

## 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 2022-03-31, 7:06 PM

Abstract  Models  Events, Application, Log, Admin	None , Projects, Resources, Team Models
Define Schemas for mongoDB	Databases
<b>Databases</b> Events, Application, Log, Admin, I	None Projects, Resources, Team Schemas
Store Data     Query Data     Create Schema	Models     Controller
Config (environment)	None None
Define environment variables (proxy port and database connection	• None
Server	None Routes, Controllerss
Provide communication instance for endpoints and frontend     Store application global variables and states	• Axios
Routes	Server Endpoints
Define paths for endpoints	Controller     Endpoints

https://echeung.me/crcmaker/

CRC Maker 2022-03-31, 7:06 PM

		Controller	Server Application, Team and Resources Endpoints
•	Define the functionality of endpoints		Databases
•	Handle requests and response		Routes
•	Query databases		

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

<b>Compo</b> Event, Login, Register, Applicatio	None onents (App) n, Resource, Projects, Admin Interface, General Info Views
Define URLs for pages	Axios
Render pages	-0-0010-000-M-07
Validate Payloads	

https://echeung.me/crcmaker/

Modifications from Sprint2 Doc:

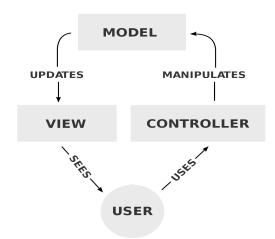
• Add event to backend models and frontend components

Software Architecture Diagram

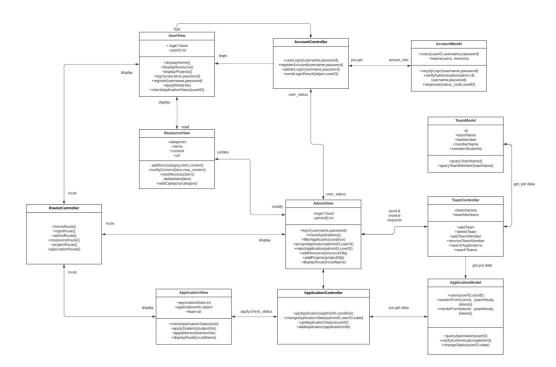
The software architecture design our team chooses is the MVC or (model-view-controller) design as it complimented the way our team wanted to

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.