**Team Cloud Engineers**

System Design Document

**Team:** Mustafa Hafeez

Smit Patel

Muhammad Raza

Abdullah Khan

Anis Saha

**Date:** July 2, 2021

Table of contents

1. CRC Cards
2. System Requirements
3. System Architecture
4. System Decomposition
5. Error Handling

CRC Cards

Since we are using React and NodeJS for our project, we don’t have classes like other programming languages, so instead our CRC cards represent the Database objects we are using to store information and data about the system.

|  |  |
| --- | --- |
| **Class Name:** Users | |
| **Parent Class:** -  **Sub Classes:** - | |
| Responsibilities   * Store user information such as id, email password, avatar link, bio * Store user role which will determine the privilege for their actions in the app * Store the stages this user has access to for e-learning | Collaborators   * Companies * Posts * Comments * Stages |

|  |  |
| --- | --- |
| **Class Name:** Companies | |
| **Parent Class:** -  **Sub Classes:** - | |
| Responsibilities   * Store company information such as id, mission statement, logo link * Store users that are members, creators of the company | Collaborators   * Users |

|  |  |
| --- | --- |
| **Class Name:** Comments | |
| **Parent Class:** -  **Sub Classes:** - | |
| Responsibilities   * Store relation information such as the post it belongs to, the user that commented it * Store the content of the comment, the time it was created | Collaborators   * Users * Posts |

|  |  |
| --- | --- |
| **Class Name:** Posts | |
| **Parent Class:** -  **Sub Classes:** - | |
| Responsibilities   * Store posts information such as title content, author, number of likes * Store links to any documents or media attached to the post | Collaborators   * Users * Comments |

|  |  |
| --- | --- |
| **Class Name:** Courses | |
| **Parent Class:** -  **Sub Classes:** Modules | |
| Responsibilities   * Represent the courses that a company will take part in, in the e-learning section * Store stage information such as title description, image | Collaborators   * Users * Modules |

|  |  |
| --- | --- |
| **Class Name:** Modules | |
| **Parent Class:** Courses  **Sub Classes:** | |
| Responsibilities   * Group relevant/similar items in an e-learning stage together * Store module information such as the title | Collaborators   * Assignments * Lessons * Courses * Users |

|  |  |
| --- | --- |
| **Class Name:** Lessons | |
| **Parent Class:**  **Sub Classes:** | |
| Responsibilities   * Store related files for this lesson * Store info about a lesson for students | Collaborators   * Lessons * Courses * Users * Files |

|  |  |
| --- | --- |
| **Class Name:** Assignments | |
| **Parent Class:**  **Sub Classes:** | |
| Responsibilities   * Store related files for this assignment * Store info about an assignment for student like duedate and expiry | Collaborators   * Assignments * Courses * Users * Files |
| **Class Name:** Files | |
| **Parent Class:**  **Sub Classes:** | |
| Responsibilities   * Store urls of related files for this assignment * Store info about uploaded files | Collaborators   * Assignments * Lessons |

|  |  |
| --- | --- |
| **Class Name:** Submissions | |
| **Parent Class:**  **Sub Classes:** | |
| Responsibilities   * Store submission related info for an assignment * Store info the student who submitted * Store the grade and comments for this submission from the instructor | Collaborators   * Assignments * Users * Files |

System Requirements

The project does not have any direct OS dependencies. Since it is a react and NodeJS project, it just needs NodeJS and npm installed. The project was developed on node version 14.16.0 and npm version 7.6.1 but it may be compatible with other versions of node and npm. The app also contains a NodeJS microservice for user auth, which needs to be run on localhost port 8080 to allow the frontend to register and log in users. Another requirement for the system to work properly is a working internet connection so that the frontend and the auth microservice can communicate with the firebase db and file storage.

System Architecture

Diagram

Description automatically generatedThe architecture of the project is fairly simple, and is based on the JAMStack. We have a frontend react app that communicates with the custom built auth service for user registration, login and logout. Once the user is logged in, the app directly communicates with firebase for all its data and operations.

System Decomposition

The front end react app will contain all the business logic and be the centerpiece where all other components will plug into. The auth microservice is a separate piece that was created in order to not use proprietary fireabse auth. The reamaing pieces piece of the architecture is only firabase which is a cloud hosted service we can use for a database and file storage. The react app communicates with firebase directly using JS SDKs included in the react project.

Error Handling

For error handling, any erroneous user input data, for example during login will alert the user of incorrect info being provided. In the case of network failure, since the entirety of the app relies on a network connection to communicate with firebase, the app cannot proceed and will alert the user that the app is not functional with a working connection.