Assignment 2 - Team 9 Project Report

Summary of Software (1 para for problem & partner, 1 para to introduce any existing software)

The product to create is a mobile application for understanding issues around global clean water supply. Our partner is Engineers Without Borders (Water for the World - WT4W). This organization is concerned with educating the world about the global clean water supply and the disparities of access to it for different parts of the world. The app is intended to educate people on the disparities of access to clean water through a series of questions along with an interactive module. The interactive module is a simple exercise where the user selects a country of the world, and depending on which country, is given a limited amount of in-app currency to create a virtual water filter whose parts have various costs. This simulation helps us understand both how a simple water filter is made, as well as the lack of resources some countries of the world face when required to build such systems. The app helps new users understand the complicated process of setting up water filters by trying on the app first. A common use case for the app would be for students to use the app to learn about the unequal distribution of resources in the world. The app would also support educators as a teaching method while also assessing and providing information on student responses to the educator.

From the previous CSC301 group, we have been given the Android APK version of the app. Our group is tasked to create the iOS version of the app, but with augmented features. Should time permits, we would try to synergize the functionalities of the app in both Android and iOS versions and streamline the process of utilizing the app.

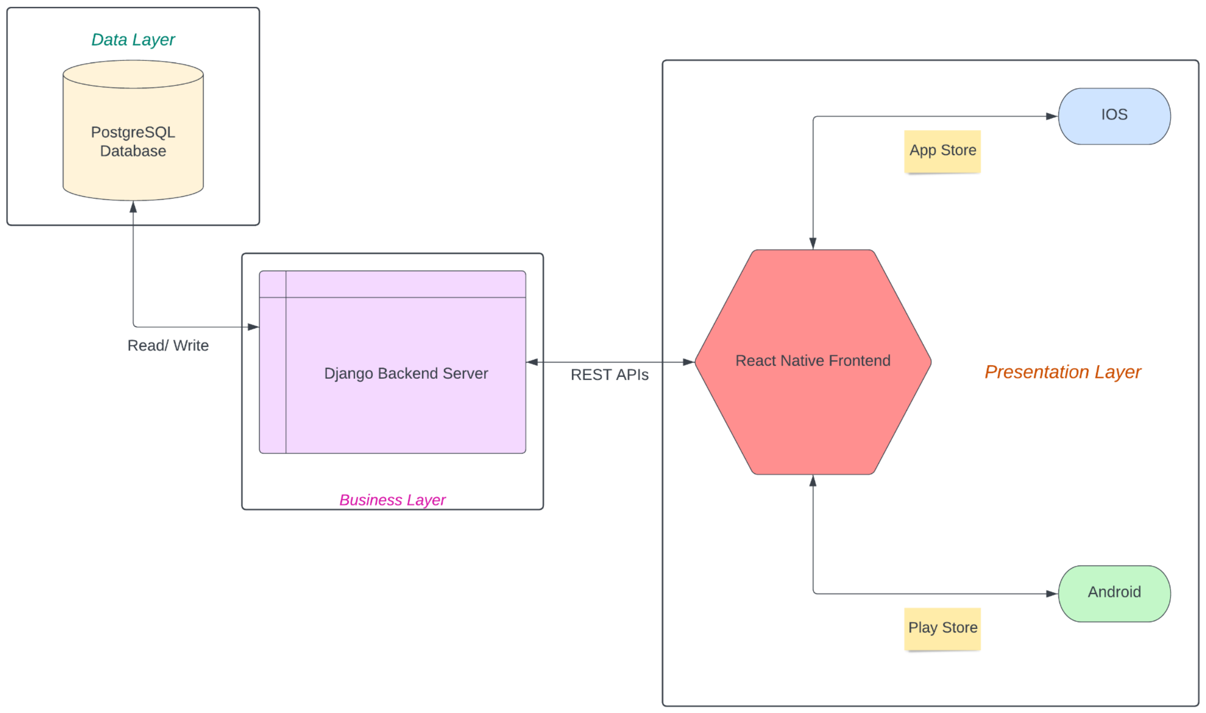
Project Distribution (1-4 paras on how we divide the project, include a software architecture diagram and how each component connects)

Having 7 members in our team, we decided to break down to 3 subteams: frontend, backend, and database. Breaking it down into the three categories fits best with our team's goal of achieving the partner's goal while ensuring that all features are functioning well and are aesthetically fitting to our expectation. We distribute ourselves to these subteams according to area of interest or expertise while focusing on the user story that our team has agreed upon.

User Story: As a player, I want to be able to log into my own account in order to share my progress with my friends/teachers and also observe other people's progress (not sure if we want to rephrase this to make it less about logging in and more about having a teacher/student account. Below is an example of a user story for the latter)

- As a player, I want to be able to have access to a student account with student functionalities (joining a homeroom) if I am a student or a teacher account with teacher functionalities (having a homeroom) if I am a teacher.

Software Architecture Diagram:



<u>Sub-Team's Contribution (one para each subteam)</u>

Frontend (Sargunan & Stefanus):

We are in charge of implementing the front-end part of the mobile application. As the team has decided, we implement the front-end with React Native. We are using React Native to be able to facilitate both android and iOS platforms, and since we are new to the language we went through the basics and attempt to utilize the built-in methods from react and react native. For the first sprint, we decided to focus on keeping things functioning, simple, and clear to feed the whole purpose of our overall project. Starting off with our first sprint, we split into different segments of the user story - one involving the sign up and login, one for the student's page so that they can add new homerooms, and another for the teacher's page to create their homeroom. Both the student and the teacher will be able to delete the homerooms on their respective lists, and both of the users can also directly try the simulation for fun if they want to. The purpose of this is because our partner wanted the app to be more like an enjoyable game and not fully educational base, and hence having the 'skip to simulation' button for all users fulfills the goal. By the end of the sprint, our subteam was able to implement the following functionalities for the front-end: (1) log-in page for existing users, (2) sign-up page for new users - student, teacher, or normal user, (3a) ability for students to join a new homeroom, (3b) ability for students to view their current list of homerooms, (4a) ability for teachers to create a new homeroom, (4b) ability for teachers to view their existing list of homerooms, and (5) the button for all users to skip through the process and try the simulation (though this part will be functional when we have the front end of the simulation). Overall, we were able to implement the front-end of the early part of the project that we will build on throughout the process.

Backend (Nigel, Young Jun, Varun):

We are in charge of implementing the back-end logic of the mobile application. We are using the Django REST framework as it works efficiently with the database that we chose. Our team first decided on the overall schema of the account creation process. We implemented different types of users to fulfill the request of our partner, which includes Student and Teacher. To prevent redundant code, we created one model where the required fields of the user creation process solely depend on the specific type of user. For example, a student will be required to enter their school, grade, and a homeroom_id that their teacher provided, while a regular user of this app will not be required to enter that information at all. In addition, we implemented url patterns, views, serializers to complement a front-end that will be added later. To test these implementations, we used Postman to mock database inputs and ensured that our code was correct and functional. Apart from that, we also implemented logic that allowed students and teachers to be grouped by a homeroom_id that was mentioned before. The purpose of this is for educators to have the ability to see the responses and scores of students when they go through this app. This was a key feature of the user story we chose as our partner wanted this app to emulate an in-person workshop as much as possible. By the end of the sprint, our subteam was able to implement the following functionalities for the backend-end: (1) Register as default user, student, or teacher, (2) Login using token authentication, (3) Teacher creates or ends a homeroom, (4) Student joins or leaves a homeroom. We have also deployed backend to a cloud service (Railway). Overall, we were able to target different types of users using non-redundant code, and implement back-end logic for creating homerooms that allowed grouping of teachers and students and teachers to monitor student progress.

Database team (Justin & Glen):

We are in charge of implementing a database that could store relevant information for our mobile application. To start off our first sprint, we wanted to create a schema of relations for all involved users/players. These include creating a player relation to store information about any user of the game, a student relation to store additional information about a player who is currently a student, a teacher relation to store additional information about a player who is currently a teacher, and a homeroom relation to mimic somewhat of an online class with a teacher in charge to conduct lessons through our game application. We made use of foreign key constraints to introduce relationships between our tables and to partition off overlapping information from each other. Following an agile framework, we focussed on problem divisions and frequent release of database functionalities. We broke our database problem down into 3 main parts: implementing functionalities, creating a fake UI to manually test those functionalities, and constructing a unit testing process. By the end of this first sprint involving technical work, we managed to implement the following functionalities for our database: (1) creating all relations, i.e. adding all required relations to database using Python, (2) deleting a relation or all relations, (3) inserting a new player account, who is either a student, a teacher, or neither, (4) deleting a player account, regardless of whether it's an account of a student, teacher, or neither, (5) verifying a player's credentials for logging in (verifying email - password), (6) updating player information, (7) viewing all accounts that exist, and (8) inserting a new homeroom and deleting it. We have also connected our database to Railway, i.e. deployed our database to a cloud service using railway.app, so we can see modifications to our database instantly on the server.