

GDSC2.0

System Design Document

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

Table of Contents

- 1. CRC cards
- 2. Software architecture diagram

CRC Cards

CRC Maker 2022-03-03, 9:49 PM

Models mentorAppModel, studentAppModel, teams, testModel, user	
Define Schemas for MongoDB	Databases
Databases	None
Store Data Query Data Create Schema	Models Controller
Config (environment)	None None
Define environment variables (server port and database connection)	• None
Server	None Routes, Controller
 Provide communication instance for endpoints and frontend. Define global variables or states which need to be used in the entire application. 	• Axios
Routes	Server applications, users
Provide endpoint paths	Controller

https://echeung.me/crcmaker/

CRC Maker 2022-03-03, 9:49 PM

Controller	Server Applications
Provide functionality to the routes.	• Routes
Manipulate Data.	Database
Validate Requests.	550 887781

2.20	None
Axios	None
Sending requests to Server	Components
Receiving response from Server	Server

None Components (App) Login & Register, Application Form, Resource, Admin Interface, General Information	
Define URLs for pages. Render pages.	• Axios
Validate Payloads.	

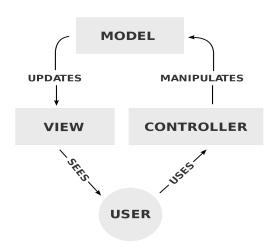
https://echeung.me/crcmaker/

Software Architecture Diagram

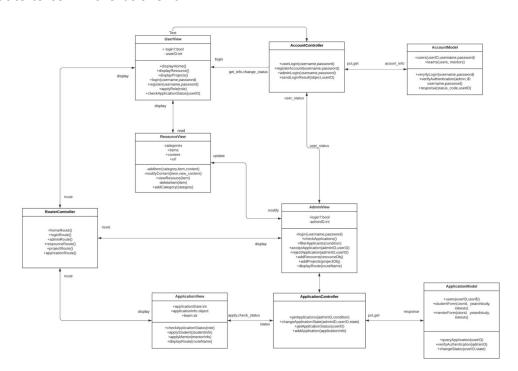
The software architecture design our team chooses is the MVC or (model-view-controller) design as it complimented the way our team wanted to

design and develop our system. The model handles the logic and data of the software. The view is responsible for the display and user interface. 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.



For a better view, please view the original pdf file in our doc.