Assassins App

Corrissa Smith
Cedarville University, corrissasmith@cedarville.edu

Hannah Oaisa
Cedarville University, hannahoaisa@cedarville.edu

Dilean Munoz
Cedarville University, dileanmunoz@cedarville.edu

Henry Anderson
Cedarville University, henryanderson@cedarville.edu

Assassins Mobile App

Faculty Advisor: *Dr. David Gallagher*

Students: *Henry Anderson, Dilean Muñoz, Hannah Oaisa, Corrissa Smith*

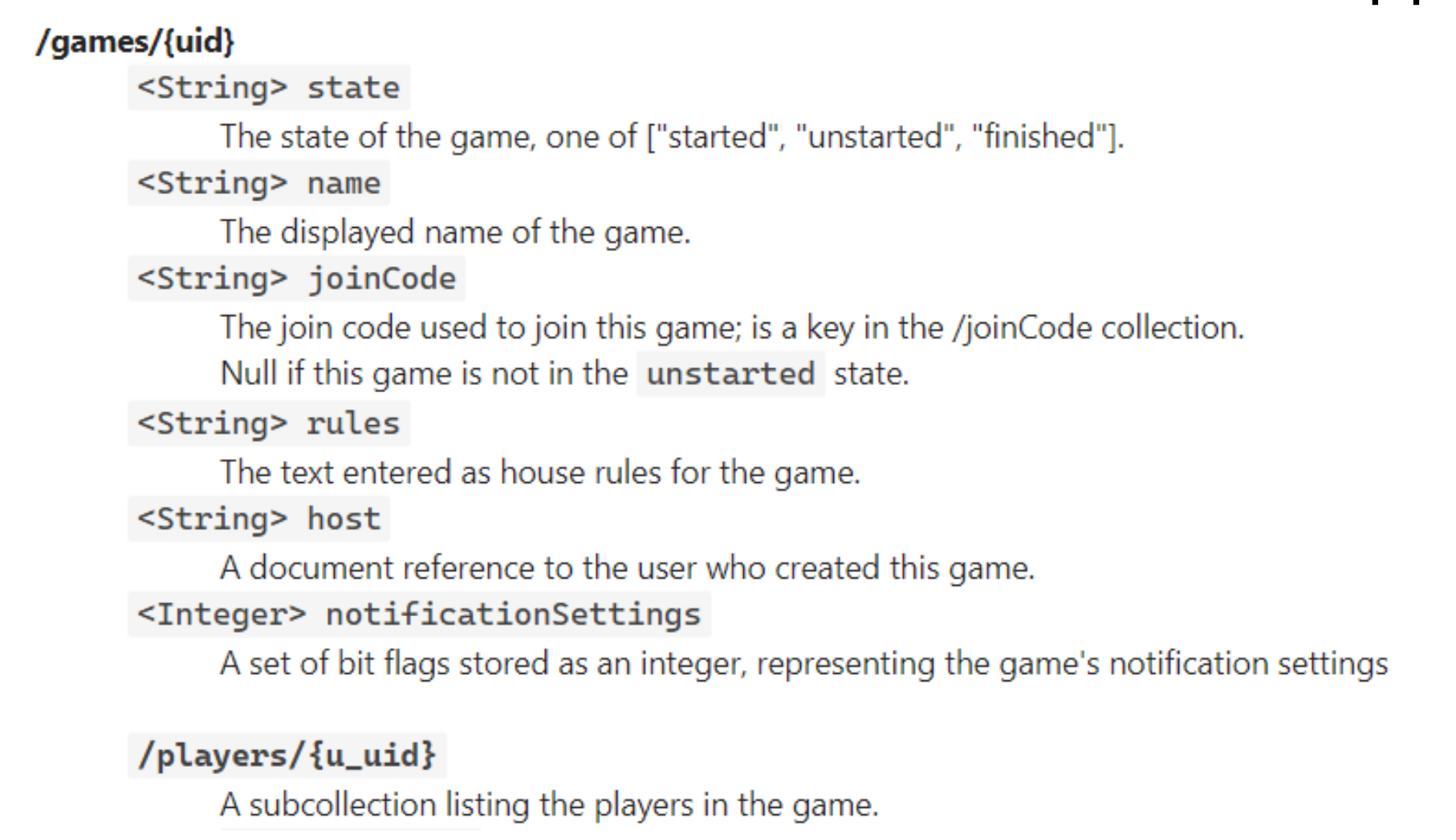
Contact Us: assassinsappbeta@gmail.com

Technical Architecture

The Assassins app is created using Flutter and Google's Cloud Firestore database. It consists of four main "layers": the database (in Cloud Firestore), the database API (implemented in the app using the FlutterFire library), the app objects (an in-app abstraction of the data), and the UI. These layers are kept as separate as possible to modularize the code.

Cloud Firestore

Cloud Firestore is a document-based database. Unlike a typical relational database, which stores data in tables associated by various unique-ids, data is stored in a hierarchical format of collections and documents (loosely similar to the idea of folders and files). As a result, data is stored in a very JSON-like format, mimicking how the app objects will store data. Cloud Firestore is external to the app.



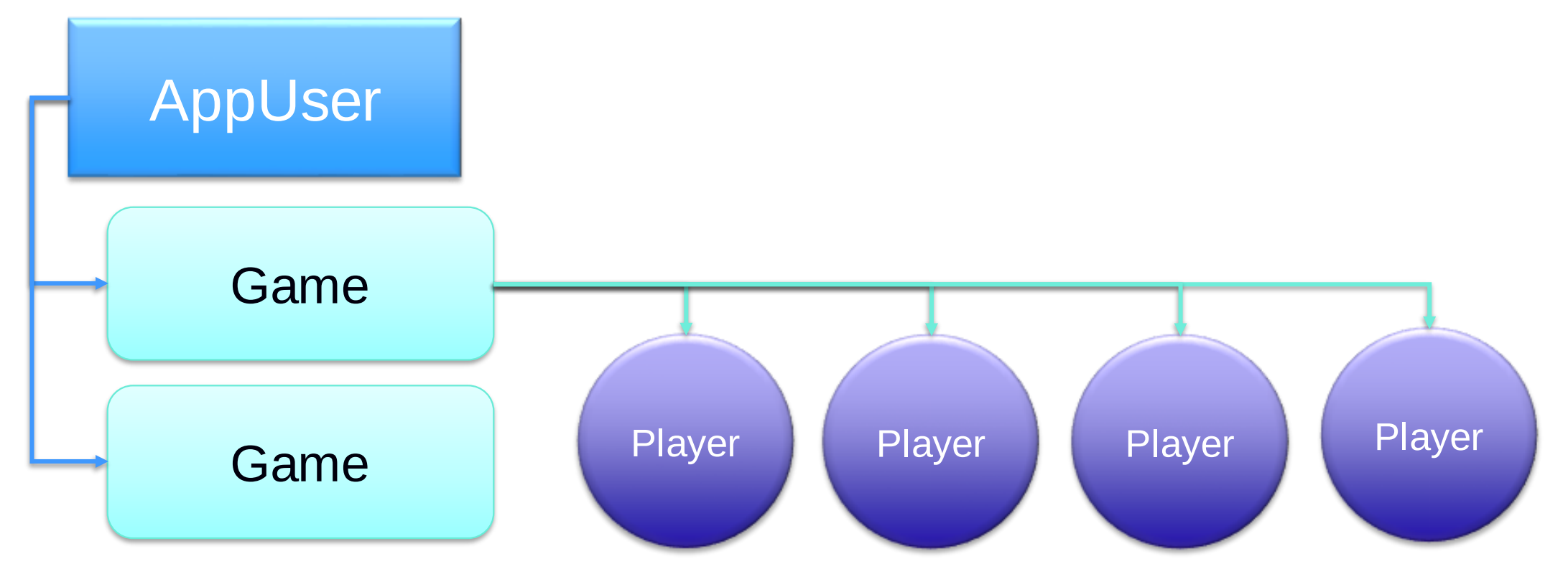
```
/games/{uid}
    <String> state
        The state of the game, one of ["started", "unstarted", "finished"].
    <String> name
        The displayed name of the game.
    <String> joinCode
        The join code used to join this game; is a key in the /joinCode collection.
        Null if this game is not in the unstarted state.
    <String> rules
        The text entered as house rules for the game.
    <String> host
        A document reference to the user who created this game.
    <Integer> notificationSettings
        A set of bit flags stored as an integer, representing the game's notification settings

    /players/{u_uid}
        A subcollection listing the players in the game.
```

A sample of the database's schema

Database API

Within the app's codebase are functions that control reading and writing from the database. These functions ensure that data is updated in real time by attaching listeners to Firestore documents. When a listener is triggered, the database's listener code passes the new data to app objects and pushes a global "update" ping to the UI. With a relational database, some sort of ORM (object-relational mapping) framework could have automated some of this API, but because we used a document-based database, we constructed these functions ourselves. The FlutterFire library provides good support for reading, writing, and listening to Cloud Firestore.

App Objects



An illustration of the app objects and their relationships

The app objects store all the relevant information about players and games. They call database API functions as necessary to read and write data from the database. However, because of the API's abstraction, they remain agnostic as to what the database really is and how these reads and writes actually work.

UI

The UI is built using Flutter's stateful widgets. These widgets are given the app objects (ie, games and players) they are supposed to show. The widgets are subscribed to a global update stream that informs the widgets if any app object's data has changed. That way, the UI does not deal with the details of an app's data changing—it is simply instructed to refresh the page.

Abstract

The Assassins mobile app is a tool created to facilitate the popular campus game of Assassins, common to Cedarville University and other youth communities. In a game of assassins, players are assigned targets in a single, randomly ordered loop, and "assassinate" each other with a designated "weapon" (potentially a plastic spoon, nerf gun, or water gun). Once a player assassinates his target, he is assigned his former target's target, and assassinations continue until only the winner remains. Assassins games can vary widely in duration, ranging from half an hour to several weeks. Traditional tools for managing game information and distributing target lists include spreadsheets, emails, and group messaging. As a result, creating and distributing the target list can be time-consuming, and short-duration assassins games are comparatively more work to organize and are therefore less common. To address the hassle in organizing games of assassins, the Assassins app keeps all game information on one platform and allows users to create, join, play, and moderate games of assassins. The app is designed using Flutter, a cross-platform language, to enable release to both Apple and Android platforms. It is backed by a Google Firestore database, and the back-end code is designed with modularity in mind to make it easier to maintain and update. By streamlining and enhancing the management of Assassins games, this app will simplify the game-playing experience and make Assassins more accessible

Development and Deployment

Language

The Assassins app is built in the language Flutter, which is based on Dart. Flutter is a cross-platform language, meaning the app does not have to be rewritten for Apple/iOS and Android devices.

Development Environments

Flutter is a cross-platform language, but it still needs to be compiled for each platform. As the Android platform allows developers much more ease of access (iOS development requires a Mac computer and is much more difficult to test on a real device), most of our development was done in Android Studio. One member of the Assassins team regularly tested the developing app on iOS using Xcode. Most cross-platform difficulties involve connecting the app to outside resources, such as the database and notification streams. Otherwise, there have been very few cross-platform problems.

Deployment

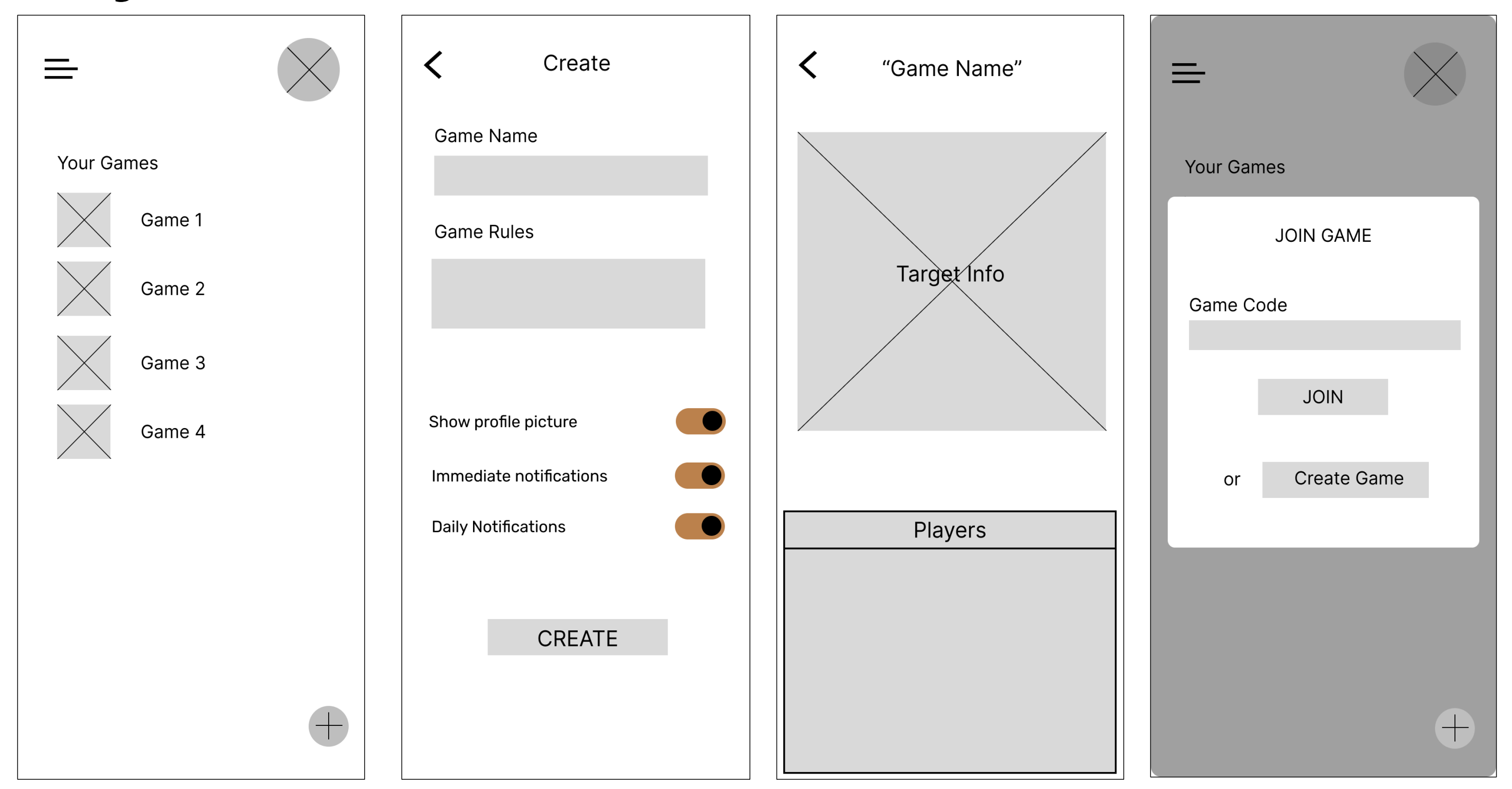
Deployment of the Assassins app is currently in its beta phase. Deploying a beta app to Android devices is relatively simple: an .apk file can be distributed directly to beta testers and installed by them. (Beta deployment can also be done through the Google Play Store, after the app and developer are registered). Deploying a beta app on iOS is much more difficult and requires an Apple developer license. To fund this license, an alpha version of the app was taken to Cedarville University's entrepreneurial contest, The Pitch, where it won second place.

Anticipated Impact

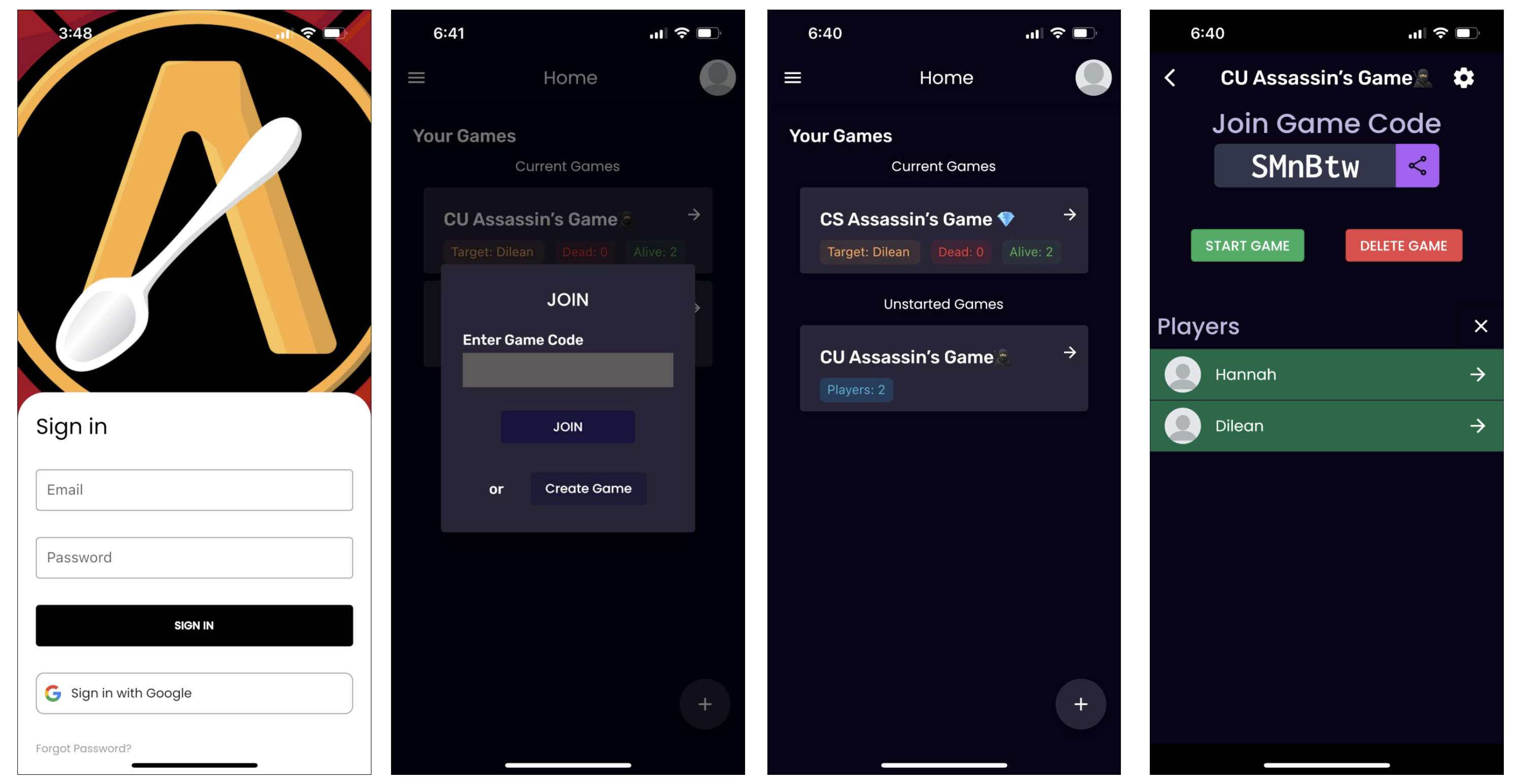
We hope for the Assassins app to simplify the experience of playing assassins, making the game more accessible and streamlined. Short-duration games may become more common, as the app eliminates the work required to organize them. We believe that technology and games should be used to connect people and support real-life interactions, rather than replace them.

User Interface

Wireframe



Mockups/High-Fidelity Designs



Download the Beta

Apple Instructions

Step 1. Install TestFlight
Step 2. Scan QR code
Step 3. Join the beta and install the app!

Android Instructions

Step 1. Scan QR code
Step 2. Tap on the .apk file
Step 3. When prompted, go to settings and enable the installation. RE-ENABLE THIS SETTING LATER.
Step 4. Install app

Note: The app only supports sign-in with Google