10/02/2022

Developer Student Clubs

University of Toronto Mississauga

GDSC2.0

System Design Document

Stephan Motha, Vishay Singh, Sahib Nanda, Dale Rodrigues, Litao Chen, Ahmed Al-Mandalawi

Table of Contents

CRC Cards

Class Name: Models
Parent Class (if any): None **Subclasses (if any):** answerModel, teams, testModel, users
Responsibilities: ● MongoDB backend model definitions
Collaborators: ● Body-parser 1.19.1 ● Express 4.17.2 ● Mongoose 6.1.7 ● Cors 2.8.5 ● Nodemon 2.0.15

Class Name: Components
Parent Class (if any): None **Subclasses (if any):** javascript file with an accompanying css file
Responsibilities: ● Create a functional user interface
Collaborators: ● "axios": "^0.25.0" ● "bootstrap": "^5.1.3" ● "npm": "^8.4.1" ● "react": "^17.0.2" ● "react-bootstrap": "^2.1.2" ● "web-vitals": "^2.1.3"

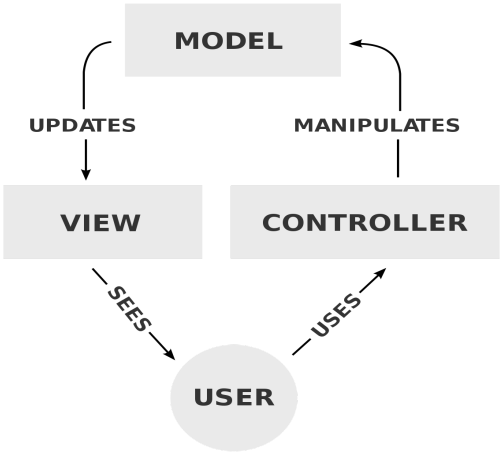
Class Name: Config
Parent Class (if any): None **Subclasses (if any):** db
Responsibilities: ● Connect to MongoDB
Collaborators: ● Body-parser 1.19.1

- Express 4.17.2
- Mongoose 6.1.7
- Cors 2.8.5
- Nodemon 2.0.15

Class Name: Server
Parent Class (if any): None **Subclasses (if any):** None
Responsibilities: ● Handle endpoints for the frontend to backend connections
Collaborators: ● Body-parser 1.19.1 ● Express 4.17.2 ● Mongoose 6.1.7 ● Cors 2.8.5 ● Nodemon 2.0.15

Class Name: Routes
Parent Class (if any): None **Subclasses (if any):** login, createAnswers, getAnswers
Responsibilities: ● Handle routes for endpoints for models in the database
Collaborators: ● Body-parser 1.19.1 ● Express 4.17.2 ● Mongoose 6.1.7 ● Cors 2.8.5 ● Nodemon 2.0.15

Software Architecture Diagram



The software architecture design our team chooses is the MVC or (model-view-controller) design as it complimented the way our team wanted to design and develop our system. The model handles the logic and data of the software. The view is responsible for the display and UI. The controller is the connection that processes information to and from the model and view.

In our software, we have a frontend folder that corresponds to the view, a backend folder that corresponds to the model, and a file called server.js that represents the controller. The frontend contains React components that interact with each other and the controller to create an interactive user experience. The backend contains models that represent how the data should look in the database. It also contains routes that perform a specific action in the backend based on instructions from the controller. These instructions are usually in the form of “get” or “post” requests. The controller server.js receives instructions from the frontend, these usually include endpoints, and decides which route to call in the backend.