Software Architecture Pattern of Online Examination System

Course Title: Software Development Project

Course No: CSE-3106

Submitted to

Dr. Amit Kumar Mondal

Associate Professor

CSE Discipline

Khulna University

Submitted by

MD. Ashiqur Rahman

Student ID: 170238

MD.Tarif Hasan

Student ID: 180238

Project Title: Online Examination System

Software Architecture: Model-View-Controller (MVC) Pattern

Model-View-Controller (MVC): MVC (Model-View-Controller) is a software architecture pattern used to structure applications. It divides an application into three components: Model, View, and Controller. The Model manages data and business logic, the View presents the user interface, and the Controller handles user input and application flow. MVC promotes separation of concerns, making applications easier to understand, develop, and maintain. It enhances modularity, scalability, and code reusability by organizing code into distinct components. MVC facilitates collaboration among development teams and supports flexible design and implementation choices. It is widely used in web, desktop, and mobile applications across various programming languages and frameworks.

MVC Architecture Details according to our System:

Model:

Description: The Model component manages the data and business logic of the application. It controls data manipulation, validation, and structure, and responds to requests from the Controller for data manipulation or retrieval.

Accountabilities:

- Manages data persistence, including user authentication details, exam questions, user profiles, exam results, and system settings.

- Implements business logic for user authentication, question management, exam conductance, result generation, and email sending functionalities.
- Responds to requests from the Controller for data manipulation, such as retrieving user profiles, saving exam results, or updating system settings.
- Notifies the View of any changes in the data, such as updating exam results or user profiles, so that the interface can be updated accordingly.

View:

Description: The View represents the presentation layer of the application. It renders graphical elements, displays data retrieved from the Model, records user input, and can be updated to reflect modifications made to the application's state.

Accountabilities:

- Renders graphical elements and user interface components for user authentication, user profile management, exam creation, question uploading, exam conductance, result display, and admin site management.
- Displays data retrieved from the Model, such as user profiles, exam questions, exam results, and system settings.
- Records user input, including login credentials, exam answers, and admin site management actions, and sends it to the Controller for processing.
- Can be updated to reflect changes in the application's state, such as displaying updated exam results or reflecting changes in system settings.

Controller:

Description: The Controller serves as the link between the View and the Model. It receives user input from the View, translates it into actions to be performed by the Model, updates the View to reflect changes in the Model's state, and implements application logic.

Accountabilities:

- Receives user input from the View, such as login credentials, exam answers, or admin site management actions, and translates it into actions to be performed by the Model.
- Carries out interactions with the Model to retrieve or modify data in response to user requests, such as authenticating users, saving exam results, or updating system settings.
- Updates the View to give the user feedback or to reflect changes in the state of the Model, such as displaying success or error messages, updating exam results, or refreshing admin site views.
- Implements application logic to control how data is processed and presented, including user authentication, exam conductance, result generation, email sending, and admin site management functionalities.

<u>Graphical Representation of MVC Pattern:</u>

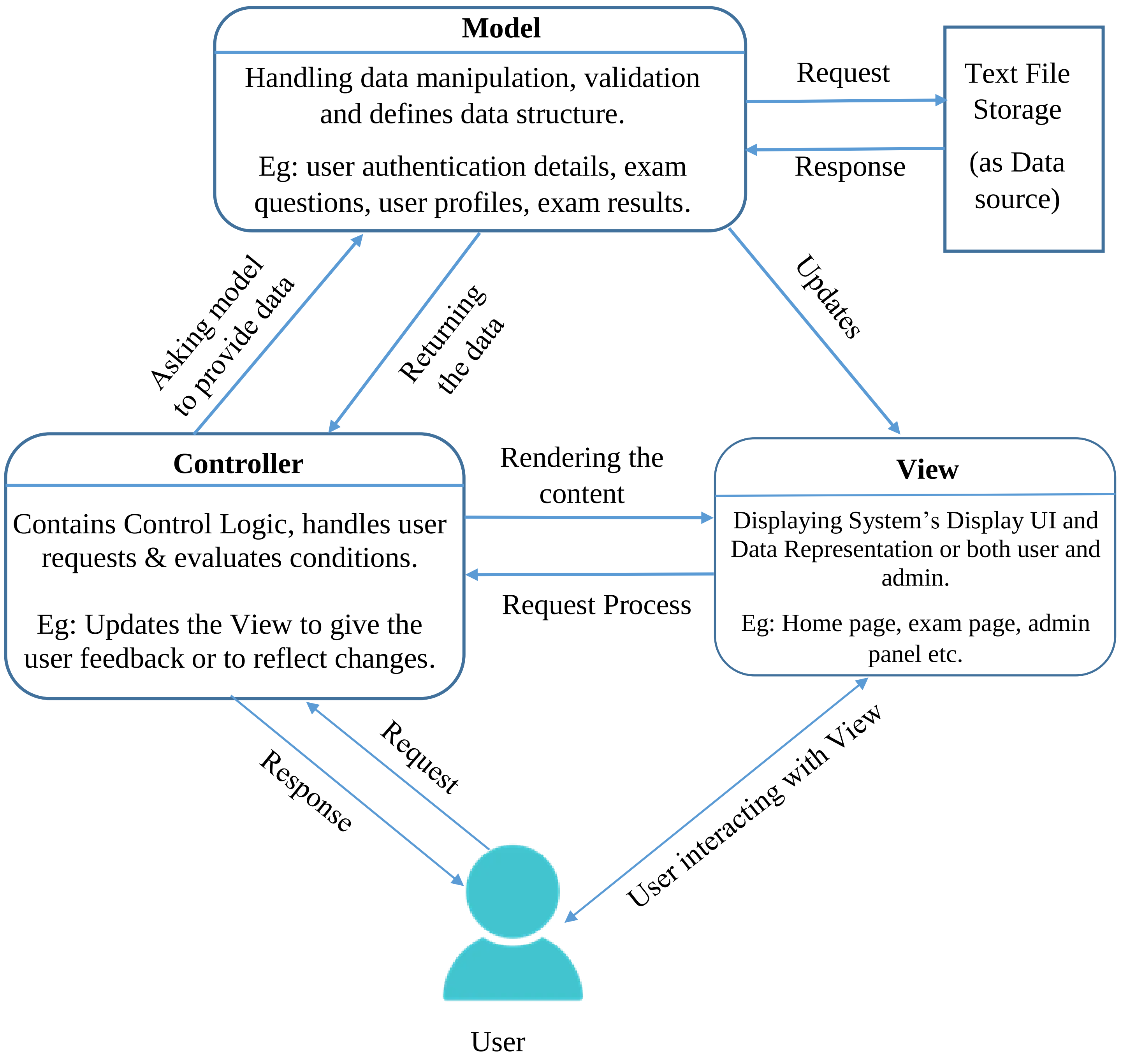
Model
Handling data manipulation, validation and defines data structure.
Eg: user authentication details, exam questions, user profiles, exam results.
Request
Response
Text File Storage
(as Data source)
Asking model to provide data
Returning the data
Updates
Controller
Contains Control Logic, handles user requests & evaluates conditions.
Eg: Updates the View to give the user feedback or to reflect changes.
Rendering the content
Request Process
View
Displaying System's Display UI and Data Representation or both user and admin.
Eg: Home page, exam page, admin panel etc.
Response
Request
User interacting with View
User

Fig: Model-View-Controller (MVC) Architecture of Online Examination System

Conclusion: By applying the MVC pattern to your online examination system project, you can achieve better separation of concerns, modularity, and maintainability, making it easier to develop, test, and maintain the application codebase.