Software Project- 3 Project Report

Project Name: Task Manager

Developer's Information:

Name: Md. Musfiqur Rahman

Student Id: 221-15-4641

Section: 61_Q

Daffodil International University

Project Principle:

The Principle of Optimized Performance; Based on the project's work, I'll follow this principle and the project's work will be carried forward. Reason for choosing this principle:

1. Minimizing Rebuilds
2. Optimizing Rendering
3. Network and Data Optimization
4. Memory Management
5. UI Responsive
6. Device Compatibility
7. Continuous Monitoring and Optimization

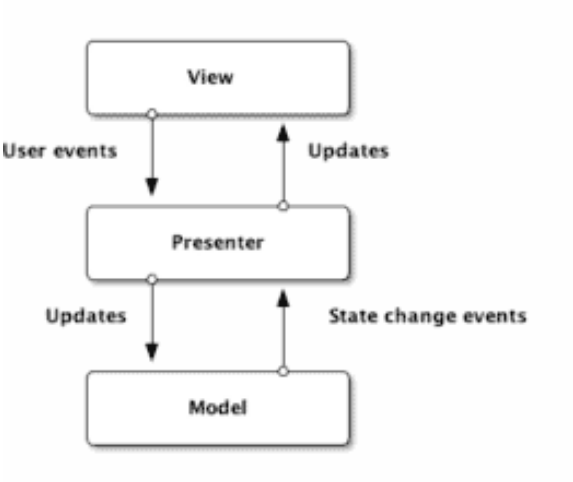
Project Architecture Patterns: Overall, reusable solutions to frequently occurring issues in software design are known as architectural patterns. They have a significant effect on the codebase. We deploy two architectural patterns in our proposal.

- **MVC Pattern:**

  Based on three different concerns, features in the MVC pattern are split into three components. First off, **Rendering UI** components is the responsibility of the view. Second, in response to **UI Actions**, the controller acts.

  Additionally, the model addresses **State Management** and business behaviors. All three parts can communicate with each other directly in the majority of implementations.

  On the other hand, depending on the implementation, the controller has to decide which view to show. The internal structure of the view and the model is not specified by MVC. The view layer is often built within a single class.

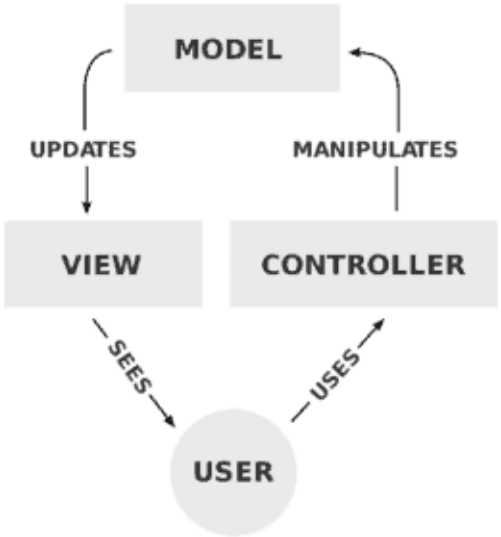


- **MVP Pattern:**
  The MVP pattern is a UI presentation approach based on the MVC pattern's ideas. It just prescribes the view's organization.

  First, rendering UI components is the responsibility of the view. Second, the presenter is loosely coupled to its view through the use of the view interface.

  Ultimately, the model governs business behaviors and state management, while the presenter engages with the view.

  In certain implementations, to get or persist the model, the presenter communicates with a service (controller) layer. Unit tests for the presenter and the model can be more easily written by utilizing the view interface and service layer.



Planning and Requirements:

I'm going to continue with the project based on the **MVP** architecture because all that is required of us in this one is **UI** design. Here, I will design my user interface's presentation layer and assign responsibilities to each UI component. Additionally, the user interface will be demonstrated, and its responsibilities and logic will be developed by the software's activity.

- **Key Features:**
  1. **Completed Task Page:** The tasks done by the user can be kept on the completed task page.
  2. **Create a New Task with Description:** After entering the app, the user can easily create his task with a description.
  3. **Task Count:** Users can see how many tasks are remaining.
  4. **Task Count Completed:** Users can see how many tasks they have completed.
  5. **Delete a Task:** The user can easily delete any task from any page.
  6. **Dark Mode:** Users can use this app in dark mode.

Development Tools, Method & Environment Explanation:

I will use **Figma** software to design the app's **UI** and for development will use **JavaScript** programming language with **React** as a framework. Moreover, the code of the website will be uploaded to **GitHub.**

I will follow the **Agile** development method to complete the entire project which will update the status of our project with **Jira** software and it is a friendly environment to work as a team.

Also, I will use **VSCode** for the software environment and will use some dependency packages and plugins, such as live-preview, open with the live server, auto-indentation, etc.

Future Features and Development of the Project:

I have big plans for this project in the future and after launching this project I will slowly move forward with its new features and development.

- **Upcoming Features:**
  1. **Task Timeline:** The user can add a limited timeline to the newly created task as per his wish.
  2. **Task Timeline Warning:** When the task timeline is over, the time extended will show a warning as a red mark.
  3. **Push Notification:** Through push notification, we will inform the user about the duration and duration of his task every day.
  4. **Assign Date and Calendar:** The user can assign tasks by selecting the date of the calendar in the app.
  5. **Sprint Tasks:** User can put their tasks in a sprint and can start the task with the deadline of the sprint.
  6. **Create Account:** At first, the user has to connect with the app by creating an account.
  7. **Login:** After creating the account, the user can log in with his registered email and password.
  8. **Forget Password:** If the user forgets his password, he can enter a new password through the forget password option.
  9. **Email and PIN Verification:** From login to forget password, user and valid email verification will be done in each case and the user will be verified by sending a 6-digit PIN to the email.
  10. **Profile Update:** User can easily change their profile information along with their profile image.
  11. **Edit Task:** The user can edit his created task at any time.

Overview of Full Developed Project and Conclusion:

I will start the project with the MVP architecture pattern and finally, I will complete the whole project by combining it with the MVC architecture pattern.

Moreover, through the development tools used in this project, I will first launch it for the web. Although it has some limitations due to lack of time, I will fully implement and maintain it later.