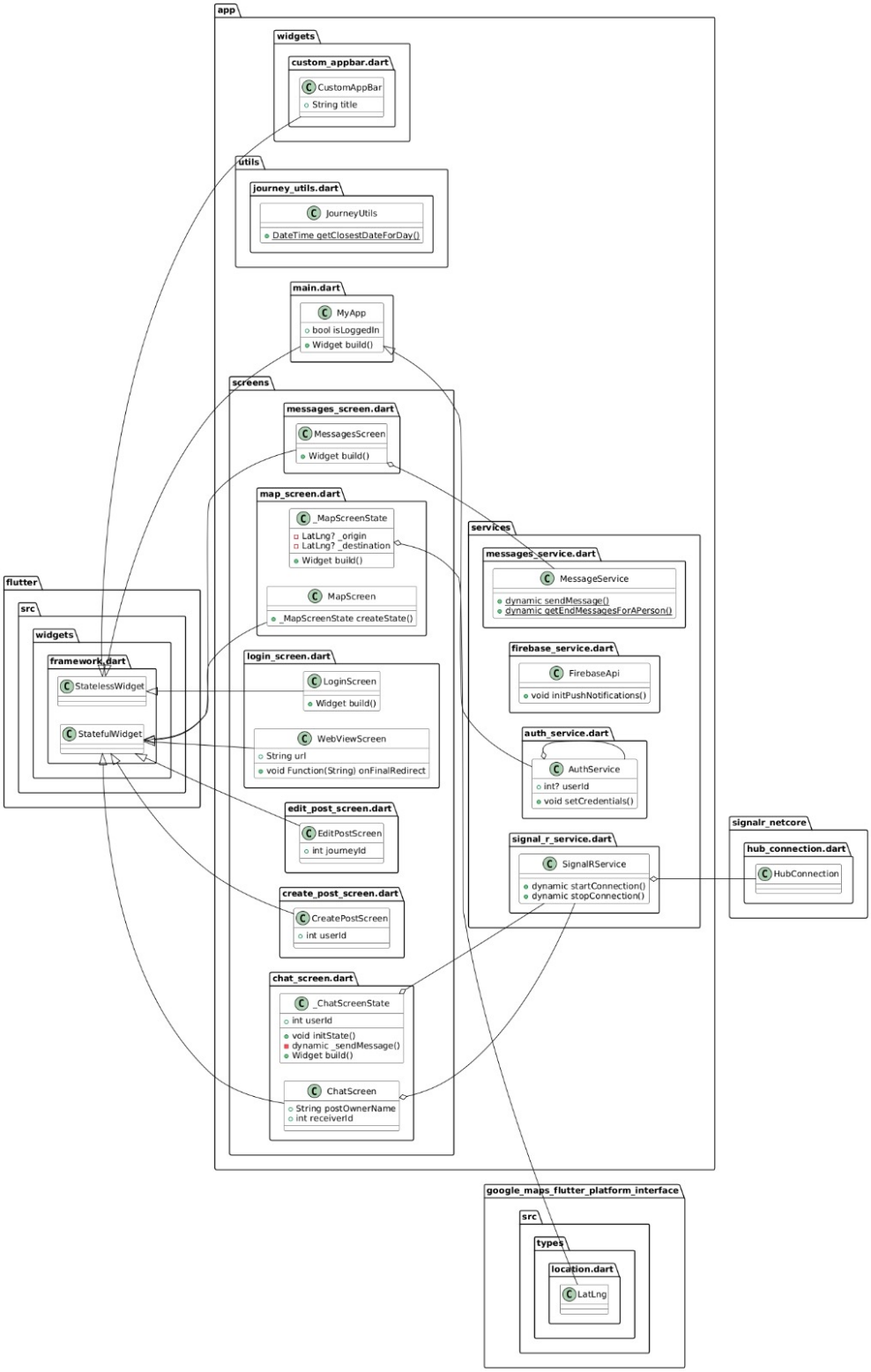

Figure 2.10: Flutter Class Diagram

3. RESULTS AND EVALUATION

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

3.1. Test Results

The application has undergone various tests to ensure the reliability and efficiency of its core functionalities. Below are the key findings:

- **Post Creation:** Users can successfully create ride posts by specifying starting and destination points. During tests, it was verified that all details (such as date, time, vehicle availability, etc.) were correctly stored in the database without any issues.
- **Messaging:** The messaging feature allows users to communicate seamlessly with each other. Messages were delivered on time and correctly routed to the intended recipient during tests.
- **Ride Request Management:** Users can send ride requests for existing posts. These requests were successfully recorded, and post owners could accept or reject them without any issues.
- **Notification System:** Real-time notifications, powered by Firebase, were tested successfully. Users received immediate alerts for new ride requests, updates to their posts, and incoming messages. This ensured users stayed informed throughout the interaction process.
- **GTU OAuth2 Integration:** User authentication using GTU's OAuth2 system worked flawlessly. Only verified university members could access the application, ensuring a secure environment for all users.

3.2. Some Outputs of Test Results

The following section showcases the key user interfaces of the car-sharing mobile application. These screens highlight the primary functionalities of the system, including user authentication, browsing ride posts, viewing post details and managing requests.

Each interface is designed to provide a seamless and intuitive user experience, ensuring that users can easily navigate through the application and perform their desired actions efficiently.

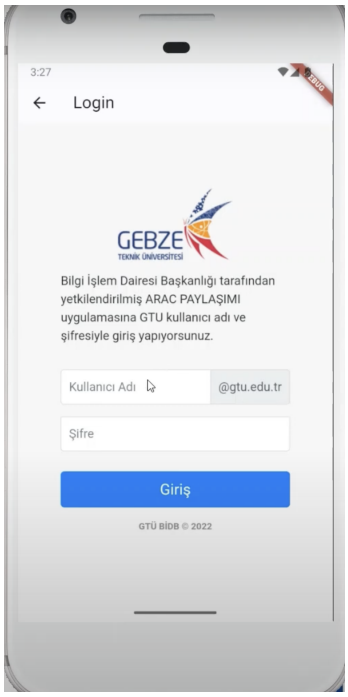

Figure 3.1: Login Screen

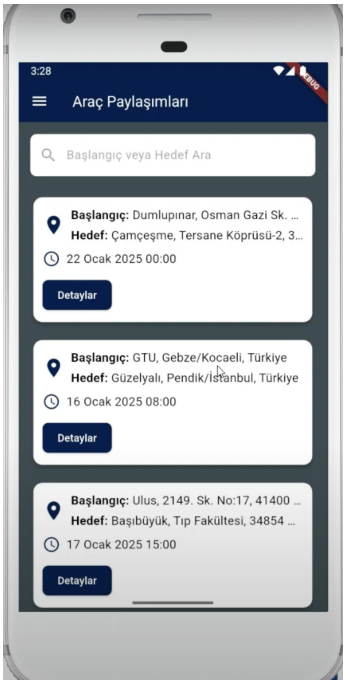

Figure 3.2: Post Viewing Screen

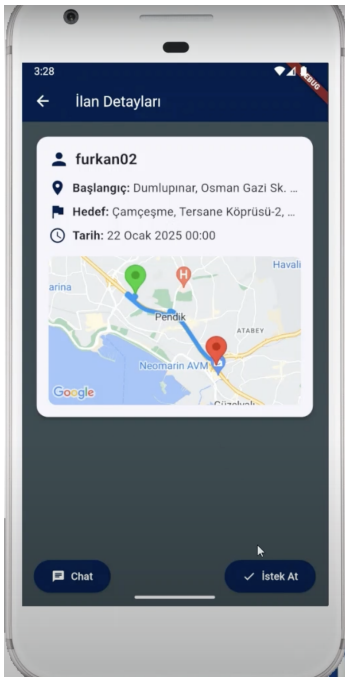

Figure 3.3: Post Details Screen

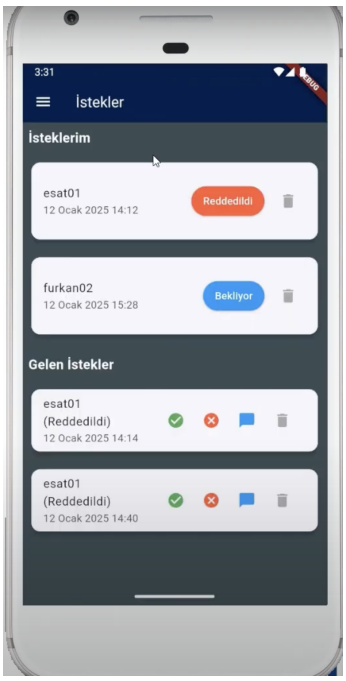

Figure 3.4: Requests Screen

3.3. Success Criteria

The application is optimized for high-speed network conditions and is designed to provide a fast, efficient, and seamless user experience. The following performance benchmarks were defined and tested. The following performance benchmarks are applicable under network conditions with a download speed of 55 Mbps or higher and an upload speed of 28 Mbps or higher.

- **Messaging Speed:** The system processes and delivers messages within **1 second**, ensuring real-time communication between users.
- **Application Launch Time:** Users can open the application and access the main dashboard in **under 2 seconds**, providing a fast and responsive experience.
- **Post Viewing Performance:** The post viewing page retrieves and displays **20 posts within 1 second**, enabling users to browse available ride-sharing opportunities efficiently.

3.3.1. External Service Dependency Note

Some functionalities of the application depend on third-party services, which influence response times and performance. These dependencies include:

- **User Authentication:** Managed through the GTU OAuth2 system, ensuring only verified university members can log in securely.
- **Notifications:** Delivered via Firebase Notification Service, providing real-time alerts for user actions like ride requests or messages.
- **Post Creation and Map Functionality:** Powered by Google Maps services to enable users to select starting and destination points visually, and view routes seamlessly.

Since the performance of these features is tied to external service providers, specific response times cannot be guaranteed for these operations. However, the application has been designed to handle such dependencies efficiently, ensuring minimal delays for end users.

By recognizing these dependencies, the defined performance benchmarks focus on features under the application's direct control while accounting for potential variations in third-party service response times.

3.3.2. Performance Test Results

This section presents the results of performance tests conducted for critical functionalities of the car-sharing application. Each test was performed under optimal network conditions to ensure accuracy.

3.3.2.1. Creating a Post

- **Description:** The user creates a new ride post by providing details such as starting and destination points, date, and time.
- **Test Results:**
  - Button Clicked: `2025-01-13 21:22:24.522886`
  - Post Created: `2025-01-13 21:22:24.570436`
  - **Total Time Taken:** 47 milliseconds
- **Conclusion:** The post creation feature is highly efficient, enabling users to create ride posts with minimal delay.

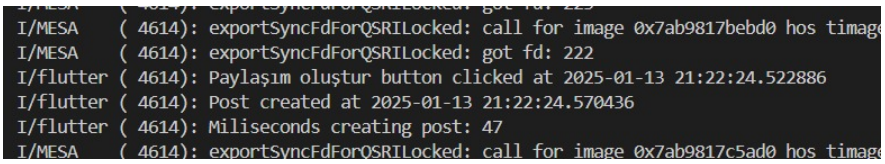

Figure 3.5: Test of Creating Post

3.3.2.2. Editing a Post

- **Description:** The user edits an existing ride post by modifying details such as time or destination.
- **Test Results:**
  - Button Clicked: `2025-01-13 21:28:00.683697`
  - Post Updated: `2025-01-13 21:28:00.720917`
  - **Total Time Taken:** 37 milliseconds
- **Conclusion:** The post editing feature is highly responsive, updating changes within a minimal time frame.

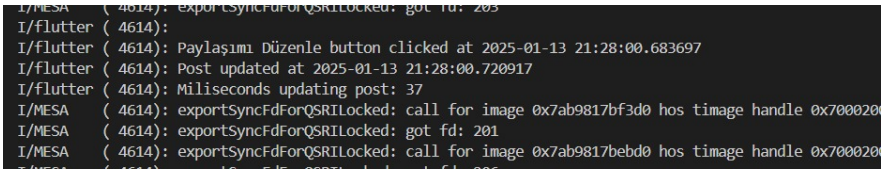

Figure 3.6: Test of Updating Post

3.3.2.3. Deleting a Post

- **Description:** The user deletes an existing ride post from the platform.
- **Test Results:**
  - Button Clicked: `2025-01-13 21:29:07.323725`
  - Post Deleted: `2025-01-13 21:29:07.418398`
  - **Total Time Taken:** 94 milliseconds
- **Conclusion:** The deletion functionality is fast and removes posts from the system promptly.

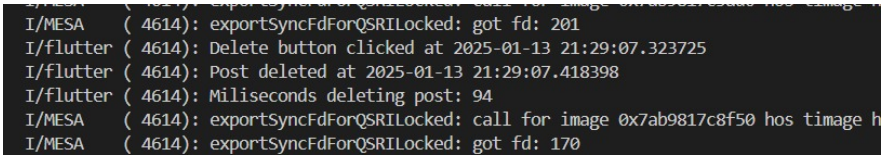

Figure 3.7: Test of Deleting Post

3.3.2.4. Loading Post List

- **Description:** The user navigates to the ”Araç Paylaşımları” page to view available posts.
- **Test Results:**
  - Content Requested: `2025-01-13 21:02:05.476401`
  - Content Fetched: `2025-01-13 21:02:05.549265`
  - **Total Time Taken:** 72 milliseconds
- **Conclusion:** The application retrieves and loads post data quickly, ensuring an efficient browsing experience.

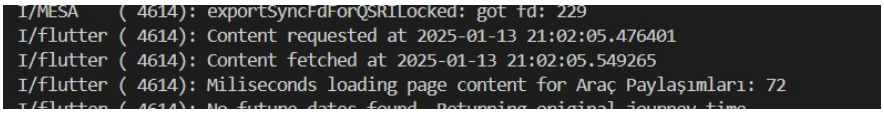

Figure 3.8: Test of Listing Posts

3.3.2.5. Sending a Message

- **Description:** The user sends a message to another user via the chat interface.
- **Test Results:**
  - Button Clicked: `2025-01-13 21:50:51.114187`
  - Message Sent: `2025-01-13 21:50:51.365543`
  - **Total Time Taken:** 251 milliseconds
- **Conclusion:** The messaging system ensures real-time communication with acceptable response times.

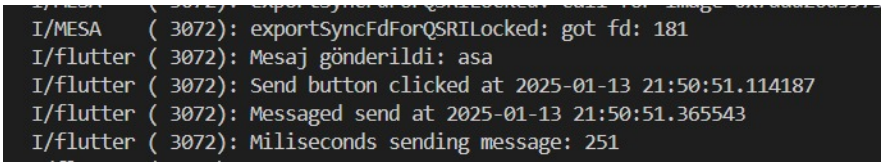

Figure 3.9: Test of Messaging Posts

The tests were conducted locally on the developer's machine, which ensured minimal latency and optimal performance. Consequently, the test results are very succesfull expected benchmarks due to the absence of network delays and server-side load. However, when the application is deployed to a live server, factors such as server capacity, network latency, and concurrent user activity may impact performance. The actual response times could vary depending on these conditions. This distinction should be considered when evaluating the test outcomes and real-world performance expectations.

BIBLIOGRAPHY

[1] A. Freeman, *Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages*, 9th. Apress, 2022.

[2] E. Çifci, "Gtu oauth servisleri," Gebze Teknik Üniversitesi, 1, 2024.

[3] Firebase Team. "Firebase documentation." (2025), [Online]. Available: https://firebase.google.com/docs (visited on 01/14/2025).

[4] Google Developers. "Building apps with flutter." (2025), [Online]. Available: https://www.youtube.com/watch?v=k0zGEbiDJcQ (visited on 01/14/2025).

[5] Microsoft Docs Team. "Asp.net core documentation." (2025), [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0&WT.mc_id=dotnet-35129-website (visited on 01/14/2025).

[6] J. Albahari, *C# 12 in a Nutshell: The Definitive Reference*, 1st. O'Reilly Media, 2023.

[7] S. Alessandria and B. Kayfitz, *Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart*, 1st. Packt Publishing, 2021.

[8] Flutter Team. "Flutter cookbook." (2025), [Online]. Available: https://docs.flutter.dev/cookbook (visited on 01/14/2025).

[9] A. Biessek, *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2*, 1st. Packt Publishing, 2019.

APPENDICES

YouTube Demo

https://www.youtube.com/watch?v=uNyVTdJrgJY