**Zip App Orange Team**

**Developer Manual**

Authors:

Trevor Aupperle, Camden Davis, Benjamin Fisk, Mohab Yousef

COMP 4710 Senior Design Team #5

Department of Computer Science and Software Engineering

Samuel Ginn College of Engineering, Auburn University

April 26, 2024

# Table of Contents

[**Table of Contents 1**](#_3tcvoaom2j9d)

[**1 Zip Gameday 2**](#_cbyj62r041of)

[**1.1 Overview 2**](#_4t277ko8zn45)

[1.2 Installation and Setup 2](#_ix6o7oxn69p)

[1.2.1 Installing Flutter 2](#_r3ah1v9gf27h)

[1.2.2 Requesting Access 3](#_45hn3trguamn)

[1.2.3 Installing the App 3](#_qqrfs871wdux)

[1.2.4 Installing Firebase CLI 3](#_ix7eg4ia25fr)

[1.2.5 Running the Application 3](#_fk8ko0i3ojws)

[**2 Firebase and Google Services 3**](#_9pz94qm2ln7w)

[2.1 Overview 3](#_3dxandsb77n1)

[2.2 Guide 4](#_1725rabggoxi)

[**3 Stripe 4**](#_gu8ohccakzvi)

[3.1 Overview 4](#_dcw88hi5n21)

[3.1 Overview 4](#_ukg9ittxa5co)

[**4 System Design 4**](#_ykr94ekzours)

[4.1 Data Models 4](#_ckfjmcg5gsom)

[4.1.1 User 5](#_1rm420urcvki)

[4.1.2 Rides 5](#_7ngn5aht0cn6)

[4.1.3 Request 6](#_56ocqzfcqgck)

[4.1.4 Driver 6](#_ex3vt9j6v6mc)

[4.1.5 Primary Payment Method 6](#_w4160jtfb5k3)

[4.2 Design Documentation 7](#_x77o9oe7zskp)

[4.2.1 Architecture 7](#_a3ys7e7knnc1)

[4.2.2 Firebase and Other Google Services 7](#_v91ngnksn2e1)

[4.2.3 Stripe 7](#_n4umo0iuglr8)

[4.2.4 Figure1 - User Flowchart 8](#_yha31ml8cxfv)

[4.2.6 Figure 2 - Firebase and Stripe Interaction Chart 8](#_ezwdgo6sie5p)

[4.2.7 Assumptions and Dependencies 9](#_nk83ontwb6gh)

[4.2.7.1 Minimum OS Versions 9](#_1ot4884uq162)

[4.2.7.2 Minimum OS Versions 9](#_ll36a8j2t734)

[4.3 User Stories 10](#_8rbr7vdsks2z)

[4.3.1 Table 1 - Flutter Migration User Story 10](#_n5gp3qabcl83)

[4.3.2 Table 2 - Operating Systems Check User Story 10](#_hp948v689g01)

[4.3.3 Table 3 - Flutter Facebook Login Package Migration User Story 11](#_d1a280xqvv9l)

[4.3.4 Table 4 - Geoflutterfire Package Migration to Geoflutterfire2 User Story 11](#_4udhcvozqhar)

[4.3.5 Table 5 - Find an Alternative to Unicorndial Package User Story 12](#_ex5o55ifdrew)

[4.3.6 Table 6 - Strip\_Payment Package Migration to Flutter\_Stripe User Story 12](#_3h8npg8imrqt)

[4.3.7 Table 7 - Location Package Update User Story 12](#_eax6zlffm5m)

[4.3.8 Table 8 - Figma Mockups User Story 13](#_5d2jtocu7ne5)

[4.3.9 Table 9 - Implement UI Changes User Story 13](#_fn7n47l5t8lo)

[4.3.10 Table 10 - Driver Directions Research User Story 13](#_g02bz6rdbepv)

[4.3.11 Table 11 - Driver Directions Implementation User Story 14](#_s58k4olu0oru)

[4.3.12 Table 12 - Rider ‘On the Way’ Notification User Story 14](#_n49aq43apqis)

[4.3.13 Table 13 - Rider Status User Story 15](#_qonpwefmwivu)

[4.3.14 Table 14 - Rider Recap Screen User Story 15](#_40eoyy1c2dkw)

[4.3.15 Table 15 - Drivers Near Me User Story 15](#_1b43cpeus7hn)

[4.3.16 Table 16 - Feedback/Rating System User Story 15](#_sndrzcv0ufu6)

[4.3.17 Table 17 - Enhance Feedback and Tips Screen User Story 16](#_e0bmha9k03q9)

[4.3.18 Table 18 - Display Past Trips User Story 16](#_nyumkaxd7etk)

# 

# 1 Zip Gameday

## 

## 1.1 Overview

Zip Gameday is a mobile application developed using the Flutter framework. Flutter utilizes the Dart programming language and its associated package library, which [can be accessed here](https://pub.dev/). The documentation for Flutter [can be found here](https://docs.flutter.dev/). The Flutter framework is supported by Google and was chosen because it supports the development of applications for IOS and Android operating systems using the same codebase. The application uses many other Google services to accomplish tasks across the system such as authentication, data storage, and user location access.

## 

## 1.2 Installation and Setup

### 

### 1.2.1 Installing Flutter

The first step to installation is installing the Flutter framework and adding it to your path. Visit the Flutter installation page [here](https://docs.flutter.dev/get-started/install) and choose your computer’s operating system. If you are developing on a Windows machine, you will choose Android as your type of app to develop. If you are developing on a Mac, you may choose Android or IOS as your target app type, but you are advised to choose IOS. Once you have chosen your target operating system, follow the steps to install the framework on your machine. It is recommended that you install Flutter using VSCode and VSCode’s Flutter Extension if you intend to use it as your IDE. Be careful to ensure that you allow VSCode to add Flutter to the path because this is required for development.

### 1.2.2 Requesting Access

This project’s sponsor - Russell Anderson - owns the services that this application employs for various tasks. You will need access to these services to run the application. Contact him at [russell.anderson@zipgameday.com](mailto:russell.anderson@zipgameday.com) to request access to the following services:

* Github Repository
* Firebase Console
* Stripe Console

### 

### 1.2.3 Installing the App

The next step is installing the application. To do this, clone the ZipApp repository from Git Hub, which you should have gotten from Russell Anderson. Once the repository has been cloned, navigate to the folder in your terminal and run “flutter pub get” to install the application’s dependencies.

### 

### 1.2.4 Installing Firebase CLI

The next step is installing the Firebase CLI. Visit the [Firebase documentation here](https://firebase.google.com/docs/cli) and follow the instructions for your operating system. It is recommended that you use npm to install the CLI.

### 1.2.5 Running the Application

**iOS:** To Run the application on Mac, follow the instructions listed here to [Add and Remove Simulators](https://developer.apple.com/documentation/safari-developer-tools/adding-additional-simulators#Add-and-remove-Simulators). Next, follow the instructions here to [start your simulator](https://developer.apple.com/documentation/xcode/running-your-app-in-simulator-or-on-a-device). Alternatively, if you have an iOS device, you may connect it to your MAC and run the app on your device. Once your target device is up and running, open VSCode and use the shortcut ‘CMD + SHIFT + P’ to show and run commands, then run the Flutter: Select Device command, and select your device. Then run your application from the main.dart file.

**Android:** To Run the application on an Android device using a Windows machine, follow [Google’s instructions](https://developer.android.com/studio/run/emulator) to set up an emulator in Android Studio. Once your emulator is running or on a physical Android device, follow [these instructions](https://docs.flutter.dev/get-started/test-drive) under ‘Run your sample Flutter app’ to run the app on your device.

# 2 Firebase and Google Services

## 2.1 Overview

Firebase provides a lot of functionality, including Firebase authentication, Cloud Firestore, and Cloud Functions. Zip uses these functions to authenticate and store data. Firebase is a real-time database and allows our app to stay up to date, which is necessary for a driving service app.

Furthermore, the app cannot function without internet access and a connection to Zip's Firebase account. Next, to restrict access to some sensitive parts of the database to users, Firebase rules will be implemented. However, these rules still need to be planned and implemented. Firebase Authentication allows users to create an account through an email/password combination. Using Firebase cloud functions, the team can write and deploy functions to handle any necessary backend processes. These functions are either triggered or called when the app does a specific action: for example, applying a promo code or making a payment. Image and file storage will be integrated through Firebase Fire storage. For location-based services, Google Maps API and various Flutter packages will be used to handle location searches, places autocomplete, address validation, and routes for driver navigation.

## 2.2 Guide

The cloud\_firestore package is utilized for all things related to Firebase. The documentation [can be found here](https://firebase.google.com/docs/reference). To start writing cloud functions please consult Google's documentation, which [can be found here](https://firebase.google.com/docs/functions/firestore-events).

# 3 Stripe

## 3.1 Overview

All payments for Zip GameDay are processed via Stripe. Stripe is used for both user payments and driver payouts. The Stripe APIs and backend processes are utilized for all customer-related payments. Stripe Connect with an Express plan will be used to pay drivers. Payments for users and drivers still need to be implemented.

## 3.1 Overview

All back-end processes are done in Firebase via Cloud Functions and all functions are written in JavaScript. For frontend capabilities, the flutter\_stripe package is utilized. For security purposes, no sensitive information is handled in the front end. As a result, flutter\_stripe is limited in its use and a separate service file called payment.dart in the business folder is utilized. The flutter\_stripe package is used to create the card form UI, the main form of paying out the drivers, and the main form of payment for the application. For more information about understanding, accessing, and implementing Cloud Function Triggers refer to [the Cloud Function Documentation](https://firebase.google.com/docs/functions/firestore-events). For all documentation on Stripe Integration refer to [the Stripe API Documentation](https://stripe.com/docs/api). For documentation and information related to Stripe Connect and the Express account type refer to [the Connect with Express Documentation](https://docs.stripe.com/connect/express-accounts). Stripe is used as a separate entity for all payment functions associated with the application.

# 4 System Design

## 4.1 Data Models

The core models that are displayed below are used in Firebase and that are used will be detailed below.

### 4.1.1 User

The user model can be found under lib/models/user.dart. The details in the code correspond to

the user attributes in Firebase. The User attributes will be stored in the Cloud Firestore

database. The User model will have:

acceptedPrivPolicy – if a user has accepted the Privacy Policy

acceptedtc – if a user has accepted the terms and conditions

credits – available promotional credits

currentRideId – the user’s current ride id

defaultTip – what the user inputs as their default tip for rides

driverPassword – the password for the user to access their driver account

email – email of the user

firstName – the first name of the user

homeAddress – address of the user

isDriver – details if the user is a driver or not

isRiding – if the user is currently ride

lastActivity – last sign-in date

lastName – last name of user

pastDrives – the previous trips that the user has driven for the program

pastRides – the previous trips that the user has paid for and been driven on

phone – phone number of user

profilePictureURL – picture URL for the user

ratingCount – the number of ratings that have been given to the user

uid – unique identifier for the user

userRating – the average rating for the user

### 4.1.2 Rides

The user model can be found under lib/models/ride.dart. The details in the code correspond to

the user attributes in firebase. The Ride attribute will be stored in the Cloud Firestore

detailing rides. The Ride model will have:

destinationAddress – destination address

drid – driver’s unique id

driverName – name of driver

driverPictureURL – picture of the driver

pickupAddress – pickup address

status – status of the ride

uid – unique identifier of the user

username – user’s username

userPictureURL – picture of the user

### 4.1.3 Request

The user model can be found under lib/models/request.dart. The details in the code correspond

to the user attributes in firebase. The Request attribute will be stored in Cloud Firestore

detailing ride requests. The Request model will have:

destinationAddress – ride destination address

id – request id

name – name

photoURL – url of the photo

pickupAddress - ride pickup address

price – price of the ride

timeout – timeout

### 4.1.4 Driver

The Driver attribute will be stored in Cloud Firestore detailing ride requests. The

Drivermodel will have:

currentRideID - ID of ride if driver is in currently in a ride

fcm\_Token - firebase cloud messaging token

firstName - first name of the driver

geoFirePoint - location of the driver

isAvailable - if driver is in an idle state

isOnBreak – if a driver is currently on break

isWorking - if driver is currently working

lastActivity - latest time that driver was signed on

lastName - last name of the driver

profilePictureURL - profile picture of the driver

totalHoursWorked – the total hours a driver has worked in the current pay period

uid - unique identifier of the driver

The first time a rider goes to sign into the driver portal, they will be asked to create a password for their driver account. After successfully inputting the password, they will gain access to the driver portal home page. When the button is clicked to switch to the driver portal, a check will take place to see if the current user has signed up for the driver portal before. If they have, then they will be prompted to enter their password. If not, then they will be prompted to create one.

### 4.1.5 Primary Payment Method

The user will have the ability to specify their primary payment method, which is stored in the user’s device cache, not on Firebase.

The primary\_payment\_method will have:

applePay – bool to indicate whether primary payment method is Apple Pay

googlePay – bool to indicate whether primary payment method is Google Pay

card – bool to indicate whether primary payment method is a card payment method

paymentMethodId – if the primary payment method is a card, then this will contain the payment method id from Stripe.

## 4.2 Design Documentation

### 4.2.1 Architecture

Zip Gameday is built utilizing the programming language Dart, and the framework Flutter. Flutter is an open-source UI software development kit supported and developed by Google, which allows cross platform mobile development within a single codebase. The app’s frontend capabilities and services are implemented through two structures: screens and business. The business structure allows separation between logic code, such as payment processing, and authentication, and the UI, e.g., the screens. Additionally, our team utilizes other Google services and products such as Maps, Firebase, and other APIs and tools for developers. Using Firebase, the Zip development team does not need to set up and create a personalized server and database. To handle payment processing to customers and payouts to drivers, we utilize the services of Stripe.

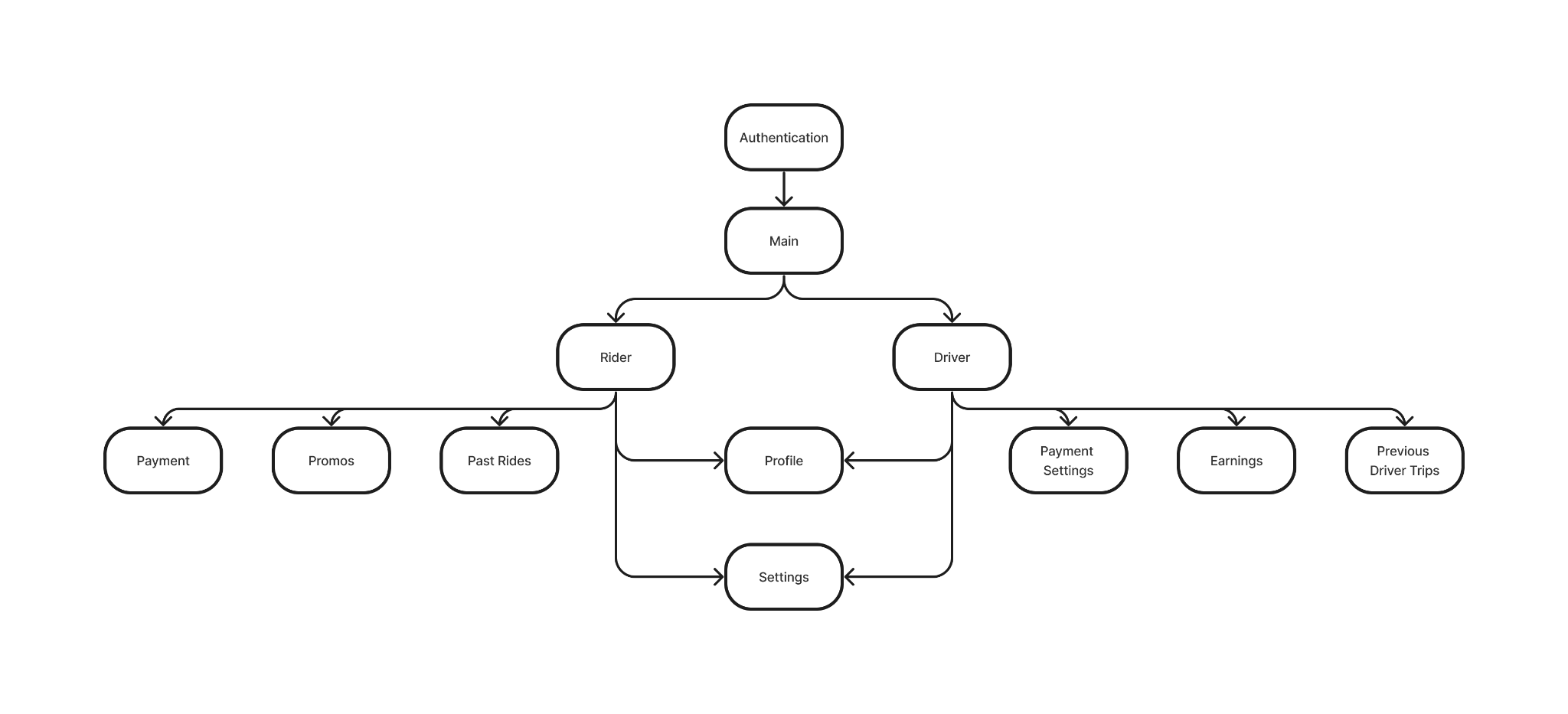
### 4.2.2 Firebase and Other Google Services

Zip Gameday utilizes Firebase services including but not limited to Firebase authentication, Cloud Firestore, and Cloud Functions. These services handle authentication, store data, and process server functionalities. Cloud Firestore is a real-time database that allows the application to be updated in real-time. Internet access and a connection to Zip’s Firebase accounts are required to allow the application to function. To ensure a safe database for users, Firebase rules need to be implemented. This can be configured through the Firebase console. Authentication for creating an account is completed via Firebase Authentication. Firebase cloud functions allow the team to write and deploy functions that are stored in the cloud. These functions may handle any backend processes needed. They can be triggered via a call within the client portion of the app or called when the application performs a specific action (i.e. making a payment or creating an account). Firebase Messaging is used to handle push notifications. Firebase Firestorage is utilized for image and file storage. For location-based services, Geolocator and Google Maps API are used to handle location-based service, places autocomplete, address validation, and routes for driver navigation.

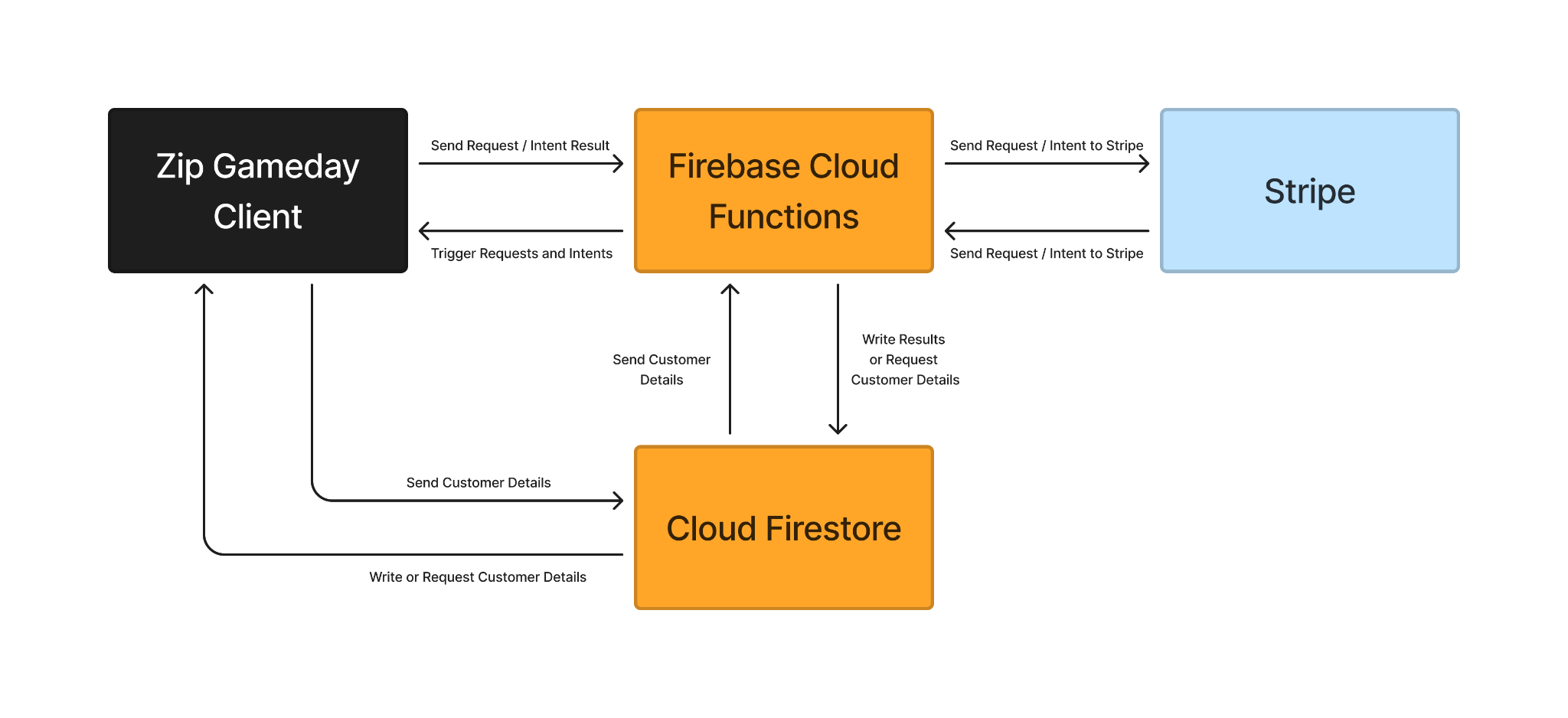
### 4.2.3 Stripe

All payment processing within the Zip Gameday application is done via Stripe. This includes functionalities such as creating customers to allow them to add payment methods, delete payment methods, make payments, and request refunds. The functionality of paying out drivers still needs to be completed and must be completed through Stripe Connect. To implement and create Stripe functions, a combination of processes between Cloud Firestore, Cloud Functions, and the client application is needed. On the client side, the flutter package stripe\_payment is used to integrate Stripe into the application through Firebase Cloud Functions and Cloud Firestore. The flutter package handles the client aspect of Stripe such as creating a card form and extracting the payment method ID. To ensure a secure backend process, cloud functions are used to connect with Stripe and perform all service processes. The process is as follows, stripe is triggered in the application by a client action that triggers a specific cloud function or customer request. Stripe is then sent the request from Firebase; Stripe handles the request and sends the result back to Firebase. If the result is a success, then Firestore is updated. The diagrams below show a diagram of how Flutter, Firebase, and Stripe interact with each other.

### 4.2.4 Figure1 - User Flowchart



### 4.2.6 Figure 2 - Firebase and Stripe Interaction Chart



### 

### 4.2.7 Assumptions and Dependencies

#### 4.2.7.1 Minimum OS Versions

1. Android 13 Tiramisu
2. iOS 16.0

#### 4.2.7.2 Minimum OS Versions

All current Dart Packages are in the pubspec.yaml file. The dependencies should automatically update when running ‘flutter pub get’ if the listen version contains a ^ symbol. When adding new dependencies, place them in the pubspec.yaml file and run ‘flutter pub get’, or use the command ‘flutter pub add [PACKAGE NAME]’. Then the library should be able to be imported into any file. Please be cautious when adding new dependencies as they may require updated versions of different sdks and packages which can potentially break the application.

1. Flutter
2. Dart
3. Firebase
4. Stripe
5. Google Maps
6. Dart Packages
   1. cloud\_firestore: ^4.14.0
   2. cloud\_functions: ^4.6.0
   3. firebase\_analytics: ^10.8.0
   4. firebase\_auth: ^4.16.0
   5. firebase\_core: ^2.24.2
   6. firebase\_messaging: ^14.7.10
   7. firebase\_storage: ^11.6.0
   8. flutter:
   9. sdk: flutter
   10. flutter\_dotenv: ^5.1.0
   11. flutter\_polyline\_points: ^2.0.0
   12. flutter\_speed\_dial: ^7.0.0
   13. flutter\_stripe: ^10.0.0
   14. geolocator: ^11.0.0
   15. geoflutterfire2: ^2.3.15
   16. google\_maps\_flutter: ^2.5.3
   17. google\_place\_plus: ^1.0.0
   18. google\_sign\_in: ^6.2.1
   19. image\_picker: ^1.0.7
   20. intl: ^0.19.0
   21. json\_serializable: ^6.7.1
   22. location\_permissions: ^4.0.1
   23. lucide\_icons: ^0.257.0
   24. mailto: ^2.0.0
   25. material\_design\_icons\_flutter: ^7.0.7296
   26. percent\_indicator: ^4.2.3
   27. shared\_preferences: ^2.2.2
   28. strings: ^3.0.0
   29. url\_launcher: ^6.2.4
   30. webview\_flutter: ^4.4.4
   31. dotted\_border: ^2.1.0

## 4.3 User Stories

User stories are marked as either “complete”, “in-progress”, or “not started.” User stories that are marked as complete have been completed and should be functional. Stories that are marked as “in-progress” have not been finished, but the team has reached a stopping point that allows the next development team to continue with a certain degree of ease. Some of the user stories listed were completed by past development teams but are included as the information is vital to understanding the application. Please note stories listed as ‘completed’ may still be subject to potential bugs especially if a future story needs to alter previous functionality.

### 4.3.1 Table 1 - Flutter Migration User Story

| **1.0 Flutter Migration** |  |
| --- | --- |
| Summary: | As a user, I need to be able to use the app to work as intended with no security flaws. |
| Description: | The current Flutter version used in the application from previous teams is Flutter 1.22.5. This is a major concern since Flutter has since released 2 major version updates which include null-safety checking, sunsetting of old iOS versions (iOS 9 and iOS 10), and performance/security improvements. All of these are necessary to ensure the application works as intended and improves the security vulnerability risks. |
| **Status:** | **COMPLETE** |

### 4.3.2 Table 2 - Operating Systems Check User Story

| **1.2 Operating Systems Check** |  |
| --- | --- |
| Summary: | As a user, I need to be able to use the app on the most recent versions of iOS and Android operating systems. |
| Description: | After migrating to Flutter 3.0, we need to ensure we can run the app on simulators for different operating systems. |
| **Status:** | **COMPLETE** |

### 4.3.3 Table 3 - Flutter Facebook Login Package Migration User Story

| **1.3 flutter\_facebook\_login Package Migration** |  |
| --- | --- |
| Summary: | As a user, I need to be able to login to the app using my Facebook profile. |
| Description: | The Flutter package, flutter\_facebook\_login, is no longer compatible with Flutter 3.0 and has not been updated in over 4 years. We need to find a new package that is capable of handling facebook sign-ins (we should probably use firebase\_ui\_oauth\_facebook since it is specifically used in conjunction with Firebase and is built by Google). |
| Status: | **NOT STARTED** |

### 4.3.4 Table 4 - Geoflutterfire Package Migration to Geoflutterfire2 User Story

| **1.4 geoflutterfire Package Migration to geoflutterfire2** |  |
| --- | --- |
| Summary: | As a user, I need specific geolocation features to work so that I can decide where to go and figure out where I am being picked up. |
| Description: | The current geoflutterfire package is incompatible with newer versions of Firebase packages. We need to migrate to the more updated geoflutterfire2 package to handle the dependency issues. To do this, we will need to remove the package flutter\_google\_places and will need to find a replacement or build our own solution. |
| Status: | **COMPLETED** |

### 4.3.5 Table 5 - Find an Alternative to Unicorndial Package User Story

| **1.5 Find an Alternative to unicorndial Package** |  |
| --- | --- |
| Summary: | As a user, I need an intuitive and friendly user interface to interact with. |
| Description: | There is a Flutter package, unicorndial, that is currently being used as a user interface component. However, it has not been updated in 5 years and no longer works with the new Flutter version. We need to find an alternative or build our own component. |
| Status: | **NOT STARTED** |

### 4.3.6 Table 6 - Strip\_Payment Package Migration to Flutter\_Stripe User Story

| **1.6 stripe\_payment Package Migration to flutter\_stripe** |  |
| --- | --- |
| Summary: | As a user, I need a reliable purchasing system to make purchases on the app for the rides I request. |
| Description: | Currently, the app uses the package stripe\_payment to handle payments with the payment provider Stripe. Stripe has migrated to a newly updated package flutter\_stripe. Changes need to be made in the app to handle the new package. |
| Status: | **COMPLETE** |

### 4.3.7 Table 7 - Location Package Update User Story

| **1.7 Location Package Update** |  |
| --- | --- |
| Summary: | As a user, I need the app to request to use my location. |
| Description: | Currently, the app uses the package Geolocator to handle getting the user’s location. The old package is poorly implemented and out of date. Either the current package needs to be updated, or changes need to be made in the app to accommodate a new package. |
| Status: | **COMPLETE** |

### 4.3.8 Table 8 - Figma Mockups User Story

| **2.0 Figma Mockups** |  |
| --- | --- |
| Summary: | As a user, I need an intuitive and friendly interface to interact with. |
| Description: | Currently, the user interface for the app is simply bad. We need to draw up high-fidelity prototypes in Figma to create a better UI/UX system for users. |
| Status: | **COMPLETE** |

### 4.3.9 Table 9 - Implement UI Changes User Story

| **2.1 Implement UI Changes** |  |
| --- | --- |
| Summary: | As a user, I need an intuitive and friendly interface to interact with. |
| Description: | After mocking up prototypes in Figma and getting them approved by the sponsor, we need to implement the features in the code. |
| Status: | **IN PROGRESS** |

### 4.3.10 Table 10 - Driver Directions Research User Story

| **3.0 Driver Directions Research** |  |
| --- | --- |
| Summary: | As a user (driver), I need to be able to see the best route to take customers to their desired destination. |
| Description: | We need to limit driver routes to streets that can accommodate golf carts legally (35 MPH or less) and also be able to show routes that will likely be “blocked” on game days for vehicles. We will have credentials to get through “blocked” roads. Need to take into account cost algorithms when researching.   Complete research of how we can accomplish this.  **UPDATE**: We are going to attempt to use the Google Roads API to get speed limits for the roads being taken by the drivers. However, there is no API or easy way to create routes to a destination using speed limits as a parameter. We have contacted Google Support to see if a developer is willing to speak with us on the matter. |
| Status: | **IN PROGRESS** |

### 4.3.11 Table 11 - Driver Directions Implementation User Story

| **3.1 Driver Directions Implementation** |  |
| --- | --- |
| Summary: | As a user (driver), I need to be able to see the best route to take customers to their desired destination. |
| Description: | We need to limit driver routes to streets that can accommodate golf carts legally (35 MPH or less) and also be able to show routes that will likely be “blocked” on game days for vehicles. We will have credentials to get through “blocked” roads. Need to take into account cost algorithms when researching.   Implementation of research conclusions from user story 3.0 |
| Status: | **NOT STARTED** |

### 4.3.12 Table 12 - Rider ‘On the Way’ Notification User Story

| **3.2 Rider ‘On the Way’ Notification** |  |
| --- | --- |
| Summary: | As a user, I would like a notification about when my driver is going to arrive. |
| Description: | Add a notification for the rider when the driver is on the way to pick them up with an estimated arrival time. Riders should see where the driver is while they are en route. |
| Status: | **IN PROGRESS** |

### 4.3.13 Table 13 - Rider Status User Story

| **3.3 Rider Status** |  |
| --- | --- |
| Summary: | As a user (rider), I would like to be able to see where I am at in my ride on a map as the drive is occurring. |
| Description: | Once a ride has begun, the rider should see the golf cart location on a map as it moves along the drive. |
| Status: | **NOT STARTED** |

### 4.3.14 Table 14 - Rider Recap Screen User Story

| **3.4 Rider Recap Screen** |  |
| --- | --- |
| Summary: | As a user (rider), I would like to see a recap of the ride I just took. |
| Description: | Once a ride is over, a screen should be shown to the rider that summarizes their trip and costs. It should also prompt the user for a rating. |
| Status: | **IN PROGRESS** |

### 4.3.15 Table 15 - Drivers Near Me User Story

| **3.5 Drivers Near Me** |  |
| --- | --- |
| Summary: | As a user (rider), I would like to see where the nearest drivers are around my current location. |
| Description: | Riders should be able to see current locations of carts around them on a map. |
| Status: | **NOT STARTED** |

### 4.3.16 Table 16 - Feedback/Rating System User Story

| **3.6 Feedback/Rating System** |  |
| --- | --- |
| Summary: | As a user, I need to be able to give feedback and rate both driver and riders after interactions with them. |
| Description: | Implement a 5-star rating system. |
| Status: | **COMPLETE** |

### 4.3.17 Table 17 - Enhance Feedback and Tips Screen User Story

| **3.7 Enhance Feedback and Tips Screen** |  |
| --- | --- |
| Summary: | As a user, I need a streamlined and engaging interface for feedback and tips. |
| Description: | The current design doesn't have a feedback screen, which we've found to be less effective. The goal is to design, add a feedback screen and merge it with the tips screen so the rider can be able to rate and tip at the same time. |
| Status: | **COMPLETE** |

### 4.3.18 Table 18 - Display Past Trips User Story

| **3.8 Display Past Trips** |  |
| --- | --- |
| Summary: | As a user, I would like to be able to see all of my previous trips as both a rider and a driver. |
| Description: | Create a driver's past trips log and a rider's past trips log. |
| Status: | **COMPLETE** |

### 