Submitted To,

Amit Kumar Mondal

Associate Professor

Computer Science & Engineering Discipline

Khulna University, Khulna

Submitted By,

Sharmika Das Banhi

Student ID: 210204

Redwan

Student ID: 210207

Computer Science & Engineering Discipline

Khulna University, Khulna

<u>Project Name:</u> Feature-rich Desktop Calendar Software

Model-View-Controller (MVC):

The Model-View-Controller (MVC) framework is an architectural design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. The main goal of this design pattern was to solve the problem of users controlling a large and complex data set by splitting a large application into specific sections that all have their own purpose.

Features of MVC:

1. It provides a clear separation of business logic, Ul logic, and input logic.
2. It offers full control over your HTML and URLs which makes it easy to design web application architecture.
3. It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs.
4. It supports Test Driven Development (TDD).

Components of MVC:

The MVC framework includes the following 3 components:

1. Controller
2. Model
3. View

Model:

Description: The Model represents the data and business logic of the application. In the case of a desktop calendar, it would manage events, appointments, reminders, and any other relevant data.

Components: Data Management: Handles the storage and retrieval of calendar data, including events, appointments, and user preferences.

Business Logic: Implements the core functionality of the calendar, such as adding, editing, and deleting events, as well as handling date and time calculations.

Dependencies: Communicates with the Controller to receive user input and update the data accordingly.

Notifies the View of any changes in the data so that the interface can be updated.

View:

Description: The View is responsible for displaying the user interface and presenting the data to the user. It should be designed to be user-friendly and provide an intuitive way for users to interact with the calendar.

Components:

Calendar Interface: Displays the monthly, weekly, or daily view of the calendar, allowing users to navigate and visualize events.

Event Details: Provides a detailed view of selected events, allowing users to view and modify event information.

User Interface Elements: Buttons, input fields, and other elements that allow users to interact with the calendar.

Dependencies: Receives data from the Model to populate the calendar with events and relevant information.

Controller:

Description: The Controller acts as an intermediary between the Model and the View. It processes user input, updates the Model accordingly, and ensures that the View reflects the current state of the data.

Components: Input Handling: Listens for user input from the View, such as button clicks, date selections, and event modifications.

Command Processing: Converts user input into actions that need to be performed on the Model.

Update Notifications: Notifies the View when changes in the Model require updates to the user interface.

Dependencies: Communicates with the Model to update and retrieve data.

Sends commands and updates to the View to ensure the interface reflects the current state of the application.

Connectors:

Model-View Connection:

1. The Model notifies the View of any changes in the data.

2. The View retrieves data from the Model for display.

3. The View sends user input to the Controller for processing.

View-Controller Connection:

1. The View sends user input to the Controller for processing.

2. The Controller updates the Model based on user input.

3. The Controller notifies the View of changes in the Model, triggering updates in the user interface.

Controller-Model Connection:

1. The Controller communicates with the Model to update and retrieve data.

2. The Controller processes user input and updates the Model accordingly.

3. The Model notifies the Controller of changes, allowing the Controller to update the View.

Advantages of using MVC architecture pattern:

1. Codes are easy to maintain and they can be extended easily.
2. The MVC model component can be tested separately.
3. The components of MVC can be developed simultaneously.
4. It reduces complexity by dividing an application into three units. Model, view, and controller.
5. It supports Test Driven Development (TDD).
6. It works well for Web apps that are supported by large teams of web designers and developers.
7. This architecture helps to test components independently as all classes and objects are independent of each other
8. Search Engine Optimization (SEO) Friendly.

Disadvantages of using MVC Patterns:

1. It is difficult to read, change, test, and reuse this model
2. It is not suitable for building small applications.
3. The inefficiency of data access in view.
4. The framework navigation can be complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC.
5. Increased complexity and Inefficiency of data.

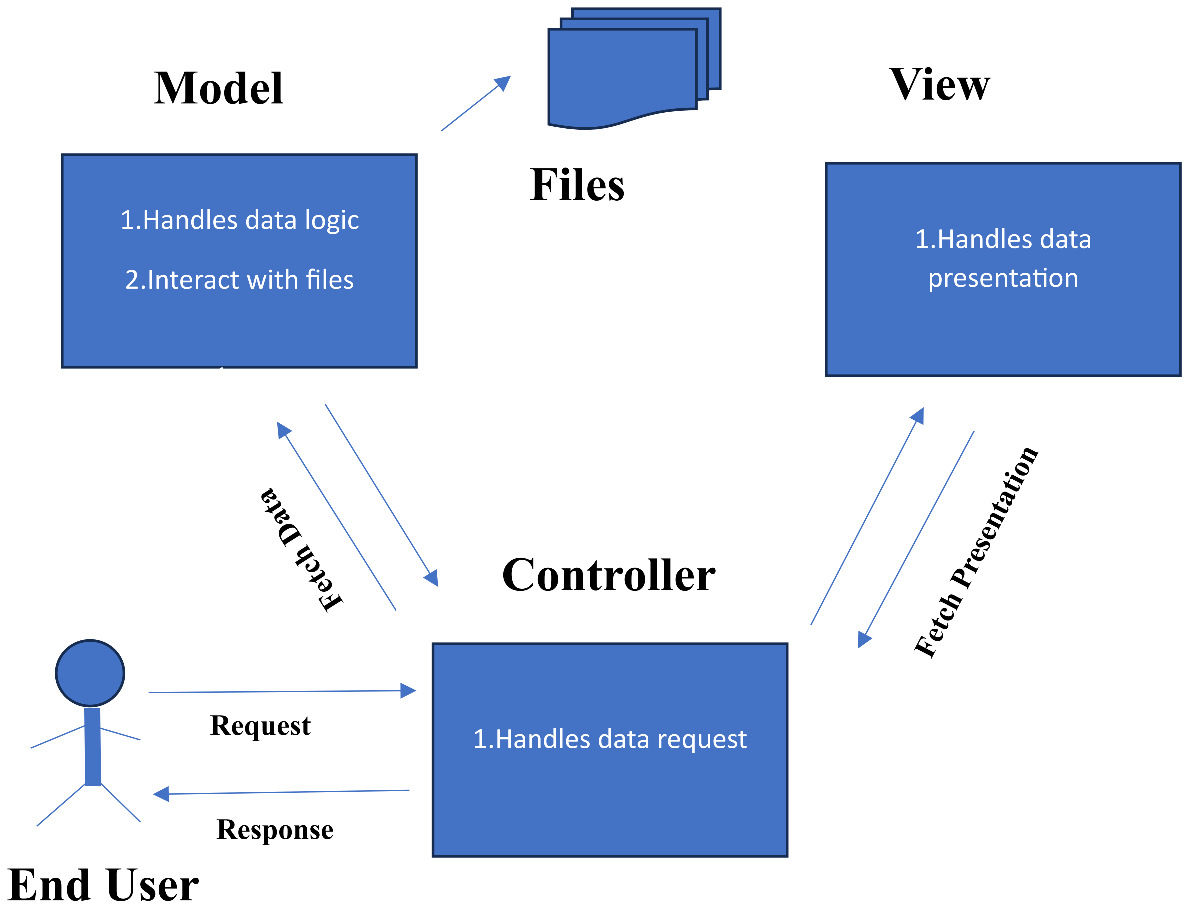

Fig: Model-View-Controller Architecture Pattern Block Diagram