Project Name: Online Face Recognition Attendance System

Developer's Information:

Name: Arefeen Ahammad Bijay

Student Id: 221-15-5080

Section: 61_Q

Institution: Daffodil International University

Project Principle: Principle of Optimized Performance

This project addresses the Principle of Optimized Performance, focusing on:

- Minimizing Rebuilds
- Optimizing Rendering
- Network and Data Optimization
- Memory Management
- UI Responsiveness
- Device Compatibility
- Continuous Monitoring and Optimization

Project Architecture Patterns:

To efficiently address common issues in software design, this project employs the MVC and MVP architectural patterns:

MVC Pattern:

The Model-View-Controller (MVC) pattern is a software design pattern that separates an application into three main components:

- **Model:** Handles state the board and business rationale. It straightforwardly deals with the information, rationale, and rules of the application.
- **View:** Answerable for delivering UI parts. It shows the information to the client and sends client orders to the regulator.

- **Controller:** Responds to UI actions and updates the view accordingly. It acts as an intermediary between the model and the view.

Example:

In a face acknowledgment framework, the Model would deal with putting away and handling facial information, the View would show the client's face and participation status, and the Regulator would oversee connections, for example, enrolling another face or logging participation.

MVP Pattern:

The Model-View-Presenter (MVP) pattern is a subset of the MVC pattern, and its primary purpose is to organize the user interface. It simplifies testing and maintenance by separating the application logic from the user interface:

- **Model:** Governs business behaviors and state management. It handles the data part of the application.
- **View:** Renders UI components and communicates with the presenter via an interface. It is responsible for displaying data to the user.
- **Presenter:** Goes about as a broker between the model and the view, taking care of the business rationale. It recovers information from the model and applies rationale to choose what to show in the view.

Example:

In the face acknowledgment framework, the Model would deal with the facial acknowledgment information, the View would show the acknowledgment results, and the Moderator would handle the information and update the view with the participation data.

Planning and Requirements:

The project will initially be based on the MVP architecture due to its focus on UI design and presentation layer responsibilities.

Key Features:

Face Detection: The system can detect any trained face.

Attendance Logging: Basic structure for logging attendance (future work).

User Interface: Design a user-friendly UI for ease of use.

Development Tools, Methods & Environment Explanation:

1. Development Tools:

Programming Language: Python

Framework: Django for backend development and application structure.

Face Recognition Library: OpenCV and face_recognition libraries for face detection and recognition.

Database: PostgreSQL for storing user data and face recognition data.

Front-end Technologies: HTML, CSS, and JavaScript for the user interface.

Version Control: GitHub for source code management.

IDE: VSCode with extensions like Python, live-server, auto-indentation, etc.

Design Tools: Figma for UI/UX design.

2. Methods:

Development Method: Methodology for incremental and iterative development known as agile This includes normal updates, criticism cycles, and nonstop improvement of the venture.

Project Management: Jira for collaboration and project progress tracking. Sprints will be used to divide tasks, allowing for clear milestones and objectives.

3. Environment Explanation:

The development environment will consist of:

Local Development: Setting up a nearby server involving Django for improvement and testing. VSCode will be the essential IDE, outfitted with the vital augmentations for productive coding.

Version Control: Using GitHub for version control ensures that changes are tracked, and multiple team members can collaborate effectively.

Testing Environment: Separate testing environment to run unit and integration tests, ensuring the application functions correctly before deployment.

Future Features and Development of the Project:

Future enhancements will be based on user feedback and evolving requirements.

Upcoming Features:

Attendance Count: Implement functionality to count and log attendance based on face recognition.

Push Notifications: Notify users of their attendance status via push notifications.

Profile Management: Users can create and manage their profiles.

Login System: Secure login with email and password.

Forgot Password: Password recovery option with email verification.

Email Verification: Verify user emails with a 6-digit PIN.

Edit Profile: Users can update their profile information and profile images.

Calendar Integration: Users can view attendance history on a calendar.

Device Compatibility: Ensure the system works across various devices.

Data Analytics: Provide analytics and reports on attendance data.

Overview of Fully Developed Project and Conclusion:

The Web-based Face Acknowledgment Participation Framework will begin with the MVP engineering for starting turn of events, zeroing in on the UI and face identification. To deal with more complicated business logic and state management, it will eventually incorporate MVC architecture patterns.

Utilizing Python and Django will give a vigorous structure to fostering a versatile and viable application. The underlying send-off will zero in on the web stage, with tentative arrangements to broaden usefulness and back extra elements as depicted. The project will be fully implemented and maintained for optimal performance and user satisfaction, despite initial limitations.

This task means to give a powerful, proficient, and easy-to-understand face acknowledgment participation framework, smoothing out participation from the executives through trend-setting innovation and plan standards. The system's overall reliability and usability are improved by combining the MVP and MVC patterns to create an architecture that is scalable, maintainable, and testable.