Content

1 System Design CRC cards

Class Name:	
Parent Class (if any): **Subclasses (if any):**	
Responsibilities:	**Collaborators:**

Class Name: ExerciseLog	
Responsibilities: ● Display all of a user's previously recorded sets in a scrollable list. ● Present options allowing users to add, update, or delete recorded sets.	**Collaborators:** ● ExerciseRecorder ● ExerciseGroupSelect

Class Name: ExerciseGroupSelect	
Responsibilities: ● Provide users the option to search for exercises by clicking on a muscle group, or querying via a search bar.	**Collaborators:** ● ExerciseLog ● ExerciseSelect

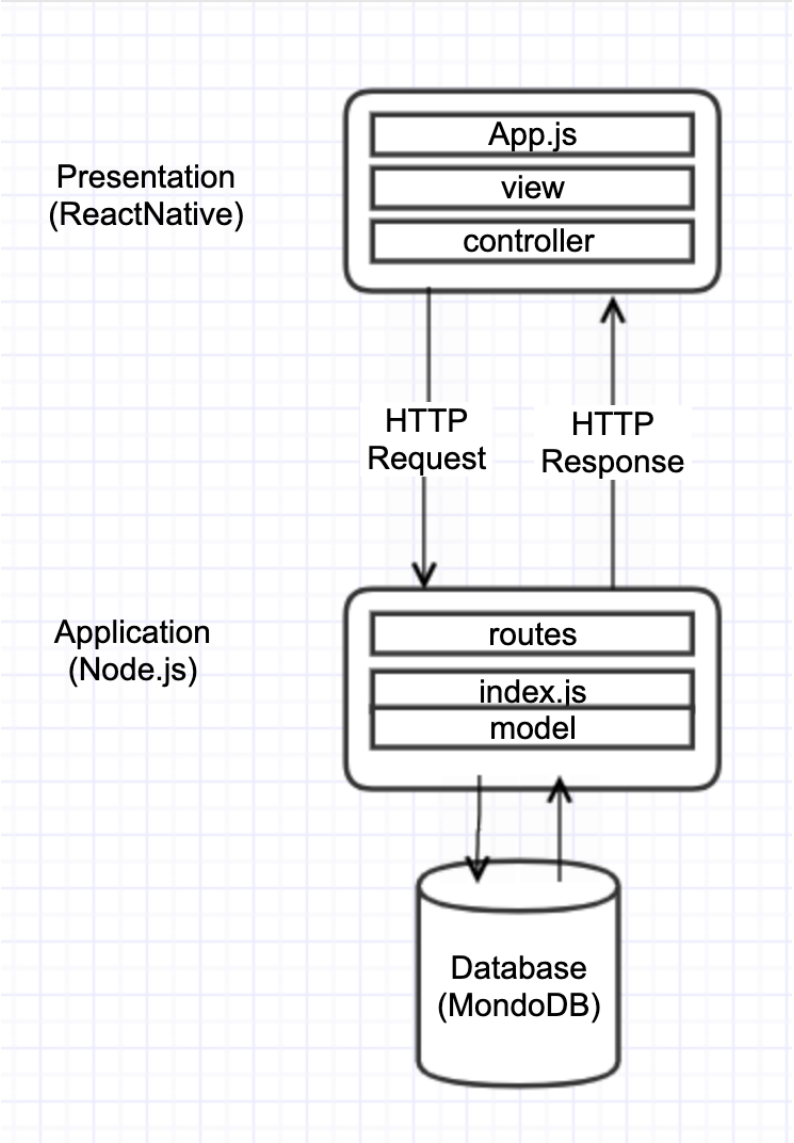
Class Name: ExerciseSelect	
Responsibilities: ● Display list of all found exercises given query from previous page (ExerciseGroupSelect) ● Allow users to select any displayed exercises, moving them to a screen where they can log the exercise.	**Collaborators:** ● ExerciseGroupSelect ● ExerciseRecorder

Class Name: ExerciseRecorder	
Responsibilities: ● Display exercise in terms of Weight & Reps or Time & Distance depending on exercise type ● Allow user to modify exercise metrics ● Allow users to log exercise, saving the exercise metrics in their history	**Collaborators:** ● ExerciseLog ● ExerciseSelect

Class Name: Survey	
Responsibilities: ● Display survey questions and answer options ● Upon click, directs user to next question or the result page ● Store user's survey results and send it to backend	**Collaborators:** ● Plan

Class Name: Plan	
Responsibilities: ● Match user's survey results with a plan ● send the plan to frontend	**Collaborators:** ● Survey

2. System Architecture diagram (Three tiers-MERN Stack)
reference: https://www.mongodb.com/mern-stack



2.2 Explanation

- the front-end display tier (React Native), application tier (Express.js & Node.js), and database tier (MongoDB Atlas).

- React Native: build up interfaces through components, connect them to data on backend server, and render them as HTML.
- Express.js: has models for URL routing (matching an incoming URL with a server function), and handles HTTP requests & responses.
- MongoDB Altas: data storage

- Interactions among the tiers: Making HTTP Requests or GETs or POSTs from React.js, we can connect to Express.js functions, which in turn use MongoDB's Node.js drivers, either via callbacks for using Promises, to access and update data in your MongoDB database.

2.3.System Decomposition/Project Structure

Backend:

- models: directory that contains the data structure to model objects from response e.g. Users.js, Exercise.js
- routes: directory that contains files that define server endpoints e.g. users.js, Exercise.js
- config.js: config file that holds the credentials to connect to the MongoDB Atlas database
- index.js: entry point of backend server
- package.json: dependencies:

```
"dependencies": {
   "body-parser": "^1.20.0",
   "cors": "^2.8.5",
   "express": "^4.18.1",
   "mongoose": "^5.13.14",
   "morgan": "^1.10.0"
}
```

Frontend:

- view: directory that contains all js files of the main screens of the mobile app
- App.js: entry point of the react native frontend (contains tab bar navigation)
- pakage.json: dependencies

```
"dependencies": {
   "@react-navigation/native": "^6.0.10",
   "@react-navigation/native-stack": "^6.6.2",
   "expo": "~45.0.0",
   "expo-splash-screen": "~0.15.1",
   "expo-status-bar": "~1.3.0",
   "react": "17.0.2",
   "react-dom": "17.0.2",
   "react-native": "0.68.2",
   "react-native-web": "0.17.7"
}
```