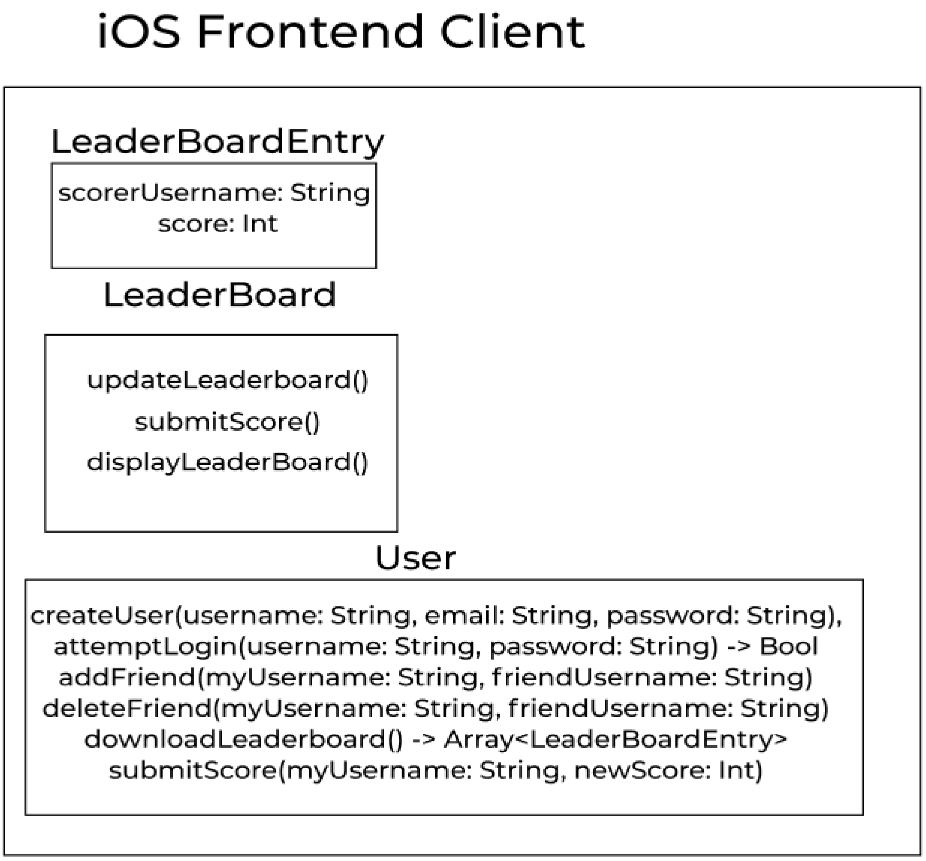
Final Project Deliverable #3 - Full Design Document

Application: Arcade Hoops

Nirvaan Reddy, Sam Donovan, Christopher Cooper, Teresa Tran, Kameron Shahabi, Natalie Shamilian

Frontend



iOS Frontend Client

LeaderBoardEntry

scorerUsername: String
score: Int

LeaderBoard

updateLeaderboard()
submitScore()
displayLeaderBoard()

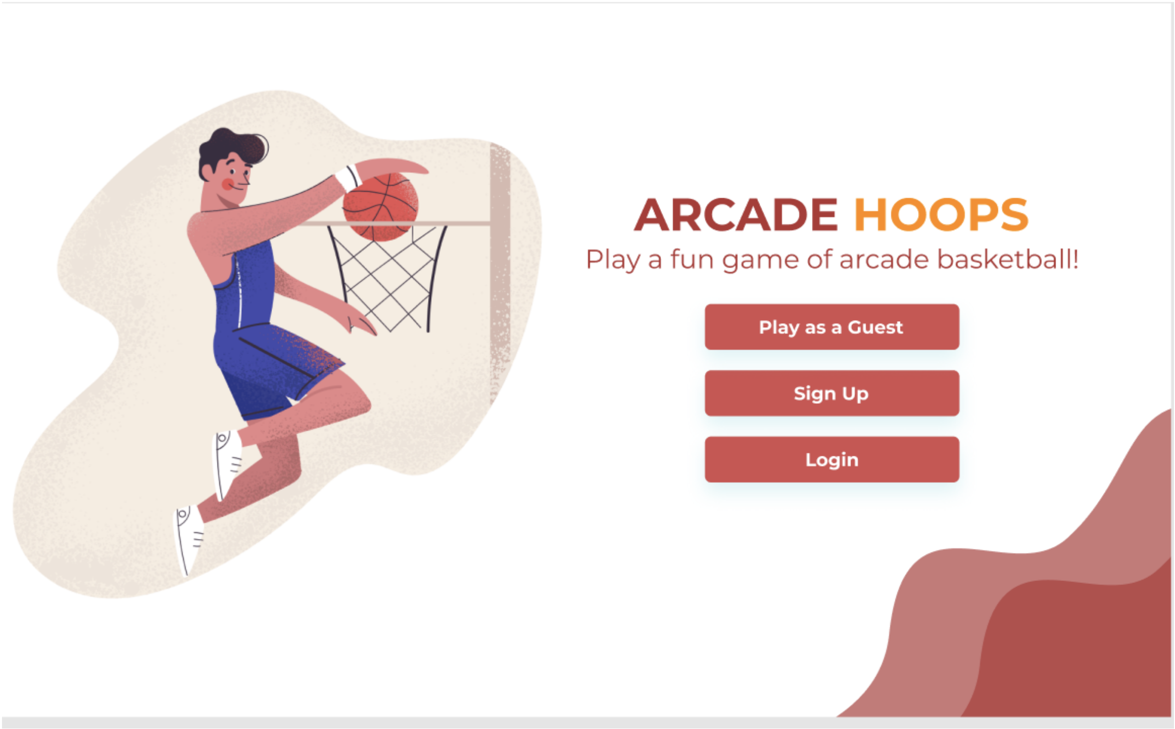
User

createUser(username: String, email: String, password: String),
attemptLogin(username: String, password: String) -> Bool
addFriend(myUsername: String, friendUsername: String)
deleteFriend(myUsername: String, friendUsername: String)
downloadLeaderboard() -> Array<LeaderBoardEntry>
submitScore(myUsername: String, newScore: Int)

<u>Initial Screen/Login Screen</u>

- Asks user to sign in, login, or to play as a guest
- Once signed in, take user to screen with two buttons: Play the game or View the leaderboard
- If logged in, scores will be saved and have a chance to be placed on the leaderboard

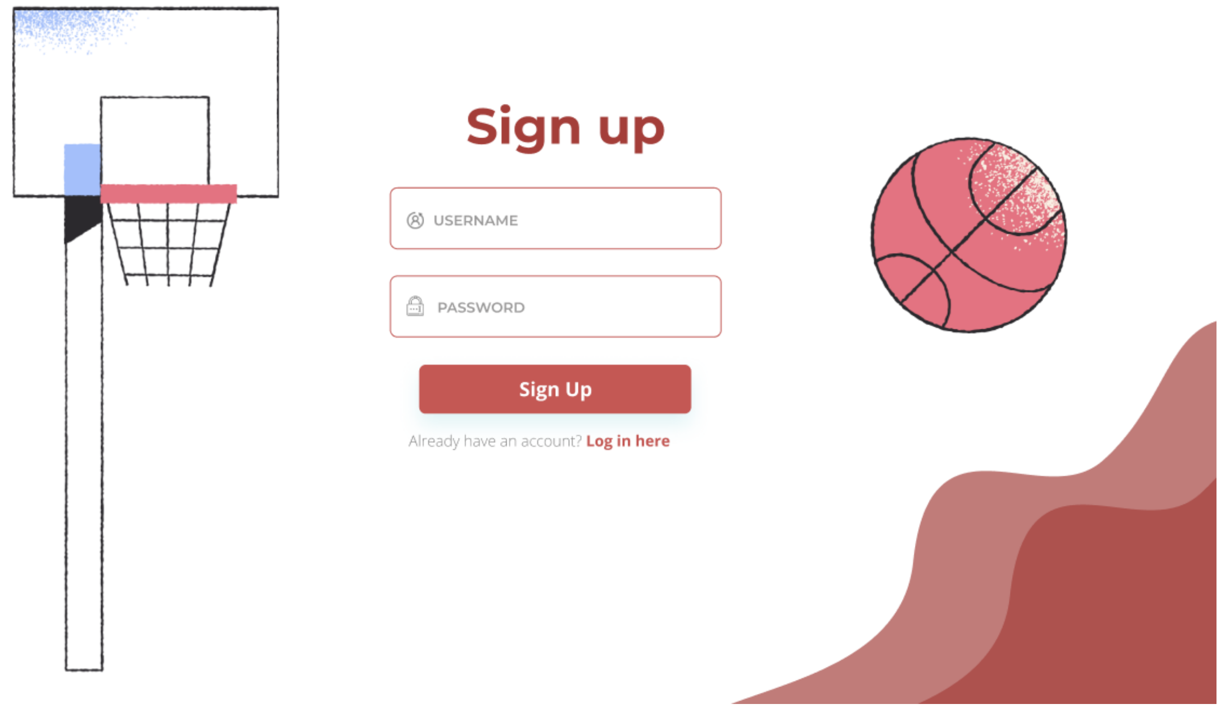
Startup page:



ARCADE HOOPS

Play a fun game of arcade basketball!

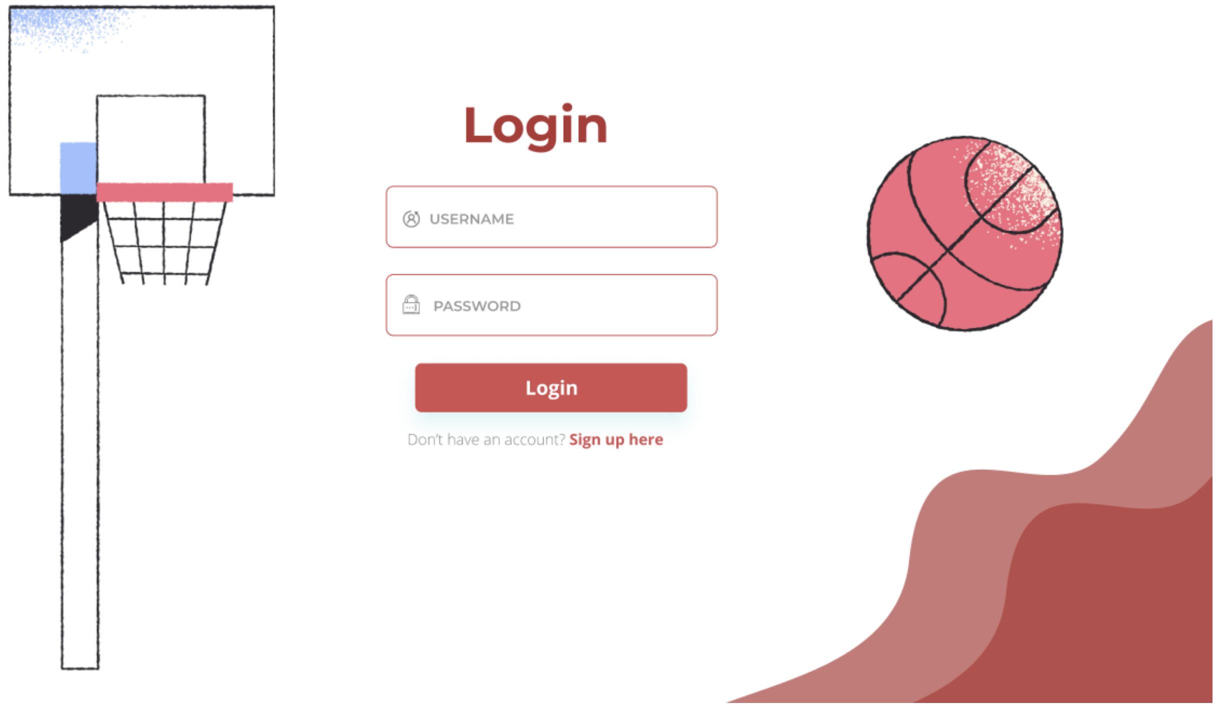
Signup page:

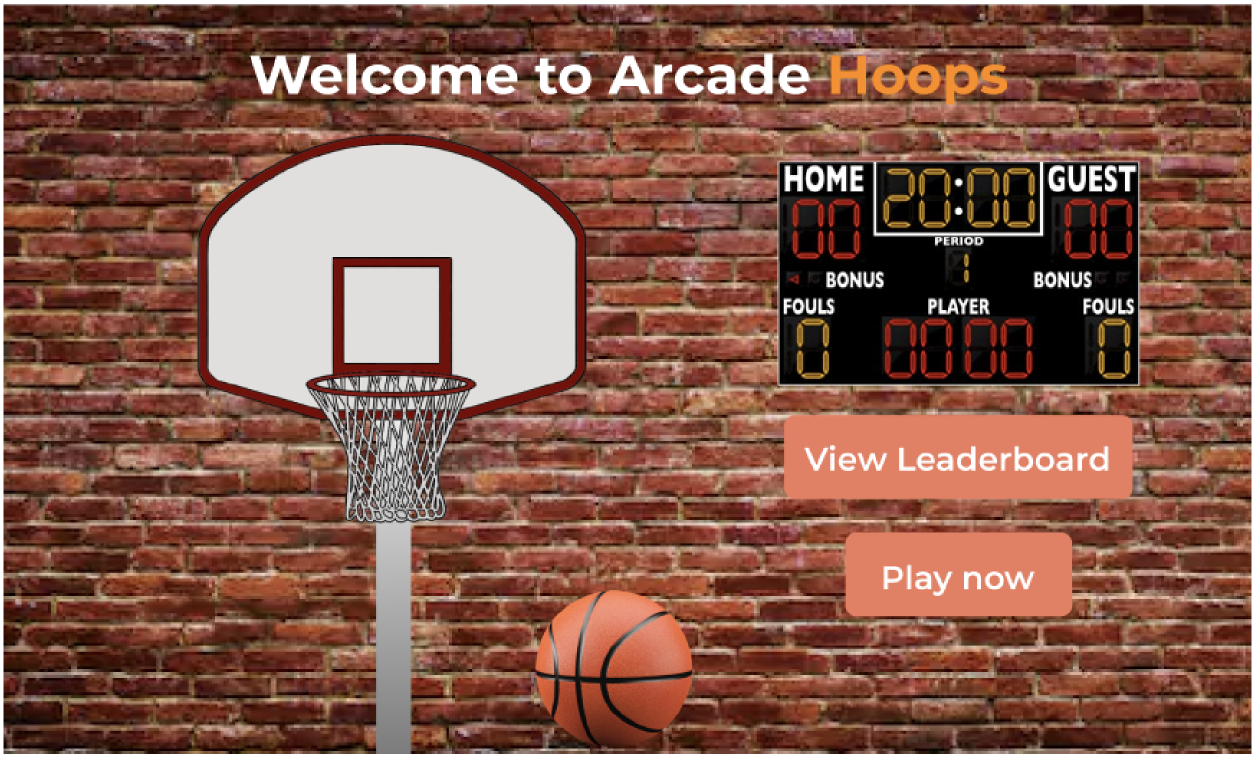


Sign up

Already have an account? **Log in here**

- Login Page:





<u>Leaderboard</u>

- Displays top 10 highest scores
- Constantly querying from backend for updates to leaderboard, display once it updates

Game

- Frontend: We will use a RealityKit for the animations and it will count the # of hoops scored. At the end of the game, we will send the score to the backend to update the leaderboard.

Backend

Live Updating Leaderboards (ask Swapnil, network/threads)

Everytime someone starts a game from their phone, they become a client (in python code) and connect to a server running on a remote computer. That connection will then cause Java code to create a new ServerThread which will get scores from every client, update the database, and tell the client to update their leaderboard whenever any other client changes it.

Tomcat Servlet

Gets data and checks if request is: create, login, score, getScores

Connection to mySQL

- Connect to database
- Import java.sql.connection
- Replace server name, username, and password with your credentials

Database Changes

void createUser(); // creates a user, will throw an exception if user exists
void attemptLogin(); // tries to log user in, throws an exception if password or username is not correct
addScore(); // Adds a user's score into the database
ArrayList<int> getHighScores(); // returns top 10 high scores from database

Database

SELECT 10 FROM Scores.score order by score

Scores
ID (primary key)
Score Int
User id INT

Only add a new user if its username is unique
Check their password matches when they attempt to log in

USERS
ID (primary)
USERNAME string
PASSWORD (hashed)

- password.hashCode()
- getAllHighestScores()
- addUser(string username, string password, Integer highestScore)
- updateHighestScore(User) (for individual user)
- updateHighestScores() (for all users, sends to frontend)
- deleteUser(User)
- Connecting JAVA to SQL