PREPARED BY
Madhu Prakash Behara
Sai Rohith Avula

LAST REVIEWED ON
12/07/2023

Smart Attendance System - Project Report

1. Introduction

1.1. Purpose and Scope

The purpose of this document is to provide a comprehensive overview of the Attendance Management System (AMS), which has been designed to automate the process of recording and managing attendance through facial recognition technology. The scope of this document encompasses the system's functionality, the intended user base, and the administrative features that allow for efficient management of attendance data.

1.2. Product Overview

The AMS is a sophisticated tool that utilizes OpenCV for facial recognition, providing a seamless and efficient method for recording attendance. Its capabilities include:

- **User Registration:** Users can register their profile along with a photograph for facial recognition.
- **Automated Attendance Capture:** The system captures attendance by recognizing registered users through a camera interface.
- **Report Generation:** It generates attendance reports that can be viewed by both users and administrators.
- **Email Notifications:** AMS sends automatic email notifications upon registration, attendance capture, and report generation.

The AMS is ideal for educational institutions, corporate environments, and event management, where regular attendance tracking is essential.

1.3. Structure of the Document

This document is structured into the following main sections:

- **Project Management Plan:** Details the organization and lifecycle of the project, along with resource requirements.
- **Requirement Specifications:** Outlines the stakeholders, use cases, and non-functional requirements of the system.
- **Architecture:** Describes the architectural style, model, and technologies used in the system.
- **Design:** Provides insights into the user interface, component design, and database structure.
- **Test Management:** Lists the system test cases, testing techniques, and the defect reports.
- **Conclusions:** Summarizes the project outcomes, lessons learned, and future development prospects.

1.4. Terms, Acronyms, and Abbreviations

- **AMS:** Attendance Management System
- **OpenCV:** Open Source Computer Vision Library
- **UI:** User Interface
- **API:** Application Programming Interface
- **DB:** Database
- **SRS:** Software Requirement Specification
- **FR:** Facial Recognition

The terms and acronyms listed above are used throughout this document and are defined to ensure clarity and avoid ambiguity.

2. Project Management Plan

2.1. Project Organization

The project team is structured into various roles to ensure effective collaboration and productivity:

- **Project Manager:** Oversees project planning, scheduling, and resource allocation.
- **Development Team:** Responsible for implementing the system features.
- **Quality Assurance Team:** In charge of testing and ensuring that the system meets quality standards.
- **Database Administrator:** Manages the database operations and ensures data integrity.
- **System Analyst:** Analyzes user requirements and system functionality.
- **UI/UX Designer:** Designs the user interface and user experience.

Regular meetings are held to ensure all team members are aligned with the project goals and deadlines.

2.2. Lifecycle Model Used

The project adopts the Agile development methodology, allowing for iterative development and frequent assessment of the project's direction throughout the development process. This model is chosen to accommodate changes in user requirements and technology enhancements.

2.3. Risk Analysis

Key project risks include:

- **Technological Risk:** Challenges in integrating OpenCV with the current system.
- **Operational Risk:** Potential issues in the facial recognition system due to varied lighting conditions.
- **Management Risk:** Delays in the project timeline due to unforeseen complexities in implementation.

Mitigation strategies involve regular code reviews, rigorous testing phases, and contingency planning for delays.

2.4. Hardware and Software Resource Requirements

- **Hardware:** Server for deployment, cameras for attendance capture, workstations for development.
- **Software:** OpenCV library, Python programming language, Flask web framework, Firebase database, Git for version control.

2.4.1 Technologies Used

- **Python** with **Flask**: For backend development and web server functionality.
- **OpenCV**: For image processing and facial recognition.
- **Firebase**: For database and cloud storage services.
- **HTML/CSS/JavaScript**: For the user interface and frontend interactivity.
- **Bootstrap and DataTables**: For responsive design and enhanced data presentation.

2.4.2 Project Dependencies

- Python libraries: flask, opencv-python, face_recognition, firebase-admin, numpy, pandas, smtplib.
- Frontend technologies: Bootstrap, jQuery, DataTables.

2.5. Deliverables and Schedule

Key deliverables include:

- **Project Proposal:** Outlines the scope and objectives (Completed).
- **SRS Document:** Detailed software requirement specifications.
- **Design Specifications:** UI/UX design and system architecture
- **Test Plan and Report:** Describes testing strategies and documents results.
- **Final System:** The fully functional attendance management system.
- **User Manual:** Guides for users to navigate and use the system.
- **Developers Guide:** Documentation to assist future development and maintenance

The project schedule is maintained using a Gantt chart, with milestones aligned to sprints in the Agile development cycle.

3. Requirement Specifications

3.1. Stakeholders for the system

Stakeholders are individuals or groups that have an interest in the success and outcome of a project. For the AMS, the stakeholders include:

- **Students and Employees (Users):** Primary users of the system who will have their attendance tracked.
- **Educational Institutions and Corporates (Clients):** Organizations that deploy the AMS for attendance management.
- **System Administrators:** Individuals responsible for managing the AMS, including user account management and report generation.
- **Development Team:** The group of developers who build and maintain the AMS.
- **Quality Assurance Team:** Specialists who ensure the system meets quality and performance standards.
- **Regulatory Bodies:** Entities that ensure the system complies with privacy laws and data protection regulations.

3.2. Use cases

Use cases describe the interactions between a user and the system to achieve a goal. They are central to understanding the functional requirements of a system.

3.2.1. Graphic use case model

A graphic use case model (typically UML diagrams) is not present in the template provided and cannot be created within this text-based medium. However, it generally includes actors (representing users of the system), use cases (representing system functionality), and associations (representing interactions).

3.2.2. Textual Description for each use case

Here are textual descriptions for key use cases of the AMS:

Register User:

- **Primary Actor:** New User
- **Goal:** To create a new user profile in the system.
- **Main Success Scenario:** The user enters their details and image into the system, which are then verified and stored, creating a new user profile.
- **Extensions:** User input validation fails, and the system prompts for correct information.

Capture Attendance:

- **Primary Actor:** User
- **Goal:** To record the user's attendance through facial recognition.
- **Main Success Scenario:** The system successfully identifies the user's face and records their attendance.
- **Extensions:** The system cannot identify the user, and the user is prompted to try again or seek assistance.

Generate Report:

- **Primary Actor:** Administrator
- **Goal:** To generate attendance reports for a specified period.
- **Main Success Scenario:** The administrator selects the report criteria, and the system generates and displays the report.
- **Extensions:** There is an error in report generation due to system issues or invalid criteria selection.

Send Email Notification:

- **Primary Actor:** System
- **Goal:** To notify users of relevant events such as successful registration or attendance capture.
- **Main Success Scenario:** The system sends an automated email to the user upon the occurrence of the event.
- **Extensions:** Email fails to send due to an incorrect email address or network issues.

3.3. Rationale for the Use Case Model

The use case model for the Attendance Management System (AMS) is developed with the goal of capturing all user interactions that the system must support. This model serves as the foundation for the system design and development phases, ensuring that the final product aligns with user needs and expectations. The rationale behind this model includes:

- **User-Centric Design:** By focusing on the use cases, the development is guided by the user's perspective, ensuring that the system is intuitive and meets their requirements.
- **Facilitation of Communication:** Use cases provide a clear and straightforward way for stakeholders to discuss system functionality without requiring technical background knowledge.
- **Scope Definition:** The model helps define the boundaries of the system, clarifying which features are within scope and which are not, thereby preventing scope creep.
- **Prioritization of Features:** By identifying and describing use cases, the project team can prioritize the development tasks based on factors such as user value and technical complexity.
- **Testing and Validation Framework:** Use cases will directly inform the creation of test cases, ensuring that all functional requirements are verified through systematic testing.

The use case model is deliberately structured to be modular, allowing for flexibility in the development process. This modularity supports iterative development and enables the team to adapt to changes in user requirements or project constraints.

3.4. Non-functional Requirements

Non-functional requirements (NFRs) are criteria that specify the operation of a system, as opposed to the behaviors or functionalities. The NFRs for the AMS include:

- **Performance Requirements:**
  - The system should be capable of processing a user's attendance within 2 seconds of face recognition.
  - The system should support concurrent usage by up to 1,000 users without significant performance degradation.
- **Reliability and Availability:**
  - The AMS should be operational and accessible 24/7, with a minimum uptime of 99.5%.
  - The system should have an automated backup process and be capable of quick recovery from any failure.
- **Scalability:**
  - The system architecture should support scaling to accommodate an increasing number of users or data volume.
  - The system should maintain performance levels as the number of users grows, with the ability to scale horizontally or vertically as needed.
- **Security:**
  - The system must adhere to best practices for data security, ensuring user data is encrypted and secure against unauthorized access.
  - The system should implement authentication measures for users and administrators to protect sensitive functionalities.
- **Usability:**
  - The user interface should be intuitive, requiring minimal training for new users.
  - The system should provide clear error messages and support to guide users through resolving issues.
- **Compliance:**
  - The system must comply with all relevant legal requirements, such as data protection regulations (e.g., GDPR) and accessibility standards.
  - The system should allow for easy updates and changes to stay compliant with new regulations.
- **Maintainability:**
  - The code should be well-documented to allow for ease of maintenance and updates.

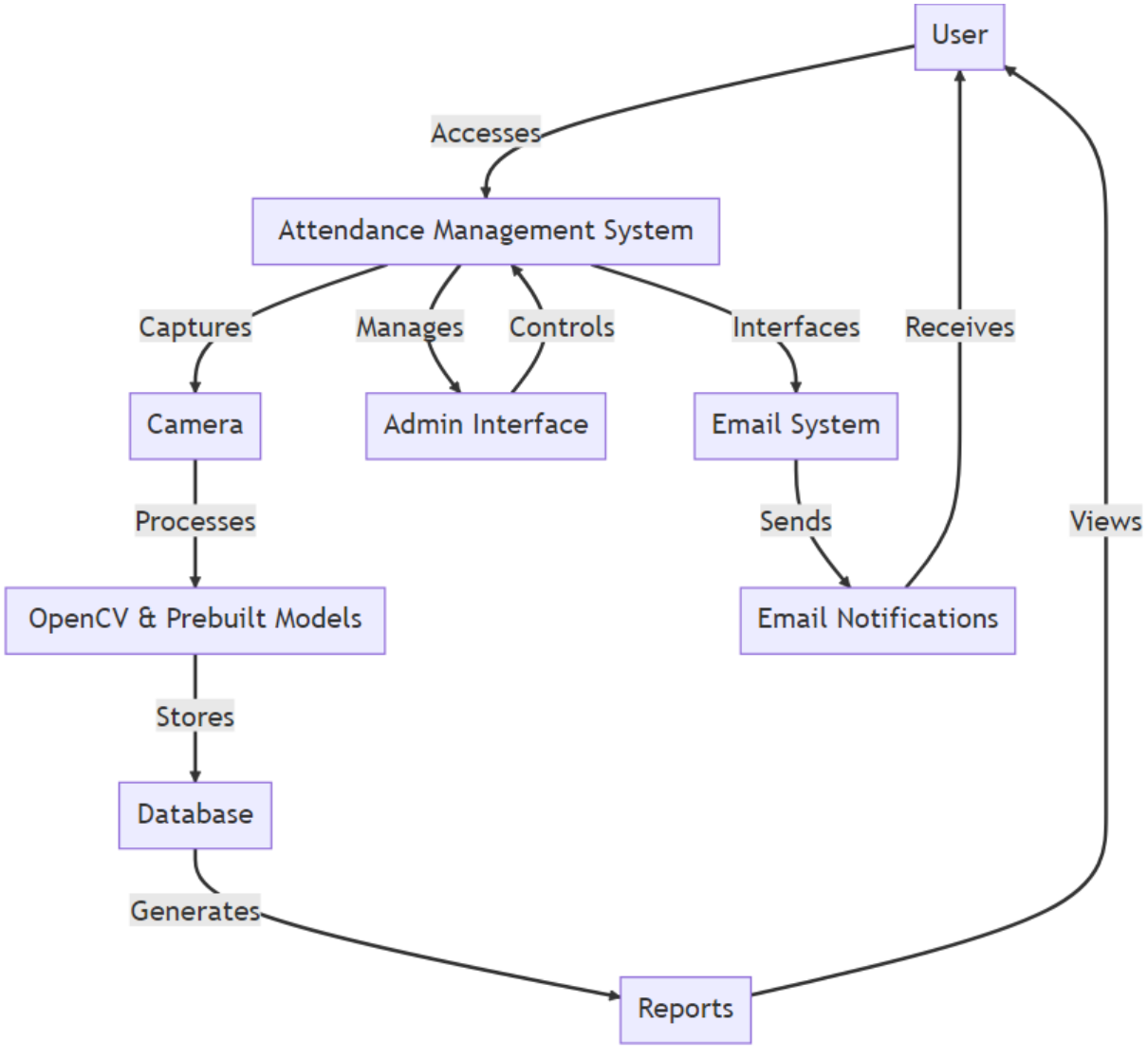
- The system should support logging and monitoring capabilities to aid in the diagnosis of issues and performance tuning.

4. Architecture

4.1. Architectural Style(s) Used

The architecture of the Attendance Management System (AMS) is primarily based on the **Model-View-Controller (MVC)** architectural style. This style is chosen for its separation of concerns, which divides the application into three interconnected components, allowing for efficient code organization, scalability, and maintainability.

- **Model:** Represents the data structure, business logic, and rules of the application.
- **View:** Represents the UI components of the application, displaying the data.
- **Controller:** Acts as an interface between Model and View components, managing user input and system output.

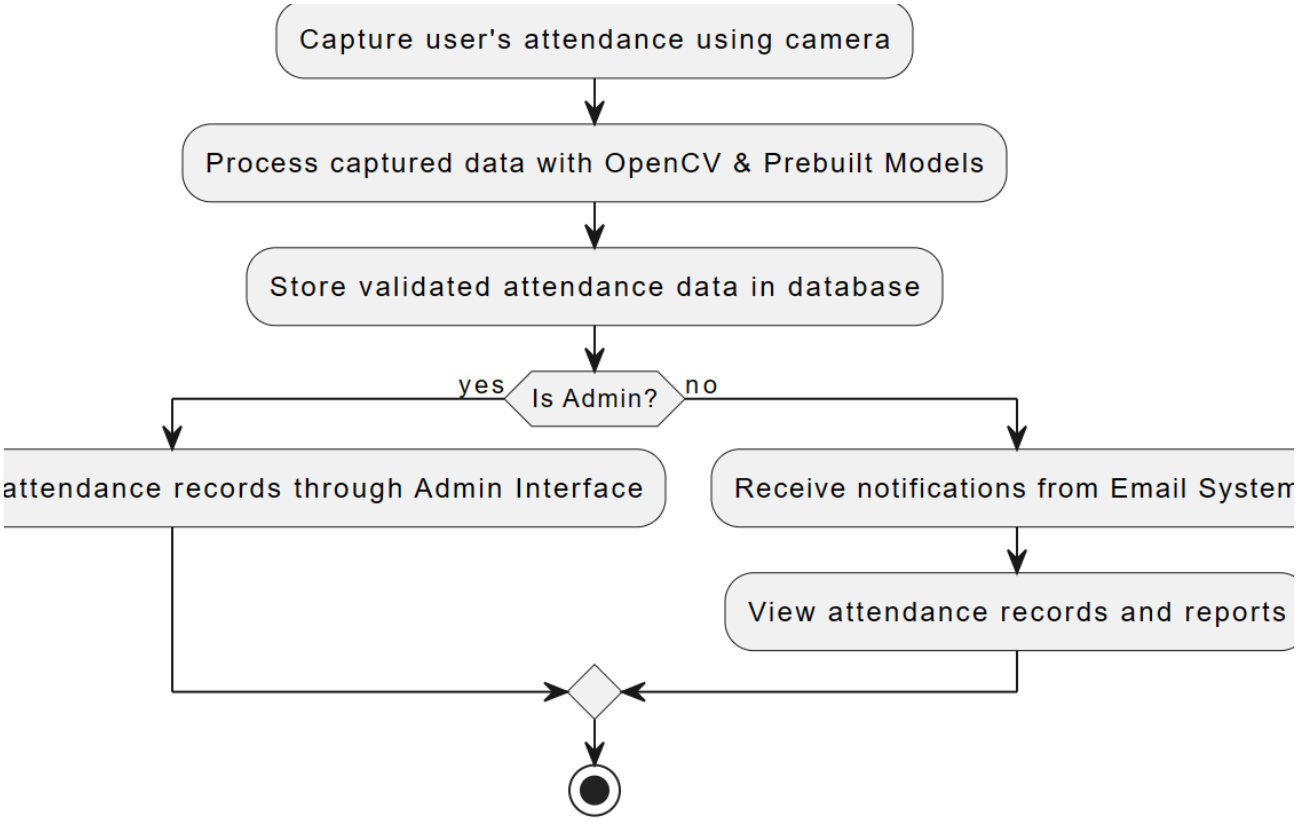


4.2. Architectural Model

The architectural model of AMS consists of several key components that interact as follows:

- **User Interface (View):** Web-based front end for user interaction, developed using HTML, CSS, and JavaScript.
- **Application Server (Controller):** Manages the HTTP requests and responses, facilitates authentication, and handles business logic. It is built using the Flask framework.
- **Database (Model):** Stores user profiles, attendance records, and other persistent data. Firebase is utilized for real-time data storage and retrieval.
- **Image Processing Service:** Integrates OpenCV for facial recognition, processing the images captured by the camera to verify user identity.
- **Email Service:** Manages sending automated email notifications for various system actions like user registration and attendance confirmation.

These components are modular and loosely coupled, allowing for independent updates and maintenance.



4.3. Technology, Software, and Hardware Used

- **Frontend Technologies:** HTML5, CSS3, JavaScript, and Bootstrap for responsive design.
- **Backend Technologies:** Python with the Flask framework for server-side logic.
- **Image Processing:** OpenCV library for facial recognition and image processing.
- **Database:** Firebase for real-time database services.
- **Server Hardware:** Cloud-based servers with adequate RAM and CPU specifications to handle the expected load.

- **Client Hardware:** Standard web browsers on desktops, laptops, or mobile devices with camera capabilities.
- **Email Notifications:** Integration with an email service provider via SMTP protocol.

4.4. Rationale for Your Architectural Style and Model

The choice of the MVC architectural style and the subsequent model is driven by several factors:

- **Scalability:** The separation of concerns within MVC facilitates scaling the application as user load increases, particularly important for institutions with large numbers of users.
- **Maintainability:** This architecture simplifies updates and maintenance. For instance, the UI can be redesigned without altering the backend logic.
- **Modularity:** MVC promotes modular development, allowing teams to work on different components simultaneously without significant overlap or conflict.
- **Technology Suitability:** The chosen technologies align with the requirements of the AMS, providing a balance between performance, ease of use, and robust community support.
- **Resource Efficiency:** The use of Flask and Firebase allows for the development of a lightweight, efficient application without the need for extensive server infrastructure, reducing overhead costs.

The selected architectural style and model aim to provide a robust framework that supports the AMS's functional and non-functional requirements while aligning with the project's overall goals and constraints.

5. Design

5.1. User Interface Design

The User Interface (UI) for the Attendance Management System (AMS) is designed to be user-friendly, accessible, and responsive across various devices. The UI includes:

- **Login/Registration Screens:** Secure input forms for user authentication and new user registration.
- **Dashboard:** A central hub for users to access their attendance records, notifications, and profile settings.
- **Admin Panel:** A specialized interface for system administrators to manage users, generate reports, and configure system settings.
- **Notification System:** Real-time alerts and email notification previews within the application.
- **Report Views:** Pages where users can view detailed attendance reports, with filtering and search capabilities.

The UI design employs a color scheme and typography consistent with the organization's branding and ensures compliance with WCAG 2.0 accessibility standards.

5.2. Components Design

- **Frontend Component:** Utilizes MVC architecture with templates for the view, JavaScript for dynamic interaction, and AJAX calls for asynchronous communication with the backend.
- **Backend Component:** Flask routes acting as controllers and Python classes for models, handling business logic and database interactions.

- **Image Processing Component:** A service component that interfaces with the OpenCV library to handle image capture, processing, and facial recognition.
- **Database Component:** Firebase real-time database with a schema designed for efficient data retrieval and updates.

Static and dynamic models for each component would be diagrammed using UML to illustrate class structures and interactions, sequence diagrams for user interactions, and activity diagrams for workflows.

5.3. Database Design

The database schema is designed to support the AMS efficiently:

- **Users Table:** Stores user credentials, profile information, and roles.
- **Attendance Table:** Records timestamps, user IDs, and attendance status.
- **Reports Table:** Aggregates attendance data for report generation.

RegisteredStudents Node

- This node stores data related to students who have registered for the facial recognition-based attendance system.
- Each student is identified by a unique ID.
- Contains the following fields for each registered student:
  - image_url: A string representing the file path or URL to the student's image used for facial recognition.
  - last_attendance_time: A timestamp indicating the last time the student's attendance was recorded by the system.

Example:

jsonCopy code

```
"RegisteredStudents": {
  "2837016": {
    "image_url": "resources/images/2837016.png",
    "last_attendance_time": "2023-12-07 16:56:52"
  }
}
```

Students Node

- This node maintains comprehensive records of all students, including those registered for facial recognition.
- Each entry includes both personal and academic information, along with attendance data.
- Fields for each student:
  - last_attendance_time: The most recent timestamp of attendance recording.
  - total_attendance: An integer count of the total number of times the student's attendance has been recorded.

- major: The student's major field of study.
- name: The full name of the student.
- standing: A character or string indicating the academic standing of the student (e.g., "B" for Bachelor, "G" for Graduate).
- starting_year: The year the student started their current academic program.
- year: The current year of study for the student.

Example:

jsonCopy code


```
"Students": {
  "2837016": {
    "last_attendance_time": "2023-12-07 16:58:18",
    "major": "SE",
    "name": "Sai Rohith Avula",
    "standing": "B",
    "starting_year": 2021,
    "total_attendance": 3,
    "year": 1
  }
}
```


Design Considerations

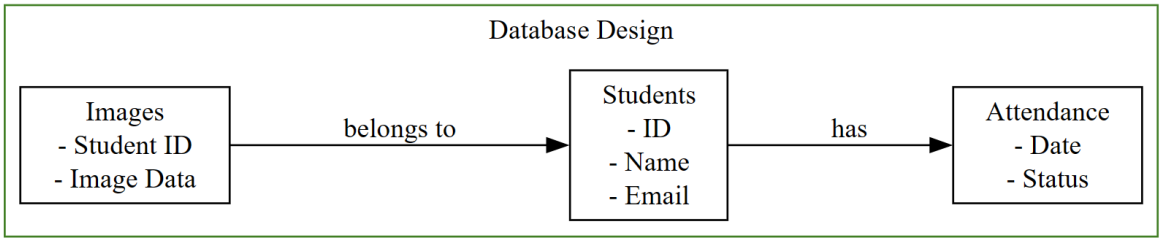
- **Normalization vs. Denormalization**: The database adopts a slightly denormalized structure for efficiency. While this can lead to some data redundancy (e.g., last_attendance_time appearing in both nodes), it optimizes the performance for read-heavy operations, which is typical in attendance systems.
- **Scalability**: The design is scalable, accommodating an increasing number of students and attendance records. Firebase Realtime Database's cloud-based architecture supports this scalability.
- **Security**: Proper security rules need to be implemented in Firebase to protect sensitive student data and ensure that access is restricted to authorized users only.

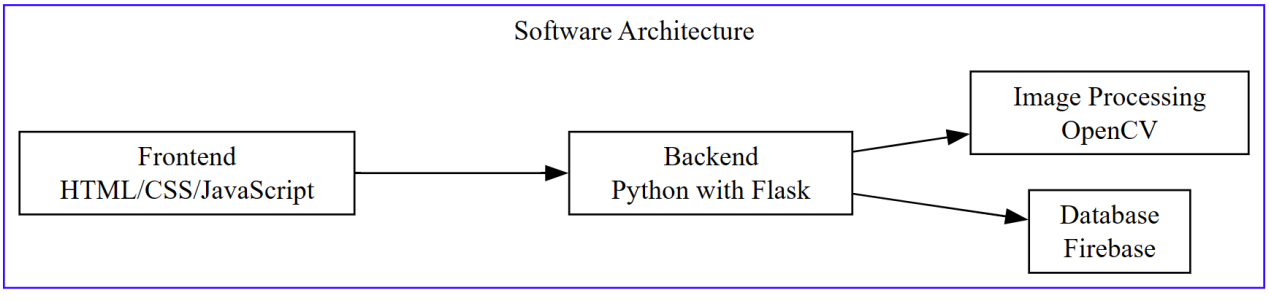
5.4. Rationale for Your Detailed Design Models

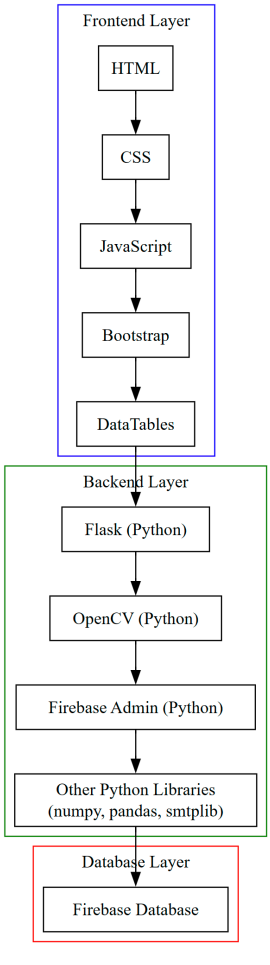
The detailed design models are created with the following considerations:

- **Usability:** The UI is designed to ensure ease of use, with a focus on minimizing the number of user steps required to perform common tasks.
- **Efficiency:** Components are designed to handle requests quickly and conserve resources, with a stateless backend suitable for cloud deployment.

- **Scalability:** The database is structured to allow for easy expansion, supporting an increasing number of users and larger data sets.
- **Security:** Security principles are applied in all designs, especially in user authentication and data protection.







-

5.5. Traceability from Requirements to Detailed Design Models

Traceability is maintained through a matrix that maps each requirement to its corresponding design element:

- **Functional Requirements:** Each user action defined in the use cases is traceable to specific UI elements, backend services, and database operations.
- **Non-Functional Requirements:** Performance, security, and usability requirements are linked to the architectural choices, technology stack, and design strategies.

This traceability ensures that all requirements are accounted for in the design and that any changes to the requirements can be reflected accurately in the design documentation.

6. Test Management

6.1. A Complete List of System Test Cases

A comprehensive set of test cases is designed to cover all functionalities of the Attendance Management System. These may include:

Test Case ID 001 - User Registration:

- Description: Ensure that a new user can register successfully.
- Precondition: New user has not been registered before.
- Test Steps: Navigate to the registration page, fill in all required fields, and submit.
- Expected Result: User is registered, and a confirmation email is sent.

Test Case ID 002 - Login Functionality:

- Description: Verify that registered users can log in with valid credentials.
- Precondition: User is already registered.
- Test Steps: Navigate to the login page, enter valid credentials, and submit.
- Expected Result: User is logged in and directed to the dashboard.

Test Case ID 003 - Facial Recognition Accuracy:

- Description: Confirm that the facial recognition process accurately recognizes registered users.
- Precondition: User's facial data is registered in the system.
- Test Steps: User stands in front of the camera for attendance.
- Expected Result: The system recognizes the user and records attendance.

Test Case ID 004 - Report Generation:

- Description: Validate the ability to generate attendance reports.
- Precondition: Attendance data is available.
- Test Steps: Admin requests a report for a specific period.
- Expected Result: The system generates the report accurately.

6.2. Traceability of Test Cases to Use Cases

Each test case is linked to a specific use case to ensure that all functional requirements are verified:

- **Test Case ID 001** traces back to the **Register User** use case.
- **Test Case ID 002** traces back to the **Login** use case.
- **Test Case ID 003** traces back to the **Capture Attendance** use case.
- **Test Case ID 004** traces back to the **Generate Report** use case.

6.3. Techniques Used for Test Case Generation

- **Boundary Value Analysis:** To test the edge cases of input fields in the registration and login processes.
- **Equivalence Partitioning:** To reduce the number of test cases by grouping inputs that should be treated the same.
- **State Transition Testing:** To ensure that the system transitions correctly between different user states (e.g., logged in/out).
- **Error Guessing:** To explore potential error conditions in the facial recognition process.

6.4. Test Results and Assessments

- **Effectiveness of Test Cases:** The test cases are reviewed regularly to ensure they are effective in finding defects and covering all scenarios.
- **Software Quality Assessment:** The defect discovery rate, severity of defects, and the ease of reproducing and fixing the defects provide insight into the software quality.

6.5. Defects Reports

Defects discovered during testing are documented with the following details:

- **Defect ID:** Unique identifier for the defect.
- **Description:** A detailed explanation of the defect.
- **Severity:** The importance of the defect based on its impact on the system.
- **Steps to Reproduce:** Clear steps that detail how to reproduce the defect.
- **Status:** Current status of the defect (e.g., Open, In Progress, Fixed).

These reports are used to track and manage the resolution of defects, ensuring that they are addressed in a timely and efficient manner.

7. Conclusions

7.1. Outcomes of the Project

The project successfully delivered a functional Attendance Management System tailored to the needs of educational institutions. The system achieved its primary goals:

- Automated attendance recording using facial recognition.
- Real-time attendance tracking and reporting.
- User-friendly interface for both students and administrators.
- Email notifications for attendance events.

However, some stretch goals, such as mobile app integration, remain unmet and are earmarked for future development.

7.2. Lessons Learned

Throughout the project, the team gained invaluable insights:

- The importance of iterative testing and continuous integration in catching defects early.
- Effective communication among team members and stakeholders is crucial for project alignment.
- Adopting Agile methodologies facilitated flexibility and adaptability in development.
- The significance of thorough documentation for future maintenance and scalability.

Challenges faced included integration issues with the facial recognition library and ensuring the system's usability across various devices.

7.3. Future Development

Moving forward, the project could be expanded with the following enhancements:

- Development of a dedicated mobile application to facilitate easier access.
- Integration of advanced machine learning algorithms for improved facial recognition accuracy.
- Implementing additional security features, such as two-factor authentication.
- Expanding the reporting features to include analytics on attendance trends.

The project laid a solid foundation for an extensible system that can evolve with technological advancements and changing user needs.