Implementation Document

for

Chalo Kart

(Golf Cart Management)

Prepared by

Group 13: **Group Name:** EE Got Latent

Trijal Srivastava	221144	trijals22@iitk.ac.in
Snehasis Satapathy	221070	ssnishasis22@iitk.ac.in
Gautam Arora	220405	garora22@iitk.ac.in
Naman Gupta	220686	namangupta22@iitk.ac.in
Divyam Agarwal	220376	divyamag22@iitk.ac.in
Udbhav Agarwal	221149	audbhav22@iitk.ac.in
Deham Rajvanshi	220335	dehamr22@iitk.ac.in
Saksham Parihar	220939	psaksham22@iitk.ac.in
Dharvi Singhal	220354	dharvis22@iitk.ac.in

Course:	CS253
Mentor TA:	Mr. Vikrant Chauhan
Date:	March 28, 2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
v1.0	Trijal Srivastava Snehasis Satapathy Gautam Arora Naman Gupta Divyam Agarwal Udbhav Agarwal Deham Rajvanshi Saksham Parihar Dharvi Singhal	The first version of the Software Implementation Document	28/03/2025

1 Implementation Details

The designed golf-cart management system app **"Chalo Kart"** has been implemented in the following broad categories -

2.1 Backend

1. **Programming Language:**
    - **Dart:** Used within the Flutter framework to manage API calls, handle state, and process user interactions efficiently.
    - **Firebase Firestore:** A NoSQL cloud database for real-time data synchronization and storage.
    - **Cloud Functions (Node.js):** Serverless backend logic to handle authentication, notifications, and ride updates.
2. **Key Features:**
    - **Real-time synchronization:** Instantly updates ride bookings, availability, and user status across devices.
    - **Scalability & Security:** Ensures secure authentication and handles large-scale user traffic.
    - **Automation:** Cloud Functions manage notifications, payments, and cancellations seamlessly.
3. **Libraries and Packages**
    - **firebase_core:** Initializes and connects the app with Firebase services for backend functionality.
    - **firebase_auth:** Manages user authentication via Firebase, supporting email, phone, and social logins.
    - **http:** Enables sending and receiving HTTP requests, essential for API communication.
    - **flutter_geofire:** Manages geofencing and real-time location updates, which help track moving carts.
    - **firebase_messaging:** Handles push notifications, enabling ride status updates and alerts

2.2 Frontend

1. **Programming Language:**
    - **Dart:** Used to build the Flutter UI and handle logic for user interactions, state management, and API communication.
2. **Framework: Flutter.** It is a cross-platform UI toolkit that provides a fast and responsive UI with a rich set of customizable widgets, ensuring smooth animations, real-time updates, and an intuitive user experience.

3. **Key Features:**

    1. **Cross-Platform Compatibility:** Flutter enables the app to run seamlessly on Android, iOS, Web, and Desktop with a single codebase, reducing development effort and maintenance costs.
    2. **Fast and Responsive UI:** Flutter's widget-based system ensures a highly responsive UI, allowing smooth animations, real-time updates, and efficient performance even on low-end devices.
    3. **Integration with Google Maps:** The front end integrates Google Maps API for real-time cart tracking and location services.
4. **Libraries and Packages:**
    - **material.dart:** Embeds the Material Design components provided by Google. It ensures that the application follows the standard design guidelines.
    - **razorpay_flutter.dart:** Integrates the Razorpay payment gateway into our

Flutter application to include payment functionality in our Flutter app.

- **http.dart:** Provides utilities for making HTTPS requests and handling HTTPS responses in our Flutter application.
- **fluttertoast.dart:** We used this to provide a simple and customizable way to display toast notifications in a Flutter application.
- **cupertino_icons**: Provides iOS-style icons for a native-looking interface on Apple devices.
- **provider:** Implements efficient state management, ensuring smooth UI updates.
- **url_launcher:** Enables opening URLs, making calls, or sending emails from within the app.
- **google_maps_flutter:** Embeds interactive Google Maps, which are crucial for ride tracking and navigation.
- **intl_phone_field:** Provides a country-based phone number input field with validation.
- **email_validator:** Ensures correct email format input, improving user registration accuracy.

2.3 Database

For the database, we used **Firebase Firestore** as our database program. We chose it for its robustness, reliability, and straightforward implementation. Some other advantages which make it a popular choice are:

- **Real-time Updates:** Firestore ensures instant updates to ride status, user activity, and cart availability across devices.
- **Scalability & Security:** Supports growing user demand while ensuring secure data storage through Firebase Authentication.
- **Offline Access:** Cached data lets users view ride history and bookings with limited or no connectivity.

Libraries and Packages:

- **firebase_database:** Provides real-time synchronization of ride bookings, user data, and cart locations.
- **firebase_storage:** Stores and retrieves images or documents, such as profile pictures or receipts.

2.4 Building Systems:

- **Gradle (Android Builds):** Automates compilation, testing, and packaging of the app for Android, ensuring a smooth build process.
- **Flutter Pub (Dependency Management):** Manages third-party packages and libraries, ensuring seamless integration and easy updates of required dependencies.

Codebase

3.1 GitHub Repository

Link for our app: https://github.com/rock42069/ChaloKart

The above repository has two branches, one for the entire User code and another for the Driver's code.

3.2 Codebase Navigation

1. For User App

Root Directory Structure

```
chalokart_user/
├── android/                       # Android-specific files
├── ios/                          # iOS-specific files
├── lib/                          # Main source code directory
├── assets/                       # Static assets
├── test/                         # Test files
├── pubspec.yaml                 # Project configuration and dependencies
└── README.md                    # Project documentation
```

`/lib` Directory

The main source code directory containing all Dart files.

```
lib/
├── main.dart                 # Application entry point
├── screens/                  # UI screens
│   ├── splash_screen.dart    # Initial loading screen
│   ├── sign_in_screen.dart   # User authentication screen
│   ├── main_screen.dart      # Main application screen
│   └── verify_email_screen.dart # Email verification screen
├── models/                   # Data models
│   └── user_model.dart       # User data structure
├── services/                 # Business logic and services
│   ├── auth_service.dart     # Authentication service
│   ├── location_service.dart # Location handling service
│   └── payment_service.dart  # Payment processing service
├── utils/                    # Utility functions and helpers
│   ├── constants.dart        # Application constants
│   └── helpers.dart          # Helper functions
├── widgets/                  # Reusable UI components
│   ├── custom_button.dart    # Custom button widget
│   └── custom_text_field.dart # Custom text field widget
├── theme/                    # Theme configuration
│   └── app_theme.dart        # Application theme settings
└── providers/               # State management
    └── app_provider.dart     # Application state provider
```

`/android` Directory

Contains Android-specific configuration and files.

```
android/
├── app/                         # Android application files
│   ├── src/                    # Source files
│   │   └── main/              # Main source directory
│   │       ├── AndroidManifest.xml  # Android manifest
│   │       └── res/           # Android resources
│   └── build.gradle           # Android build configuration
├── gradle/                     # Gradle wrapper files
└── build.gradle               # Root build configuration
```

`/ios` Directory

Contains iOS-specific configuration and files.

```
ios/
├── Runner/                 # iOS application files
│   ├── AppDelegate.swift   # iOS app delegate
│   ├── Info.plist         # iOS configuration
│   └── Assets.xcassets/   # iOS assets
└── Podfile                # iOS dependencies
```

`/assets` Directory

Contains static assets used in the application.

```
assets/
├── images/                 # Image files
│   ├── logo.png           # Application logo
│   └── icons/             # Icon assets
├── fonts/                 # Custom fonts
└── translations/          # Localization files
```

`/test` Directory

Contains test files for the application.

```
test/
├── unit/                   # Unit tests
│   └── auth_test.dart      # Authentication tests
└── widget/                 # Widget tests
    └── button_test.dart    # Button widget tests
```

2. For Driver App

Root Directory Structure

```
chalokart_user/
├── android/                      # Android-specific files
├── ios/                         # iOS-specific files
├── lib/                         # Main source code directory
├── assets/                      # Static assets
├── test/                        # Test files
├── pubspec.yaml                 # Project configuration and dependencies
└── README.md                   # Project documentation
```

/android Directory

Contains Android-specific configuration and files.

```
android/
├── app/                         # Android application files
│   ├── src/                    # Source files
│   │   └── main/              # Main source directory
│   │       ├── AndroidManifest.xml # Android manifest
│   │       └── res/           # Android resources
│   └── build.gradle            # Android build configuration
├── gradle/                     # Gradle wrapper files
└── build.gradle                # Root build configuration
```

/ios Directory

Contains iOS-specific configuration and files.

```
ios/
├── Runner/                  # iOS application files
│   ├── AppDelegate.swift   # iOS app delegate
│   ├── Info.plist         # iOS configuration
│   └── Assets.xcassets/   # iOS assets
└── Podfile                # iOS dependencies
```

/assets Directory

Contains static assets used in the application.

```
assets/
├── images/                 # Image files
│   ├── logo.png           # Application logo
│   └── icons/             # Icon assets
├── fonts/                 # Custom fonts
└── translations/          # Localization files
```

/lib Directory

The main source code directory containing all Dart files.

```
├── lib/                      # Main source code directory
│   ├── main.dart            # Application entry point
│   ├── screens/             # UI screens and pages
│   │   ├── loginScreen/     # Authentication screens
│   │   │   ├── sign_in_screen.dart    # User login interface
│   │   │   ├── sign_up_screen.dart    # New user registration
│   │   │   └── forgot_password_screen.dart # Password recovery
│   │   ├── homeScreen/      # Main app screens
│   │   │   ├── main_screen.dart       # Main app container
│   │   │   ├── new_trip_screen.dart   # Trip booking interface
│   │   │   └── history_screen.dart    # Ride history
│   │   ├── profileScreen/  # User profile screens
│   │   │   ├── profile_screen.dart    # User profile view
│   │   │   ├── settings_screen.dart   # App settings
│   │   │   └── edit_profile_screen.dart # Profile editing
│   │   ├── car_info_screen.dart       # Vehicle information
│   │   ├── drawer_screen.dart         # Navigation drawer
│   │   └── splash_screen.dart         # App launch screen
│   ├── models/              # Data models and classes
│   │   ├── userModel.dart  # User data model
│   │   ├── rideModel.dart  # Ride data model
│   │   └── locationModel.dart # Location data model
│   │
│   ├── services/           # Business logic and API services
│   │   ├── authService.dart # Authentication services
│   │   ├── locationService.dart # Location tracking services
│   │   └── databaseService.dart # Firebase database services
│   ├── widgets/            # Reusable UI components
│   │   ├── customButtons/  # Custom button widgets
│   │   ├── customTextFields/ # Custom text field widgets
│   │   └── mapWidgets/     # Map-related widgets
│   ├── utils/              # Utility functions and constants
│   │   ├── constants.dart  # App constants
│   │   └── helpers.dart    # Helper functions
│   ├── tab_pages/          # Tab-based navigation pages
│   ├── themeProvider/      # App theme management
│   ├── Assistance/         # Support and help features
│   ├── global/             # Global variables and configurations
│   ├── infoHandler/        # Information handling logic
    └── pushNotification/   # Push notification handling
```

3.2.1 Instructions to Setup Development Environment:

- **Frontend and Backend(FireBase)**

- Instructions are mentioned in the READM E.md file on our GitHub repository.

Completeness

4.1 Implemented Features

4.1.1 Login + Registration

- **Registration Page**
  - Allows users to register with Chalo Kart by providing their full name, phone number, and email address. The user also creates a password and confirms it.
  - For security purposes, we have also incorporated phone number verification via OTP.
- **Login Page**
  - Allows registered users to log in using their email and password.
- **Forgot Password Page**
  - Enables users to initiate the password reset by providing their registered email address. A link is mailed to the users so they can reset their password.

4.1.2 Ride Now

- **Google Map Integrated**
  - Displays a Google Map where users can select the pickup and drop locations.
  - Users can either move the pointer on the map, and it will auto-detect the address, or they can explicitly type out the address. By default, the pickup location is mapped to their current locations, which is enabled by GPS service.
  - Once the two locations are set, a route is shown between the two pointers. The user can then request the ride.
- **Ride Request**
  - When we request a ride, we can choose from 'Public Cart' and 'Private Cart.'
  - Then, a driver is assigned to the user, showing the expected ETA, price, and number of seats chosen.

4.1.3 Active Ride

- **User Profile Display**
  - Shows the user's profile picture and full name at the top of the screen, providing personalization and user recognition.
- **Ride Status**

- Allows users to check their ride status.
- Users can view ongoing ride details, duration, and distance traveled by tapping this button.

– **Payment Mode**

- Payment is only done via the user's wallet. If the wallet has an insufficient balance, it will go negative, preventing the user from booking any more rides before the wallet is recharged.

– **End Ride Button**

- Initiates the process of ending the ride from the driver's side.

4.1.4 Driver's Page

– **Accepting Rides**

- Drivers will be prompted with "Accept Ride" whenever a new ride request is processed. They can either accept it or ignore it.
- Once the cart reaches the drop location, the driver will get an option to "End Ride." The amount the user pays to the driver is displayed next. After this, the driver can return to the home page and wait for further rides.

– **Statistics**

- The driver will be able to see their earnings through the sidebar, and the users will be able to see their wallets.

4.1.5 Ride History

– **Completed Rides**

- Lists all past rides that have been completed.
- Includes information such as start location, end location, and duration for each ride and the amount paid by the user

4.1.6 Feedback

- **Rating Drivers and Users**
  - Users can rate the driver after a ride is completed. Similarly, the driver also has the option to rate the particular user.

4.1.7 Wallet and Payments

- **Wallet for Users**
  - All the users will have a wallet to add an amount for current and future ride payments.
  - When a ride has been completed, and the wallet doesn't have enough balance, the amount will go negative, but this privilege is present just for once.

4.2 Future Development Plan:

4.2.1 Algorithm Development

- We want to optimize the algorithm to make efficient use of pooled rides. This will cut the ride's time to reach a user and enable us to cater to many more users instantly.

4.2.2 Shuttle Routes

- Locations on the campus between which there is a significant need for users to travel throughout the day can be solved by golf carts moving on such fixed routes.

4.2.3 Discounts for campus residents

- We can provide discounts to campus residents when they upload their ID cards. This will help us analyze groups of people using the service more and the durations of the day when there is most traffic.
- Frequent users on the app can also be given further promotions and discounts.

Group Log

NOTE: Every team member worked regularly on assigned tasks, collaborating whenever needed. This log only covers online/in-person meetings and significant discussions.

S.No	Date	Timings	Venue	Description
1	14/02/2025	21:30 to 23:00	Google Meet	Discussed about Implementation Work decided the timeline and divided work among teammates
2	22/02/2025	11:00 to 14:00	RM Building	Gathered Resources for learning about Flutter and Django, and set our machines for the development process
3	06/03/2025	21:30 to 22:00	Discord	Meet to compile our work and analyze our progress. **Switched from Django to Firebase for Backend.**
4	14/03/2025	14:30 to 17:00	RM Building	Completed Backend and Frontend. Started working on the integration process and Implementation Document
5	23/03/2025	17:30 to 18:30	Google Meet	We discussed a few doubts regarding testing and integration.
6	27/03/2025	21:00 to 23:30	Library	Completed the Implementation Work and Finalised the Implementation Document