System Design Document

High level overview of OpenRun

Presented by ScarDev

Featuring:

Damian Bhatia
Salik Chodhary
Rahul Doguparty
Jameson Joseph
Vigaash Sivasothy

Table of Contents:

CRC Cards:

Frontend

Module Name: Home	
Responsibilities ● Home page for the user to be on when they first access the website ● Allows the user to log in or make a new account and displays the team slogan	**Collaborators** ● SignIn ● SignUp

Module Name: Landing	
Responsibilities ● Main page that the user goes to after signing into the website ● Lets the user access other tabs like profile, maps etc. from the landing page	**Collaborators**

Module Name: SignIn	
Responsibilities ● Sign in component for the user to log in to the website with their account ● Interacts with Home and Landing page components for the user to access the site to log in then after see their account details	**Collaborators** ● Home ● Landing

Module Name: SignUp	
Responsibilities ● Sign up component that allows the user to create an account for the user to use the website with	**Collaborators** ● Home

Module Name: NavigationBar	
Responsibilities ● Allows the user to navigate to other pages on the website ● Available on the landing page after the user has signed in	**Collaborators** ● Landing

Module Name: Profile	
Responsibilities ● Where the user has their public information for other users to view ● Is able to be customized by the user	**Collaborators** ● NavigationBar

Backend

Module Name: Server	
Responsibilities ● Initializes the server instance for the app to run on and for users to connect to ● Hosts the app service for the user	**Collaborators** ● Routes ● UserController

Module Name: Routes	
Responsibilities ● Parses request URL and calls the corresponding controller method ● Routes the verification token to the AuthorizationMiddleware	**Collaborators** ● AuthorizationMiddleware

Module Name: AuthorizationMiddleware	
Responsibilities ● Verify the request for valid token ● Gets request routed from Routes ● Grants access if given valid token	**Collaborators** ● None

Module Name: UserController	
Responsibilities ● Where most of the work of the app occurs ● UserController is like the main processing unit of the app, handles user requests and redirects them to other pages ● Connects to the UserModel to call and use necessary functions	**Collaborators** ● Helpers ● UserModel

Module Name: Helpers	
Responsibilities	**Collaborators**

● Provides useful utility functions that are used by the app backend ● Various functions for sending requests, handling requests, etc	● None

Module Name: DatabaseClient	
Responsibilities ● Connects to MongoDB and provides various functionality for the instance ● Allows for modification to the database through the app	**Collaborators** ● None

Module Name: UserModel	
Responsibilities ● Provides basic CRUD functions for the MongoDB documents ● These basic functions will allow for mutability of database that will be called by various methods in the app backend	**Collaborators** ● DatabaseClient

System Interaction

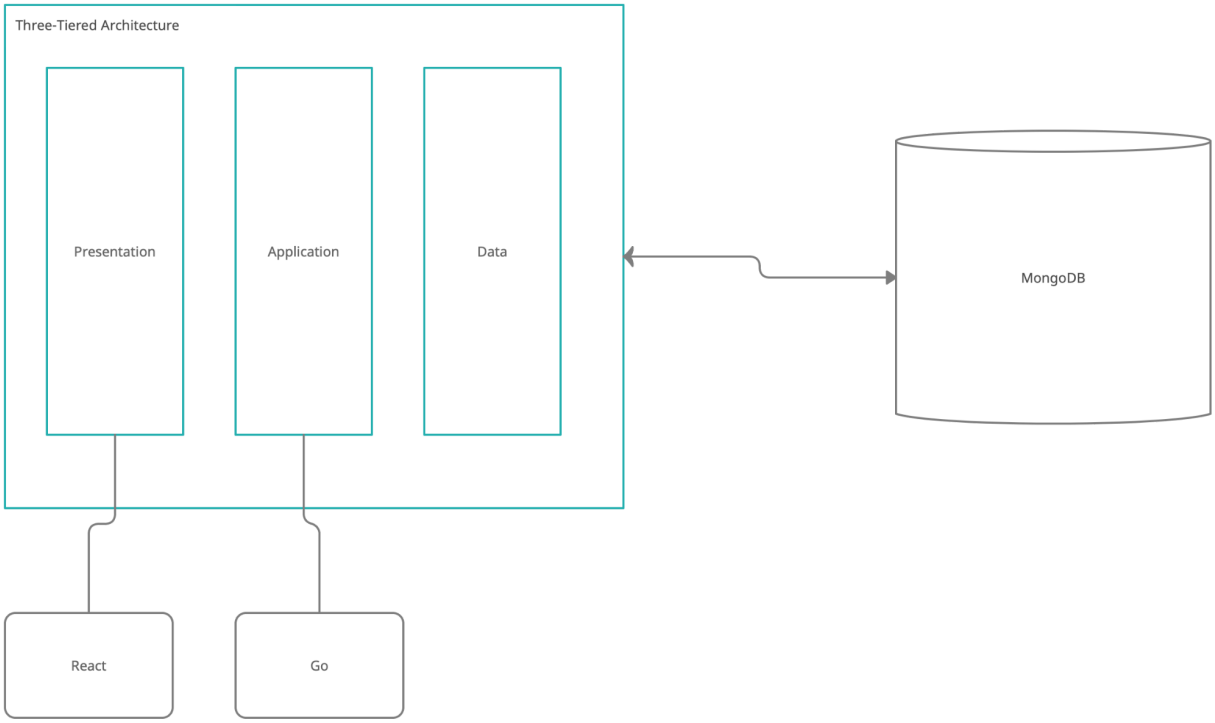
Within our system environment, there are a multitude of dependencies and prerequisite software. Firstly, the user must have the Node.js installed to run the react web app. Furthermore, the user must also have npm installed on their machine as well. In addition, users must have yarn installed. These modules are the primary software tools to run the web app. Users must also have golang to run the backend of the app. The required operating system to run the app is either Windows 10, MacOS or Linux. The user will also need a browser and a suitable network connection to view and connect to the service.

System Architecture Diagram

Reference to Three Tier Architecture: https://www.ibm.com/cloud/learn/three-tier-architecture

The system architecture we will be following is a Three-Tier Architecture. In this architecture there are three separate tiers. The presentation tier is focused on displaying information to the user and collecting information from the user. The application tier is focused on processing the information collected from the presentation tier, processing user inputs and modifying the data tier as needed. The data tier is the database where information is stored.

Relating back to our project, the technologies we will be using are React, Go, and MongoDB. React serves as the presentation tier/web server, Go serves as the application tier/application server, and MongoDB as our data tier/database server.



System Decomposition:

React will allow us to display our information to the user and also collect that information. Through routers and http requests, users will be able to navigate through different components of our application, such as going to the homepage to the profile page. The different components that the user will be able to interact with include the homepage, the sign up page, the search page, etc. These will serve as the presentation tier. With http requests we'll be able to collect information from the user, such as account creation/login information, search requests, updating profile information, feed posts, and more. These requests will be sent to Go, our application tier, for further processing and handling. It'll serve as the communication between React and MongoDB. For instance, when a user wants to create a new account or login, Go will process all that information and authorization accordingly, modifying the database as needed so that React will be able to display the new information correctly.

In the case of errors such as invalid user input, the software should be able to handle these errors accordingly. Say a user tries to login with an invalid account, they should be met with an appropriate page that'll say so.