Programming Club - Technical Board Website
Indian Institute of Information Technology, Guwahati

Phase 2 (Backend and API Development)

Welcome to **Phase 2** of the project. This phase will consist of developing APIs for different tasks like **routing, updating, creating, deleting** etc. along with database persistence. Go through this guide to have a better knowledge about this phase.

Phase Introduction

- API

Architecture

- MVC Architecture

API Routes and Endpoints

- Routes
- Database Endpoints

Phase Introduction

API (Application Programming Interface)

API or **Application Programming Interface** is an interface between different programmes, be it interface between **Frontend** and **Database** (like MongoDB or MySQL) or between 2 C programs. Here we will be developing APIs as an interface between our Database (MongoDB) and Frontend (here HTML, CSS and Javascript).

Through APIs, frontend developers do not need to know the logic behind the code. They can simply use the API provided to make a **CRUD (**Create, Read, Update and Delete) event on the data present in the database.

In this phase, we will be developing APIs for enabling the **CRUD** feature on the database. Using these APIs we can easily integrate **Frontend** with the **Database** in **Phase 3.**

Architecture

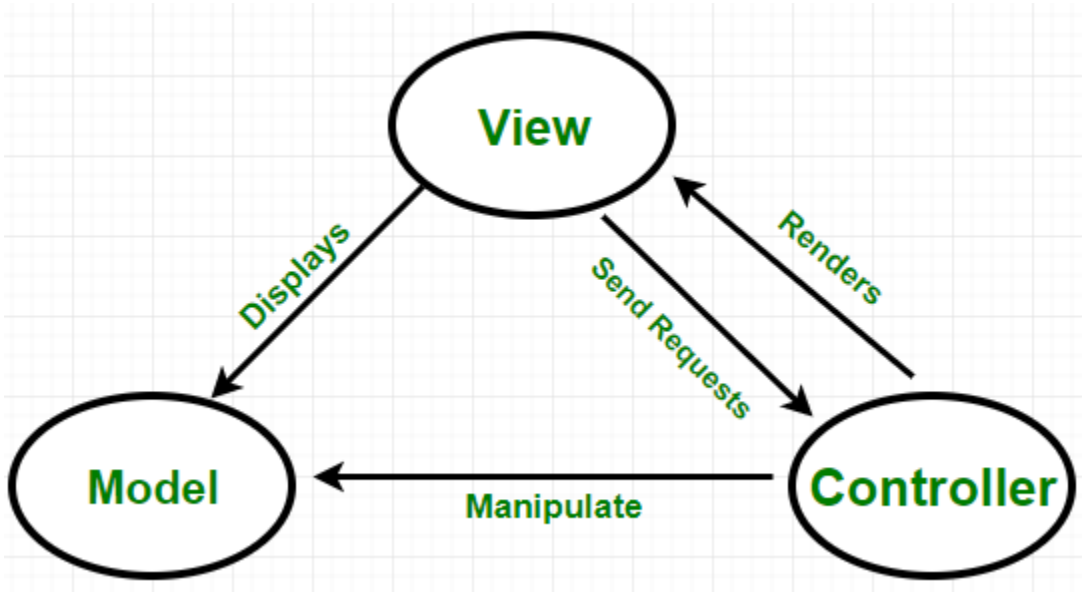
MVC Architecture

The current project is based on **MVC (**Model-View-Controller**)** Architecture. MVC architecture divides an application into **3** major parts, each of which can be developed individually. Feature and functions of each of these are described below.

MODEL part deals with the structure of data and constraints imposed on the acceptance of data in the database. It basically regulates the functioning of the database. In this particular project, we will be having **models** directory for having database models.

VIEW part deals with the client part of the application which is basically UI/UX (design). Components of VIEW may use the data from the MODEL layer to present it to the user in a better way. And thus making VIEW part consisted of both **static** and **dynamic** pages.

CONTROLLER. The data and web pages from the MODEL and VIEW layer are rendered to the users in the form of **requests** and **response.** The user requests the server for a particular data or a page (HTML) and the server responds with the requested data. This request and response are handled by the CONTROLLER layer of the application. Controlling involves which data to provide as a response, or whether or not to provide etc.



(Ref: Geeks for Geeks)

API Routes and Endpoints (This is not the API Documentation, just a development guide)

API endpoints are the connecting point between programmes. Here endpoints are the URLs, which a client will use to interact with server and database. We'll develop endpoints using Node.js.

We'll be following **REST** Protocol for developing our API. Use this link to know about the jargons used.

ROUTES

METHOD	ROUTES	FUNCTION
GET		
	/techboard	Landing page
	/techboard/gallery	Technical Board's Gallery
	/techboard/events	Technical Board's Events
	/techboard/initiatives	Technical Board's Initiatives
	/techboard/hof	Technical Board's Hall of Fame
	/techboard/facilities	Technical Board's Facilities
	/club?clubID=x	X Club's Landing Page
	/club/gallery?clubID=x	X Club's Gallery
	/club/resources?clubID=x	X Club's Resources
	/club/events?clubID=x	X Club's Events
	/club/hof?clubID=x	X Club's Hall of Fame
	/club/projects?clubID=x	X Club's Projects

Database Endpoints

(Learn about JSON and how to access data when provided in JSON format from here)

METHOD	ENDPOINTS	FUNCTION
GET		**(RETURNS in JSON)**

	/api/techboard	Landing {
		logo : <url>
		quote : <Quote String>
		about : <About us String>
		team : [
		name : <Name String>
		position : <Position String>
		level : <Number>
		contact : [ ]
		]
		contact: [
		socialURLs : [ ]
		]
		}

	/api/techboard/facilities	TBF {
		logo : <url>
		quote : <Quote String>
		about : [ ]
		facilities : [
		title: <String>
		desc: <String>
		learn: <url>
		]
		
		}

	/api/techboard/initiatives	TBI { logo: <url>, about: <String>, initiatives: [image: <url>, title: <String>,

		about: <String>] }

	/api/club?clubID=x	XClub {
		clubID: <Number>
		name : <String>
		logo : <url>
		quote : <Quote String>
		about : <String>
		coordinators : [
		name: <String>
		position: <String>
		contact: [ ]
		]
		
		}

	/api/club/gallery?clubID=x	XCGal {
		clubID: <Number>
		quote : <Quote String>
		about : <String>
		image : [
		title: <String>
		about: <Stirng>
		coverImage: <url>
		url: [<urls>]
		]
		}

	/api/club/hof?clubID=x	XCHof { clubID: <Number> quote : <Quote String> about : <String> years : [Number] members : [year: <Number> name: <String> other: [ ]]

		}

	/api/club/resources?clubID=x	XCRes {
		clubID: <Number>,
		quote : <Quote String>,
		about : <String>,
		fields : [
		fieldID: <Number>,
		fieldName: <String>
		],
		fieldRes : [
		fieldID: <Number>,
		logo: <url>
		title: <String>,
		docs: <url>
		],
		otherCard: [
		logo: <url>,
		cardTitle: <String>,
		about: <String>,
		links: [
		title: <String>,
		docs: <url>
		]
		]
		}

	/api/club/events?clubID=x	XCEvents {
		clubID: <Number>,
		quote : <Quote String>,
		about : <String>,
		events: [
		image: <url>
		title: <String>
		description: <String>
		date: <Date>
		venue: <String>
		]
		}

	/api/club/projects?clubID=x	XCProj {
		clubID: <Number>,
		quote : <Quote String>,
		about : <String>,
		projects: [
		image: <url>
		title: <String>
		description: <String>
		other: [ ]
		]
		}

POST		**FUNCTION**
	Same endpoints as the GET request but with POST method.	Updates the database accordingly.
DELETE		**FUNCTION**
	/api/techboard/team?index=x	Deletes team member with index x.
	/api/techboard/facilities?index=x	Deletes facility with index x.
	/api/techboard/initiatives?index=x	Deletes team intiatives with index x.
	/api/club/coordinators?clubID=y&index=x	Delete coordinators with of Club Y
	/api/club/gallery/image?clubID=y&index=x	
	/api/club/hof/member?clubID=y&index=x	
	/api/club/resources/fieldRes?clubID=y&index=x	**REST are SELF EXPLANATORY**
	/api/club/reources/otherCard?clubID=y&index=x	
	/api/club/events?clubID=y&index=x	
	/api/club/projects?clubID=y&index=x	