Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2023

Team Corps Catch
Corps Catch

Kevin Le
Kouassi Brou
Oluwaseetofunmi Komolafe
Bijan Saud
Jianliang Liu

Revision History

Revision	Date	Author(s)	Description
0.1	1.23.2023	KL	document creation
0.2	2.12.2023	KL,BS,KB	document revision
1.0	2.12.2023	KL	complete draft

Contents

List of Figures

List of Tables

1 Introduction

The Corps Catch application would make exploring the outdoor trails in a Corps facility more fun. Its goal is to provide entertainment and education to its users, and this is achieved by asking increasingly difficult questions based on the current subject of choice during the hiking trail. If all the questions are answered correctly the player is awarded a digital Corps Coin which they can view in their collection. This reward system is based on the actual coins that outstanding personnel receive in the Corps. The aim of the application is to collect as many coins as possible along the hiking trail by answering questions correctly.

Key requirements for this project include tracking the user's location and have questions pop up when they are within a certain radius of the question, the award of a corp coin if they get all the questions right on the trail, and keeping user information including login and any corp coins they have stored in a database.

2 System Overview

The front-end will consist of the visual aspects of the applications and is essentially everything that the users see and interact with. This layer also handles the inputs and placements of the outputs. In The Corps Catch Application, this layer should allow users to interact with and navigate to all the pages of the app, be visually appealing, and also easy to utilize. The back-end layer will pass information between the front end and the database. Since information from the front end cannot be directed to the database, this layer will be used to communicate between the two. The database layer will store all user information including login information and profile information. This layer will also store questions for each trial, each question will have information including, the question, answer, difficulty, and which trail the question is for.

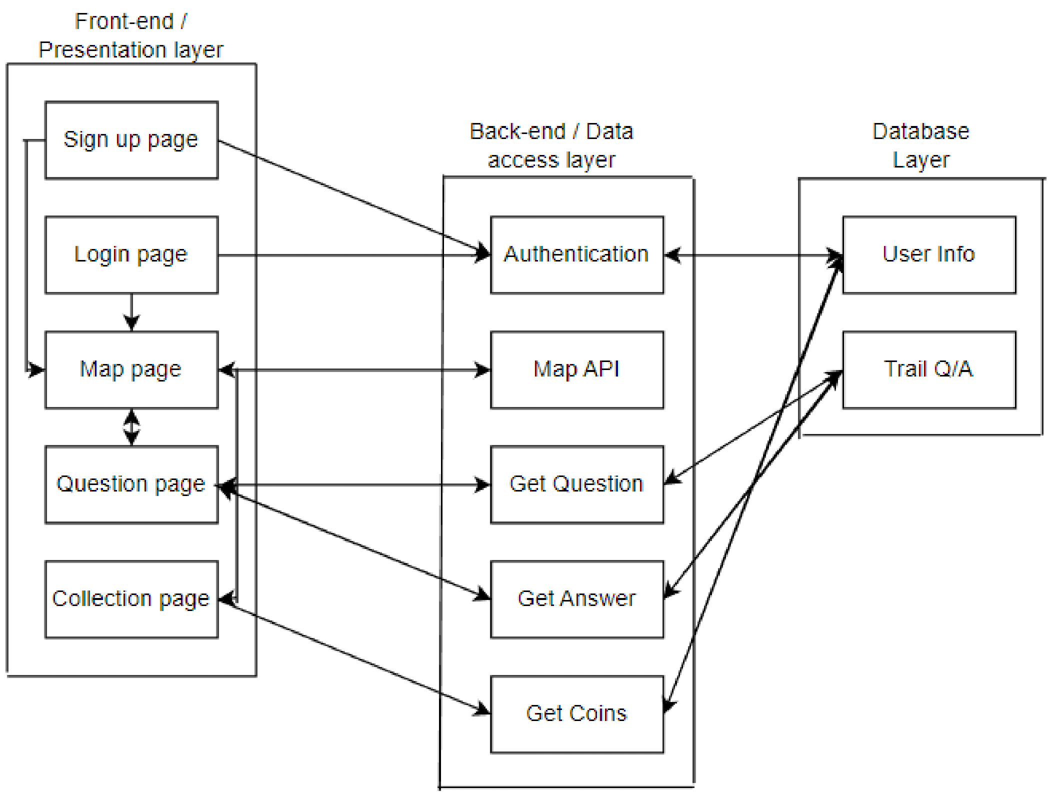

Figure 1: System architecture

3 Front-end / Presentation Layer Subsystems

3.1 Layer Hardware

Android 4.1 (API level 16) or higher is needed to run the application, and for Apple devices iOS 11 and later. The device will also need to be connected to the internet.

3.2 Layer Operating System

For this application the user will have to have either Linux or iOS for their operation system. These will be the only operating systems needed because the app will only be put on the App Store and the Google Play store.

3.3 Layer Software Dependencies

- Firebase Core: 1.10.0
- Flutter SVG: 0.22.0
- Flutter Spinkit: 5.1.0
- Firebase Authentication: 3.1.3
- Cloud Firestore: 3.1.3

3.4 Sign up Page

The purpose of the sign up subsystem within the front-end is to allow new users to create an account within the application. The basic design includes: User Input, This subsystem requires the user to input their name, email, password, and age, Firebase Authentication: This subsystem would connect to the Firebase Authentication service to create a new user account. Firebase would handle the secure storage and management of the user's credentials, User Data Storage: Upon successful creation of an account, the user's data would be stored in the user subsystem of the database.

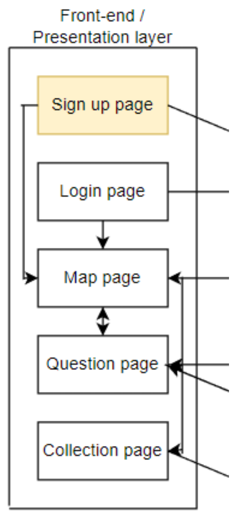

Figure 2: Sign up subsystem description diagram

3.4.1 Subsystem Operating System

Same as overall front-end layer.

3.4.2 Subsystem Software Dependencies

Same as overall front-end layer.

3.4.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and Flutter-Flow.

3.4.4 Subsystem Data Structures

This subsystem connects to the Firebase Authentication service and map page (once signed up user is directed to map page, and stores the user's data in the database.

3.4.5 Subsystem Data Processing

We query the data given by the user to validate and create the account for the user.

3.5 Login Page

The purpose of the login subsystem is responsible for allowing existing users to access their account. The basic design would include the user's input as described in the sign up but just email and password, Firebase Authentication to verify the user exists, User data retrieval: once logged in the user's data would be retrieved and the user can see their earned coins.

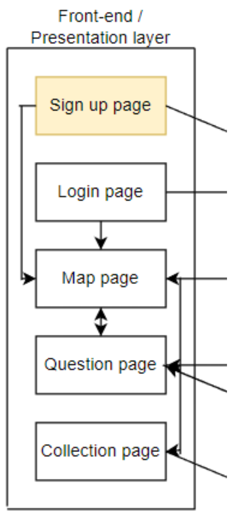

Figure 3: Login subsystem description diagram

3.5.1 Subsystem Operating System

Same as overall front-end layer.

3.5.2 Subsystem Software Dependencies

Same as overall front-end layer.

3.5.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and Flutter-Flow.

3.5.4 Subsystem Data Structures

This subsystem connects to the Firebase Authentication service and map page (once logged in page directs to map page), and stores the user's data in the database.

3.5.5 Subsystem Data Processing

We query the data given by the user to validate and create the account for the user.

3.6 Map Page

The purpose of the map subsystem provides a visual representation the user's location. The basic design would include the Map API Integration in which this subsystem connects with a map API to retrieve and display the user's location on the map, User Location Tracking, Question Pop-Ups as this subsystem would also display pop-up questions as the user moves through the trail.

3.6.1 Subsystem Operating System

Same as overall front-end layer.

3.6.2 Subsystem Software Dependencies

Google Maps Flutter: 2.1.1

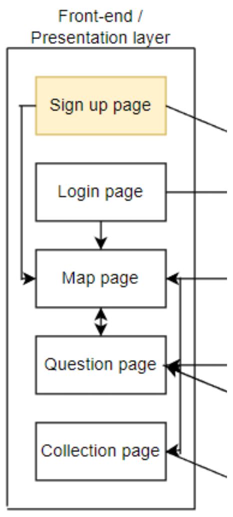

Figure 4: Map subsystem description diagram

3.6.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and Flutter-Flow.

3.6.4 Subsystem Data Structures

This subsystem connects to the collection page and is connected to the get question subsystem in the back-end and would vary based on the user's location within the trail.

3.6.5 Subsystem Data Processing

We query the database for the appropriate question as the user goes through the trail, and we are still working on the algorithm to detect when to make the question pop up whenever the user gets within a certain distance from the question's location.

3.7 Question Page

The purpose of the map subsystem is to display the questions to the users as they traverse the trails. The basic desigen of this subsystem involves utilizing FlutterFlow pro and Flutter to display the questions to the user and to receive user inputs. The subsystem would also connect to the "get question" and "get answer" back-end subsystems to retrieve the questions and answers. When the user is near a location where a question is set to pop up, the question subsystem would retrieve the question and display it to the user. The user would then provide an answer, which would be checked against the correct answer stored in the "get answer" subsystem. If the user provides the correct answer, they would be awarded a coin, which would be tracked in the collection subsystem.

3.7.1 Subsystem Operating System

Same as overall front-end layer.

3.7.2 Subsystem Software Dependencies

Same as overall front-end layer.

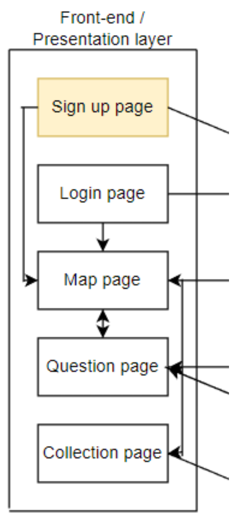

Figure 5: Question Page subsystem description diagram

3.7.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and FlutterFlow.

3.7.4 Subsystem Data Structures

This subsystem connects to the "get question" and "get answer" back-end subsystems to retrieve the questions and answers.

3.7.5 Subsystem Data Processing

We query the database for the appropriate question and get the answer as well then we compare the user's answer to the one in the database.

3.8 Collection Page

The purpose of the collection page subsystem is to manage and display the collection of coins that a user has earned. The basic design of this subsystem would involve connecting to the "get coins" subsystem in the back-end, which would provide information about the coins that the user has earned. The collection subsystem is an important component of the overall app design, as it allows the user to see their progress and keep track of their achievements.

3.8.1 Subsystem Operating System

Same as overall front-end layer.

3.8.2 Subsystem Software Dependencies

Same as overall front-end layer.

3.8.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and FlutterFlow.

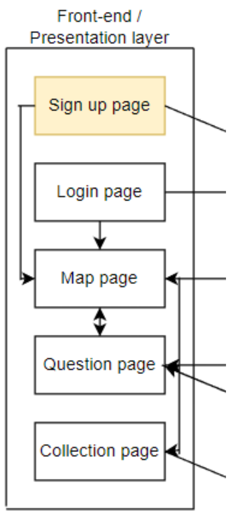

Figure 6: Collection Page subsystem description diagram

3.8.4 Subsystem Data Structures

This subsystem connects to the get coins subsystem in the back-end.

3.8.5 Subsystem Data Processing

We query the database for the user's coins, and display it to the user.

4 Back-end / Data Access Layer Subsystems

4.1 Layer Hardware

The primary component used is a Mobile Device or a tablet, with GPS, gyroscope and accelerometers embedded. Android 4.1 (API level 16) or higher is needed to run the application, and for Apple devices iOS 11 and later. The device will also need to be connected to the internet.

4.2 Layer Operating System

For this application the user will have to have either Linux or iOS for their operation system. These will be the only operating systems needed because the app will only be put on the App Store and the Google Play store.

4.3 Layer Software Dependencies

Firebase Authentication: This library would be used to provide user authentication and authorization, Firebase Realtime Database: This library would be used to store the data related to trails, questions, and user information in the database, and FlutterFlow SDK: this library provides a set of Flutter plugins for the FlutterFlow platform, including authentication, database, and storage.

- Firebase Core: 1.10.0
- Firebase Authentication: 3.1.3
- Cloud Firestore: 3.1.3

4.4 Authentication

The purpose of the authentication subsystem within the back-end using Firebase and Flutter would be to provide secure and reliable user authentication and authorization. This subsystem would authenticate users using Firebase Authentication by communicating with the database. For part of the design we have authorization and access control: This subsystem would also implement access control to ensure that only authorized users can access specific data and functionality within the app. The authentication subsystem would also provide a user interface for users to sign up, log in, and manage their account information, such as resetting passwords or updating profile information.

4.4.1 Subsystem Operating System

Same as overall back-end layer operating system.

4.4.2 Subsystem Software Dependencies

A description of any software dependencies (libraries, frameworks, design software for mechanical parts or circuits, etc) required by the subsystem.

4.4.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and Flutter-Flow.

4.4.4 Subsystem Data Structures

A description of any classes or other data structures that are worth discussing for the subsystem. For example, data being transmitted from a micro controller to a PC via USB should be first be assembled into packets. What is the structure of the packets?

4.4.5 Subsystem Data Processing

Query the database to see if user exists or not, or if the password matches the user.

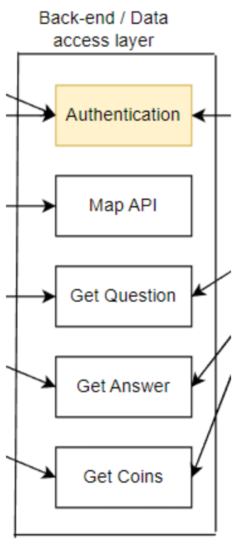

Figure 7: Authentication subsystem description diagram

4.5 Map API

The purpose of the Map API subsystem within the back-end is to provide a map and location-based services for the app. The map API subsystem will interact with the questions in the database to determine the user's location and trigger the display of questions based on the user's current location. The basic design of the Map API subsystem will involve using the Google Maps API to display a map of the trail and track the user's location as they move around the trail. As the user moves, the subsystem will use the user's location data to trigger the display of questions at specific locations.

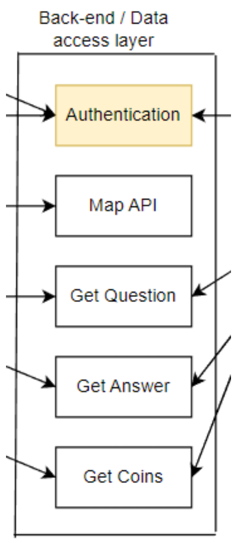

Figure 8: Map API subsystem description diagram

4.5.1 Subsystem Operating System

Same as overall back-end layer operating system.

4.5.2 Subsystem Software Dependencies

Google Maps API: This API provides map and location-based services for the app, such as displaying a map of the trails and showing the user's location on the map.

- Geolocator: 8.0.1
- Google Maps Flutter: 2.1.1

4.5.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and Flutter-Flow.

4.5.4 Subsystem Data Structures

This Map API will work with the class with questions to know what question to prompt at the specific location.

4.5.5 Subsystem Data Processing

We are tracking and displaying the user's location by getting the coordinates of the user and displaying it on the map.

4.6 Get Question

This subsystem is responsible for retrieving the appropriate question for the user based on their current location. It receives the user's current location from the map API subsystem and uses that information to determine which trail the user is on and which question they should receive. The subsystem then queries the question subsystem in the database to retrieve the appropriate question and returns it to the front-end for display. The basic design of this subsystem involves a function or API that receives the user's location, retrieves the appropriate question from the database, and returns it to the front-end.

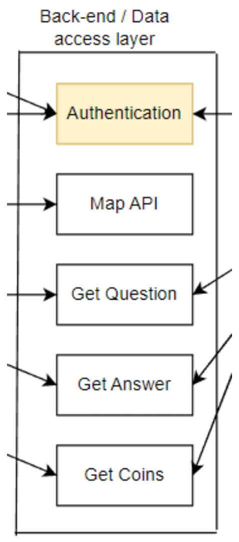

Figure 9: Get Question subsystem description diagram

4.6.1 Subsystem Operating System

Same as overall back-end layer operating system.

4.6.2 Subsystem Software Dependencies

Same as overall back-end layer software dependencies.

4.6.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and FlutterFlow.

4.6.4 Subsystem Data Structures

This subsystem will work with the Map API to get the location, and prompt the appropriate question.

4.6.5 Subsystem Data Processing

We are retrieving questions from the database by querying.

4.7 Get Answer

The purpose of this subsystem is to provide the correct answers for the questions that are asked to the users during their trails. When the user submits an answer, this subsystem will validate the answer and determine if it is correct or not. The basic design of this subsystem involves retrieving the questions and their corresponding answers from the database, comparing the submitted answer with the correct answer, and returning a response to the user indicating whether the answer is correct or not. The subsystem will also keep track of the number of correct answers for each user and update the user's progress accordingly.

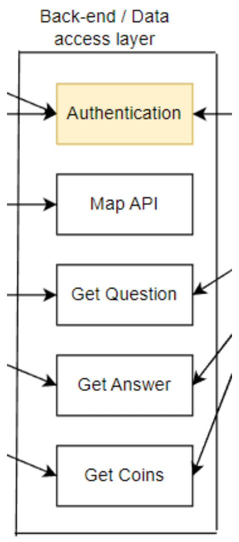

Figure 10: get Answer subsystem description diagram

4.7.1 Subsystem Operating System

Same as overall back-end layer operating system.

4.7.2 Subsystem Software Dependencies

Same as overall back-end layer software dependencies.

4.7.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and Flutter-Flow.

4.7.4 Subsystem Data Structures

This subsystem will work with the User and Question classes to get the correct answer and to keep track of how many questions the user has right.

4.7.5 Subsystem Data Processing

We are retrieving answers with no specific algorithm.

4.8 Get Coins

The purpose of this subsystem is to manage and keep track of the coins earned by the user. The basic design of this subsystem includes a database that stores information about the user's earned coins and an API that allows the front-end to retrieve and display the coins earned by the user on the collection page. The subsystem will also handle any updates to the user's coin collection as the user earns additional coins. The overall goal of this subsystem is to incentivize users to explore the different trails and complete the various trails by earning unique coins.

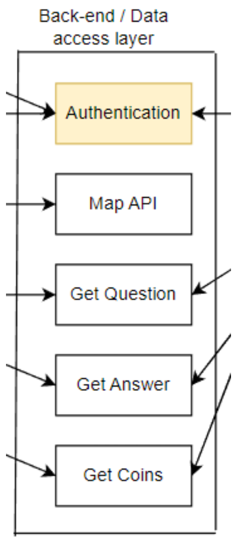

Figure 11: Get Coins subsystem description diagram

4.8.1 Subsystem Operating System

Same as overall back-end layer operating system.

4.8.2 Subsystem Software Dependencies

Same as overall back-end layer software dependencies.

4.8.3 Subsystem Programming Languages

Dart programming language: This is the primary programming language used by Flutter and Flutter-Flow.

4.8.4 Subsystem Data Structures

This subsystem will work with the collection page in the front-end and the user class to get the coins earned by the user.

4.8.5 Subsystem Data Processing

We are retrieving the user's coin by querying from the database.

5 Database Layer Subsystems

5.1 Layer Hardware

Android 4.1 (API level 16) or higher is needed to run the application, and for Apple devices iOS 11 and later. The device will also need to be connected to the internet.

5.2 Layer Operating System

For this application the user will have to have either Linux or iOS for their operation system. These will be the only operating systems needed because the app will only be put on the App Store and the Google Play store.

5.3 Layer Software Dependencies

The following dependencies must be added to the pubspec.yaml file: firebase_core (The core Firebase SDK), cloud_firestore (A Flutter plugin that provides an API for interacting with Cloud Firestore, a NoSQL document database that's part of Firebase), firebase_auth (Flutter plugin that provides an API for authenticating users with Firebase Authentication.), and firebase_storage (A Flutter plugin that provides an API for uploading and downloading files to/from Firebase Cloud Storage) (versions will be updated as we get further into development)

5.4 User Info Subsystem

The question subsystem would be responsible for managing and presenting questions to the user on each trail. The question subsystem would store question data in the Firebase Cloud Firestore database and associate questions with specific trails. The question subsystem would also provide a the question for the user interface for answering questions and receiving feedback on whether the answer is correct. When a user answers all the questions correctly on a trail, the question subsystem would trigger the user subsystem to award the user with a unique coin associated with that trail.

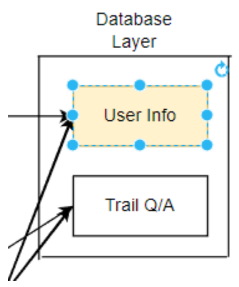

Figure 12: User Info subsystem description diagram

5.4.1 Subsystem Operating System

The operating systems needed are iOS for Apple Users, and Linux for Android users (stated in overall layer OS).

5.4.2 Subsystem Software Dependencies

Same as overall database layer software dependencies.

5.4.3 Subsystem Programming Languages

Dart for both Apple and Android app versions to implement Firebase into the app.

5.4.4 Subsystem Data Structures

The user class will contain the user's name (string), age (integer), email(string), password (string), and corp coins (array of strings) earned.

5.4.5 Subsystem Data Processing

There is no algorithm being used to store or extract data we will simply be querying data when needed.

5.5 Trail Q/A Subsystem

The purpose of this user info subsystem is to authorize and store data into the database. For user authorization users would need to sign up and log in to use the app. Firebase Authentication will used to provide a secure way to authenticate users and store their credentials. In regard to storing data into the database, user data including name, email password, and age, and total number of coins collected, would be stored in the Firebase Cloud Firestore database.

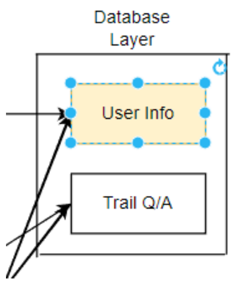

Figure 13: Trail Q/A subsystem description diagram

5.5.1 Subsystem Operating System

The operating systems needed are iOS for Apple Users, and Linux for Android users (stated in overall layer OS).

5.5.2 Subsystem Software Dependencies

Discussed in overall layer database subsystem

5.5.3 Subsystem Programming Languages

Dart for both Apple and Android app versions to implement Firebase into the app.

5.5.4 Subsystem Data Structures

The question class will contain the trail question (string), trail id (int), and trail answer(string), and question difficulty (int).

5.5.5 Subsystem Data Processing

There is no algorithm being used to store or extract data we will simply be querying data when needed.

6 Appendix A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

References