Software Project- 3 Project Report

Project Name: Task Manager

Developer's Information:

Name: MD. ABDUL HAMIM

Student Id: 221-15-5264

Section: 61_Q

Daffodil International University

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

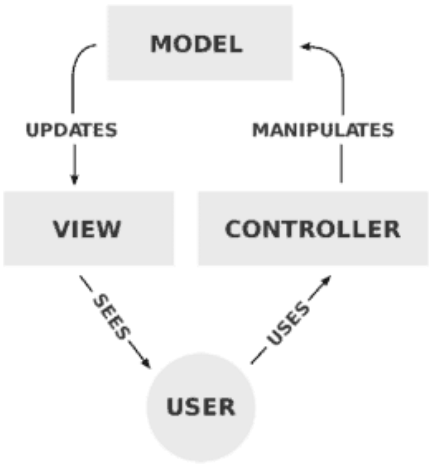
1. Minimizing Rebuilds
2. Optimizing Rendering
3. Network and Data Optimization
4. Memory Management
5. UI Responsive
6. Device Compatibility
7. Continuous Monitoring and Optimization

Project Architecture Pattern's: 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.

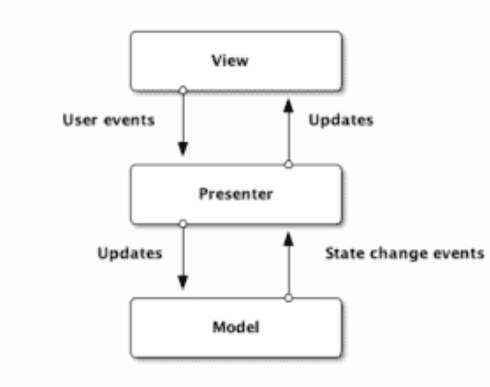


- **MVP Pattern:**
  A UI presentation approach based on the MVC pattern's ideas is called the MVP pattern. It just prescribes the view's organization.

  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, in order 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 Requirement's: 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 in accordance with the software's activity.

- **Requirement's:**
  1. **Create Account:** At first user has to connect with the app by creating an account.
  2. **Log In:** After creating the account, the user can log in with his registered email and password.
  3. **Forget Password:** If the user forgets his password, he can enter a new password through the forget password option.
  4. **Email and PIN Verification:** From log in 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.
  5. **Profile Update:** User can easily change their profile information along with their profile image.
  6. **Create New Task:** After entering the app, the user can easily create his task.
  7. **New Task Page:** After creating a new task, all the user's task's will be stored by default on the new task page.
  8. **Completed Task Page:** The tasks are done by the user can be kept on the completed task page.
  9. **Cancelled Task Page:** By cancelling the tasks that the user is not interested in, they can keep them on the cancelled task page.
  10. **Progress Task Page:** The user can keep the tasks that he starts on the progress task page.
  11. **Update Task Status:** User can update the status of his task very easily.
  12. **Delete Task:** User can easily delete any task from any page.
  13. **Edit Task:** User can edit his created task at any time.
  14. **Count Bar:** User can see how many tasks are new, home many completed, how many cancelled and how many are in progress in the count bar.
  15. **Dark Mode:** User can use this app in both light and dark mode.
  16. **Log Out:** User can log out of the app by clicking on the log out button.

Development Tools, Method & Environment Explanation: I will use **Figma** software to design the app's **UI** and for development and will use **Dart** programming language with **Flutter** as a framework. Moreover, the code of the software will be uploaded in **GitHub.**

I will follow **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 **Android Studio** for the software environment and will use some dependency packages and plugins.

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 Waring:** When 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:** 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.

Overview of Full Developed Project and Conclusion: I will start the project with MVP architecture pattern and finally I will complete the whole project by combining with MVC architecture pattern.

Moreover, through the development tools used in this project, I will first launch it for Android and through cross platform in the future, will also launch it in IOS, Desktop version.

Although it has some limitation due to lack of time, I will fully implement and maintain it later.