Hot Air Final Presentation

By Titus Biel, Tyler Tejera, and John Reynolds

Project Recap

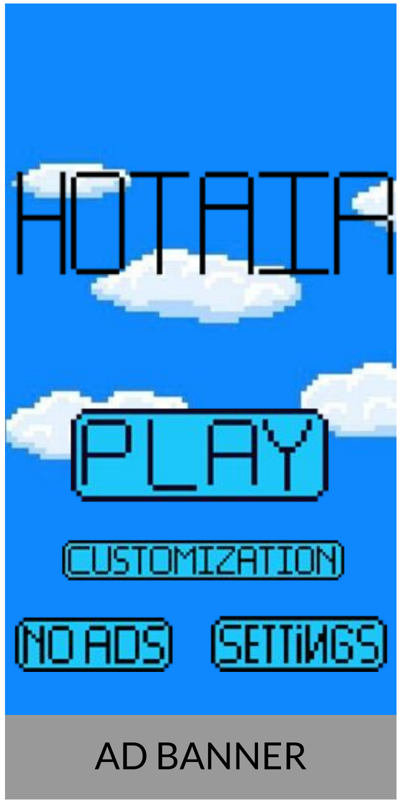
Project Conducted During CSC-431 This Semester

- This project focuses on a free-to-play mobile game
  - Players use swipes to control a balloon, avoid obstacles, and collect coins used for rewards
- Advertisements will be included in the game
  - These can be removed with a remove ads option connected to the device's App Store
- Our goal is to create a simple, quick and easy to play, mobile game
  - Think of users on the go, commuting, waiting in at the store, etc.

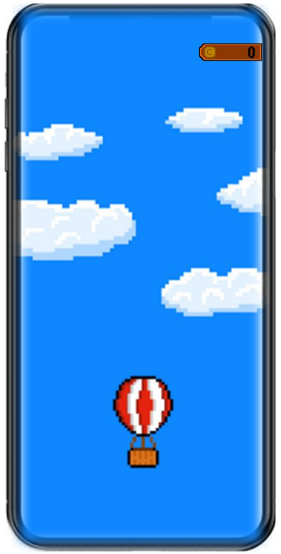
Project Recap Continued

- Our App is intended for mobile devices, and will be distributed on the Apple App Store and Google Play Store
- Hot Air will include a main menu, gameplay, and customization screen
- Much of the application will be coded using C# and Java
  - Firebase will be used for the backend, while Unity will be the primary development tool for the game itself and its UI
- Hot Air requires internet for users who want to remove advertisements
  - Gameplay will always be available offline

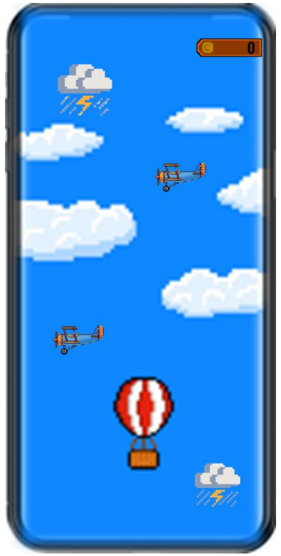
Main Menu Concept and Prototype



Gameplay Concepts and Prototypes

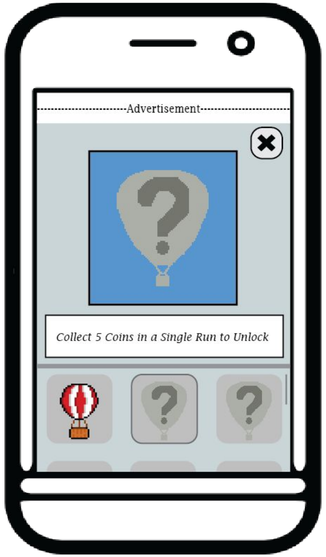


Without
Obstacles

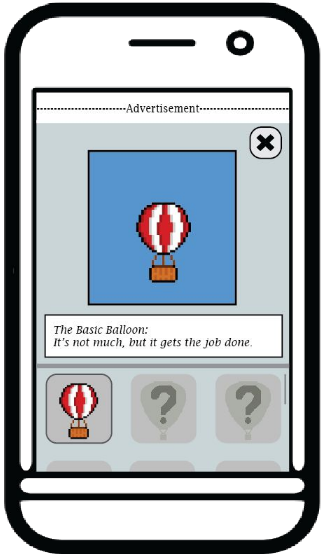


With
Obstacles

Customization Concepts and Prototypes

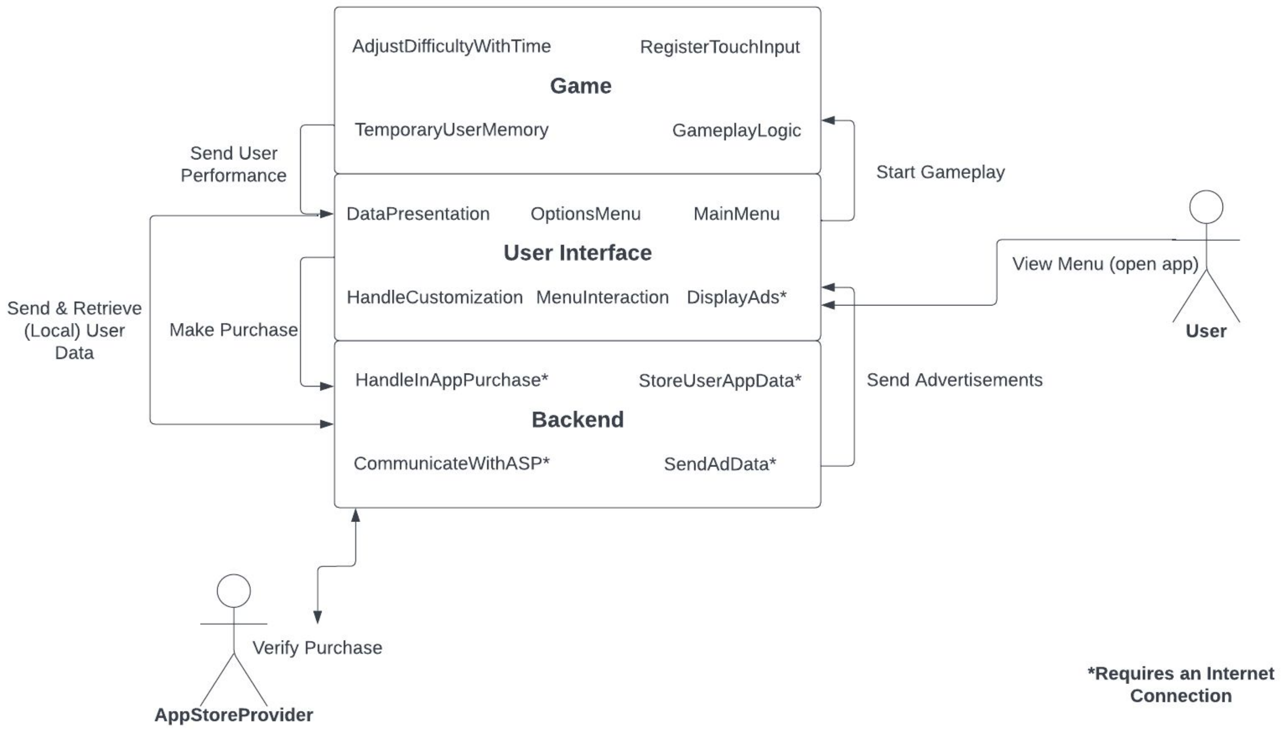


Without Unlock



With
Unlock

Hot Air System Diagram



***Requires an Internet Connection**

Actors

- User
  - The vast majority of interactions with the system
    - Gameplay, customization, menu selections, etc.
- App Store or Google Play Server
  - Process our in-App purchases
    - The Ad-free purchase option
  - Needs data from our App to complete the transaction

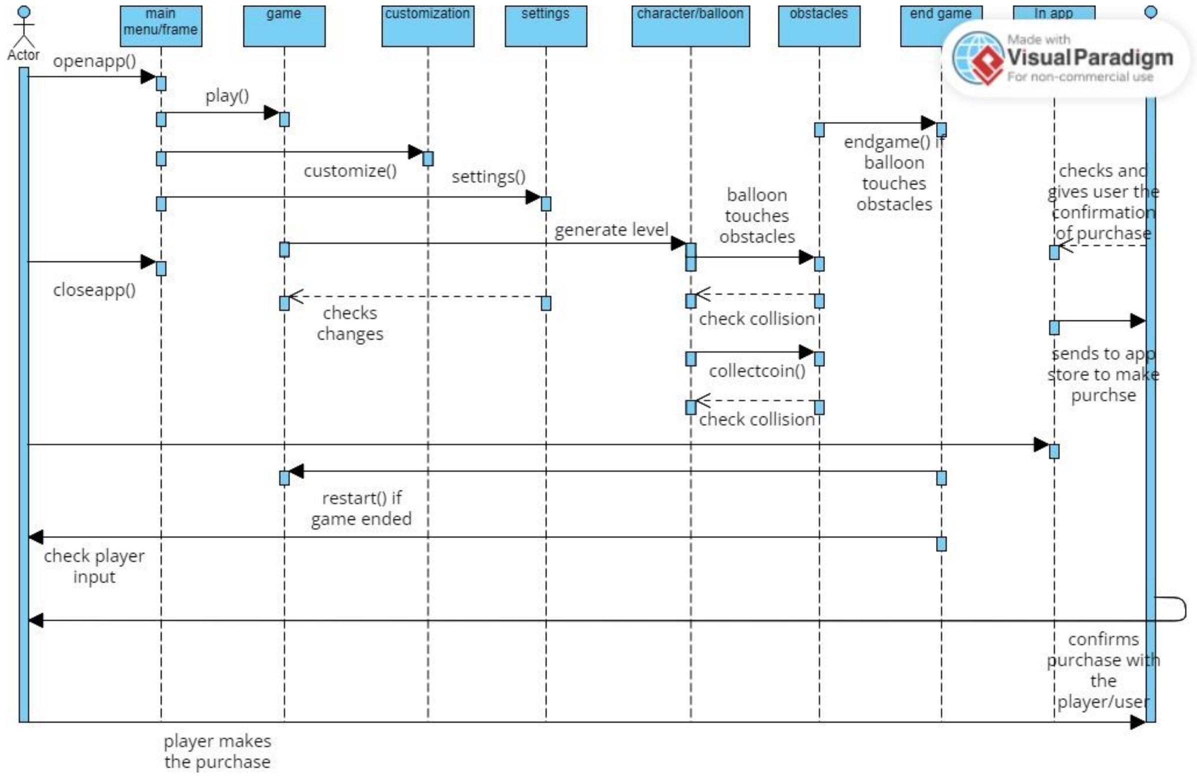
Design Patterns

- Most of the framework and tools have built-in compatibility
    - Focus is on making each layer as efficient as possible
- Factory Method will be used for Gameplay Layer
    - Intend to make game generation quick and efficient
    - Levels will be infinite and get progressively more difficult as they continue
- Facade will be used for the UI/Menu Layer
    - May require data transfer (due to the App Store)
    - Will include visual menus and also advertisements
- Back-end will require some adaptors
    - As mentioned previously, the back-end will mainly be coded in C#
    - Adaptors will ensure proper communication and transfer of data with the UI and App Store or Google Play Store if necessary

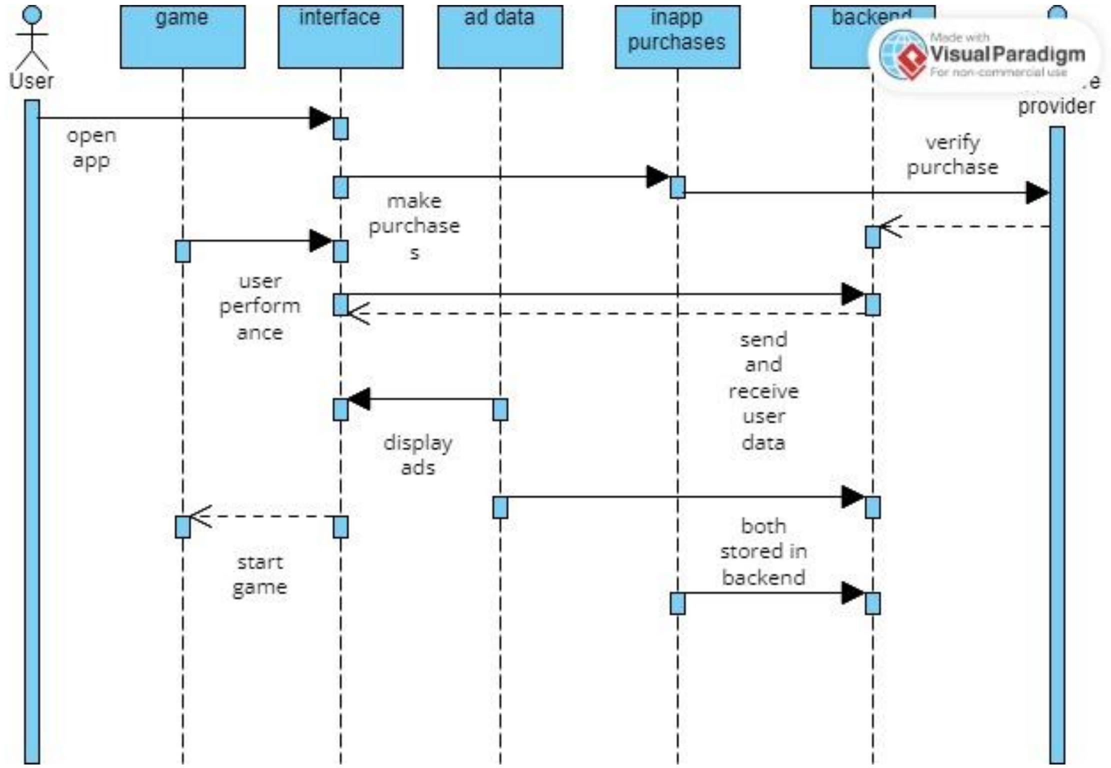
Framework

- Unity Game Engine will be used for the Gameplay and UI Layers
    - It is a known product with a good reputation, and is free to use while the game has produced less than $100,000 in revenue
    - Includes built-in compatibility with iOS and Android
- Firebase will be used by the back-end
    - Hosted by Google, and also includes built-in capacity for iOS and Android
    - Uses Java, which is a familiar language to each member of the development team
    - Compatible with Unity through SDKs, making it easy to work with while we build the Gameplay and UI layers

Gameplay and Interface Diagram


Actor
main menu/frame
game
customization
settings
character/balloon
obstacles
end game
In app
Made with Visual Paradigm For non-commercial use
openapp()
play()
customize()
settings()
generate level
balloon touches obstacles
endgame() if balloon touches obstacles
checks and gives user the confirmation of purchase
closeapp()
checks changes
check collision
collectcoin()
check collision
sends to app store to make purchse
restart() if game ended
check player input
confirms purchase with the player/user
player makes the purchase

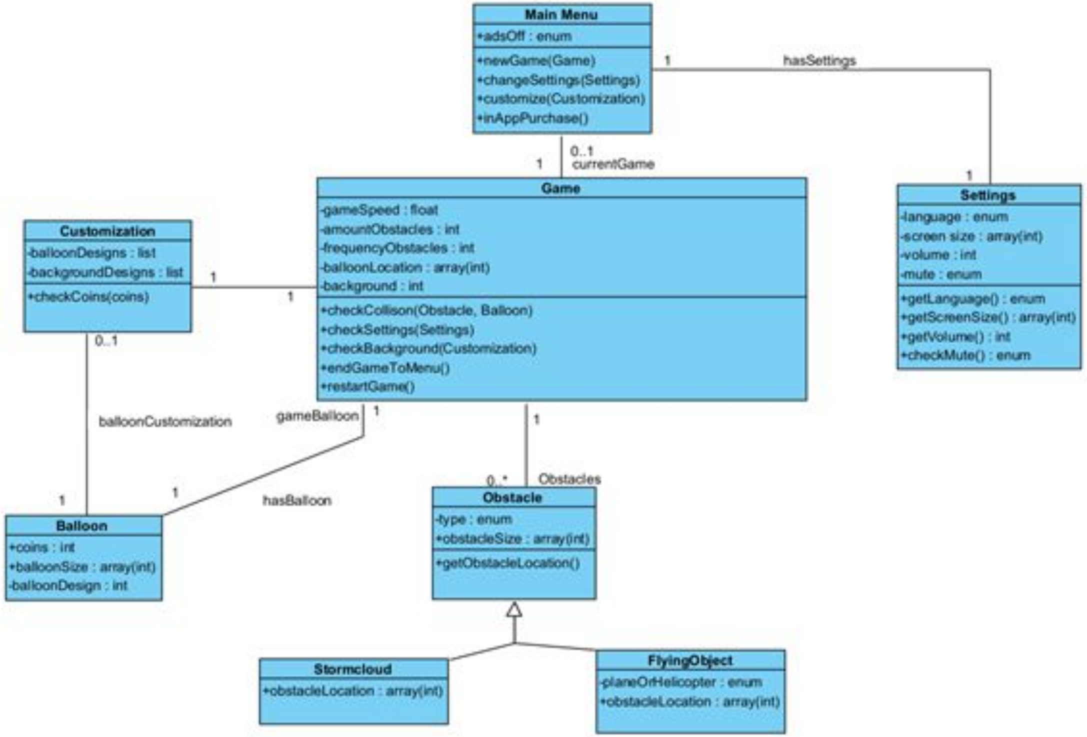
Back-end and App Store Diagram



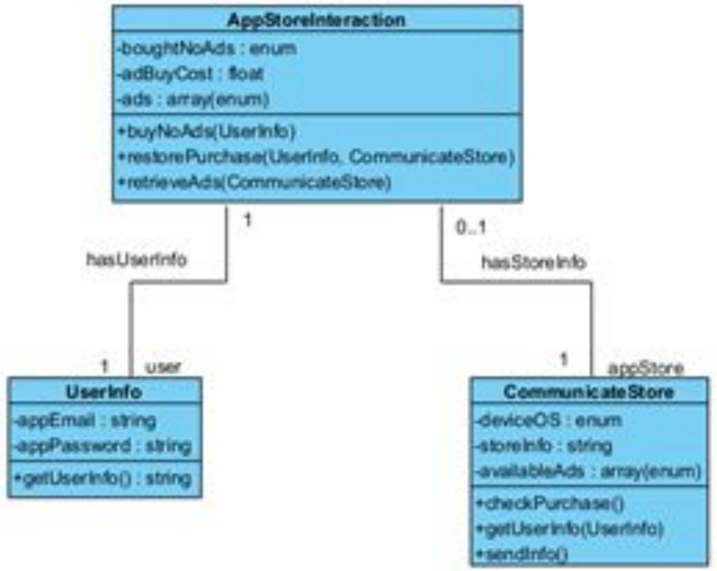
The Class Diagram

- Follows over the next few slides
- The front-end focuses on the Game class
  - Obstacle information, information on the balloon asset (needed due to customization), and settings all flow to it or are directly connected
- The back-end focuses on interactions with the pertinent App Store
  - Contains AppStoreInteraction, which knows the price of the purchase, what Ads need to be in the game, and whether or not the user has bought the Ad-free option
  - Also contains areas for user information and communication with the App Store

Hot Air Front-end Class Diagram



Hot Air Back-End Class Diagram


AppStoreInteraction
-boughtNoAds : enum
-adBuyCost : float
-ads : array(enum)
+buyNoAds(UserInfo)
+restorePurchase(UserInfo, CommunicateStore)
+retrieveAds(CommunicateStore)
1
hasUserInfo
1
user
0..1
hasStoreInfo
1
appStore
UserInfo
-appEmail : string
-appPassword : string
+getUserInfo() : string
CommunicateStore
-deviceOS : enum
-storeInfo : string
-availableAds : array(enum)
+checkPurchase()
+getUserInfo(UserInfo)
+sendInfo()