Attendance System

Mentor : Prof. Sanjay Kumar Pal

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

Name	Mohsin Ansari	Pranab Ghosh
Course	BCA (5th SEM)	BCA (5th SEM)
Roll No.	23401221027	23401221006
Paper	Minor Project	Minor Project
Paper Code	BCAD581	BCAD581

ATTENDANCE SYSTEM

ACKNOWLEDGEMENT

I would like to express my gratitude to the all the individuals who have contributed directly or indirectly to the success of this Attendance System project.

Face-API-JS: A special thanks to face-api-js for providing the powerful facial recognition capabilities that form the backbone of this system.

Open Source Community: Thanks to the entire open-source community for creating and maintaining the various libraries and tools that were instrumental in building this project.

Contributors: I appreciate the contributions from anyone who has provided feedback, reported issues, or contributed code to this project.

Mentors and Advisors: A heartfelt thank you to Prof. Sanjay Pal who have guided us throughout the development process.

Users: Last but not least, thank you to all the users who have tested and used this Attendance System, providing valuable insights for improvement.

Team member's Signature

Mohsin Ansari

Pranab Ghosh

CERTIFICATE

This certificate is to acknowledge that the minor project entitled "Attendance System" has been completed by Mohsin Ansari & Pranab Ghosh under my guidance. The work presented in this project is original and has not been submitted elsewhere for any purpose. The successful completion of this project reflects the dedication and hard work put forth by the students.

Professor Sanjay Kumar Pal
[NSHM Knowledge Campus]

Date:

Index

Introduction

The Attendance System is a cutting-edge web application designed to streamline attendance tracking using facial recognition technology. This project offers a user-friendly interface for managing attendance, facilitating efficient and accurate tracking of individuals. With features such as dynamic real-time updates, user authentication, and downloadable reports, the Attendance System provides a comprehensive solution for modern attendance management. Built on powerful technologies, including **Face-API-JS**, this project represents a fusion of innovative software engineering and emerging facial recognition capabilities.

Features :

Facial Recognition: Utilizing the capabilities of Face-API-JS, our system ensures accurate and secure attendance tracking through facial recognition.

Real-time Updates: Experience dynamic updates as attendance data is instantly reflected, providing administrators with the latest information.

User Authentication: Ensure data security with robust user authentication, allowing only authorized users to access and manage attendance data.

Downloadable Reports: Easily generate and download comprehensive attendance reports for efficient record-keeping and analysis.

Objectives of Attendance System :

The primary objective of the Attendance System is to revolutionize traditional attendance tracking methods by implementing facial recognition technology. The project aims to provide a comprehensive and user-friendly solution for efficient attendance management, catering to various industries such as education and corporate sectors. Key objectives include:

Accuracy and Precision:
Implementing facial recognition through Face-API-JS to ensure accurate and precise attendance tracking, minimizing errors associated with traditional methods.

Real-time Updates:
Facilitating real-time updates of attendance data, allowing administrators to access the latest information instantly.

User Authentication:
Prioritizing data security with robust user authentication mechanisms to control access and maintain the integrity of attendance records.

Simple User Interface:
Creating an user-friendly web interface to enhance the overall user experience and simplify the process of attendance monitoring.

SRS (Software Requirement Specification) :

Software Requirements Specification (SRS) is a document that specifies the requirements for a software system. It is a detailed description of the software that is being developed and includes information such as the functional requirements, performance requirements, design constraints, and other factors that need to be taken into consideration during the development process. The SRS is used as a reference point throughout the software development life cycle and helps to ensure that the final product meets the needs and expectations of the customer or end user.

Purpose

The purpose of the Attendance System is to provide a modern and accurate solution for attendance tracking, eliminating the shortcomings of traditional methods. The system utilizes facial recognition through the Face-API-JS library to enhance precision and streamline the process.

Scope

The system is intended for use in educational institutions, corporate environments, and any other scenario where accurate attendance monitoring is crucial. It encompasses user authentication, real-time updates, intuitive interfaces, and comprehensive reporting.

Technical Specifications (Tech Stacks) :

1. Visual Studio Code

- Visual Studio Code (VS Code) is a lightweight, open-source code editor developed by Microsoft. It is known for its flexibility, extensive language support, and a rich set of features that enhance the development experience.
- Its support for various programming languages, extensions, and built-in Git integration makes it a versatile choice for web development projects.
- VS Code provides a clean and user-friendly interface, allowing developers to focus on code without distractions.
- The editor supports a vast array of extensions that can be installed to enhance functionality, ranging from language support to debugging tools.
- VS Code includes a built-in terminal, enabling developers to run commands, scripts, and interact with the development environment without leaving the editor.
- Git integration is seamlessly integrated into VS Code, allowing for version control operations, commit history visualization, and branch management.
- VS Code features intelligent code completion, syntax highlighting, and error checking, providing a smooth coding experience.
- The built-in debugger simplifies the process of identifying and fixing issues in the code by providing step-through and breakpoint functionality.

Visual Studio Code

2. Node.js

- Node.js is an open-source, cross-platform JavaScript runtime environment built on the V8 JavaScript engine. It allows developers to execute JavaScript code server-side, enabling the development of scalable and high-performance network applications.
- The ability to use JavaScript for both server-side and client-side development streamlines the development process and promotes code reuse.
- Node.js is built on the V8 JavaScript engine, known for its high-performance execution of JavaScript code.
- npm, the Node.js package manager, provides access to a vast ecosystem of open-source libraries and tools that can be easily integrated into Node.js projects.

3. Express.js

- Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications.
- It facilitates the creation of server-side logic and simplifies the handling of HTTP requests and responses.
- Express.js serves as the primary web application framework for the Attendance System, enabling the definition of routes, middleware, and the overall structure of the server-side code.
- **Routing:** Express.js allows the definition of routes to handle different HTTP methods and URL patterns, making it easy to structure the application's endpoints.
- **Middleware:** Middleware functions in Express.js provide a way to execute code during the request-response cycle. This is used for tasks such as logging, authentication, and error handling.
- **Template Engine Support:** Express.js supports various template engines like EJS, allowing the dynamic generation of HTML content on the server.

4. EJS (Embedded Javascript)

- EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. It provides a straightforward way to embed dynamic content and logic directly into HTML files.
- **Partial Views:** EJS supports the inclusion of partial views, allowing the reuse of common HTML components across multiple pages.
- EJS is the chosen template engine for the Attendance System, allowing the server to generate dynamic HTML content based on data from the MongoDB database.

5. CSS (Cascading Style Sheets)

- CSS is a stylesheet language used for describing the presentation of a document written in HTML or EJS, including colors, layouts, and fonts.
- CSS is utilized to style and design the user interface of the Attendance System web application, providing a visually appealing and responsive layout.
- EJS is the chosen template engine for the Attendance System, allowing the server to generate dynamic HTML content based on data from the MongoDB database.

6. MongoDB

- MongoDB is a NoSQL database that provides a flexible, schema-less data model, making it suitable for storing and managing large volumes of unstructured data.
- **Document-Oriented**: MongoDB stores data in JSON-like Collections & documents, allowing for a flexible and dynamic schema.
- **Scalability:** MongoDB is designed to scale horizontally, making it suitable for handling large datasets and high read and write loads.
- **Query Language:** MongoDB supports a rich query language, making it easy to retrieve and manipulate data.
- **Usage:** Attendance records, user details, and other relevant data in the Attendance System are stored in MongoDB collections.

7. Mongoose

- Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It provides a schema-based solution to model the application data and simplifies interactions with MongoDB databases.
- **Schema Definition:** Mongoose enables the definition of schemas with typed fields, validation rules, and default values.
- **Query Building:** Mongoose provides a query builder API, making it easier to construct complex MongoDB queries using a fluent syntax.
- **Usage**

1. Mongoose is initialized with a MongoDB connection URI and is used to define models based on schemas.
2. Models are employed to interact with MongoDB collections, providing methods for CRUD operations.
3. Mongoose is used in the Attendance System project to define schemas for the user and attendance data, allowing for structured data storage and retrieval.

- **Schema (Structure):** Users schema which contains credentials like username, email, password. Attendance Schema contains the Name, Batch, DateTime.

8. Bcrypt (Password Hashing)

- Bcrypt is a widely-used cryptographic hashing library designed to securely hash passwords. It employs the bcrypt hashing algorithm, which incorporates salt to enhance password security.
- **Salted Hashing:** Bcrypt automatically generates and applies a unique salt to each hashed password, adding a layer of complexity to thwart rainbow table attacks.
- **Ease of Use:** Bcrypt is straightforward to integrate into Node.js applications, providing a simple API for hashing and verifying passwords.
- **Usage**

1. Bcrypt is typically used during user registration to hash and store passwords securely.
2. During login, bcrypt is employed to verify the entered password against the stored hash.
3. Bcrypt is utilized in the Attendance System project to securely hash and verify user passwords, safeguarding sensitive authentication information.

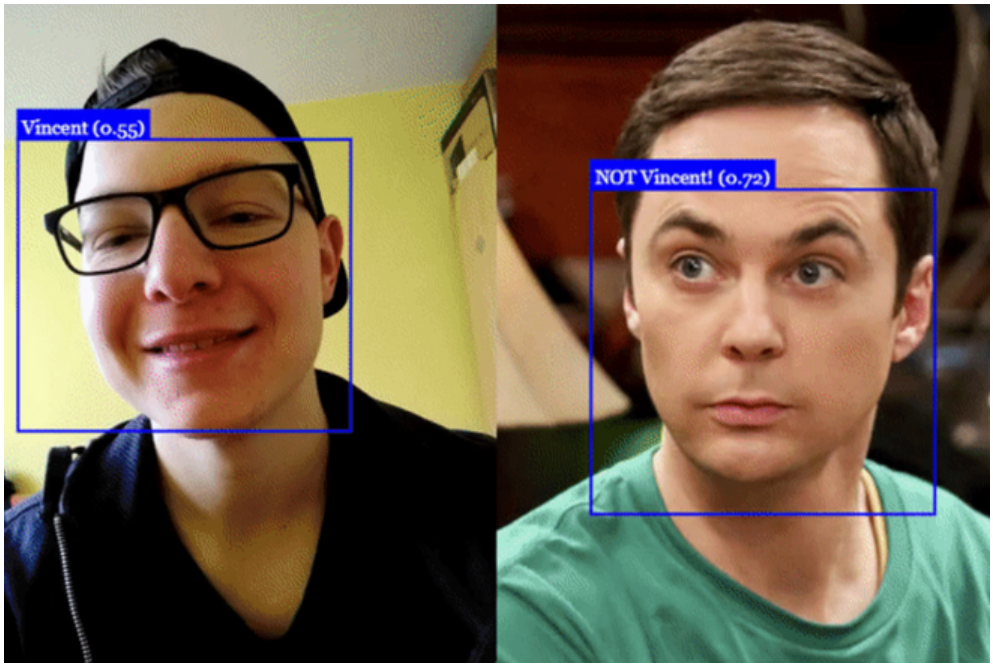
9. face-api.js (Facial Recognition Library)

- face-api.js is a JavaScript library for face detection and recognition in the browser or Node.js environment. It leverages machine learning, pre-trained model capabilities and provides a convenient API for facial analysis tasks.
- **Face Detection:** face-api.js offers robust face detection, identifying facial landmarks and bounding boxes.
- **Face Recognition:** The library supports face recognition, allowing the association of detected faces with known individuals.
- **Usage**
  In the Attendance System project, face-api.js is employed for real-time facial recognition, allowing automatic tracking and recording of attendance based on recognized faces.
- **Considerations**

1. Proper lighting conditions and clear camera feed contribute to the accuracy of face detection and recognition.
2. Regular updates to face-api.js ensure compatibility with the latest versions of TensorFlow.js and maintain security.

- **License**

1. face-api.js is released under the MIT License, permitting its usage, modification, and distribution in both open-source and proprietary projects.
2. face-api-js is open for contributions.



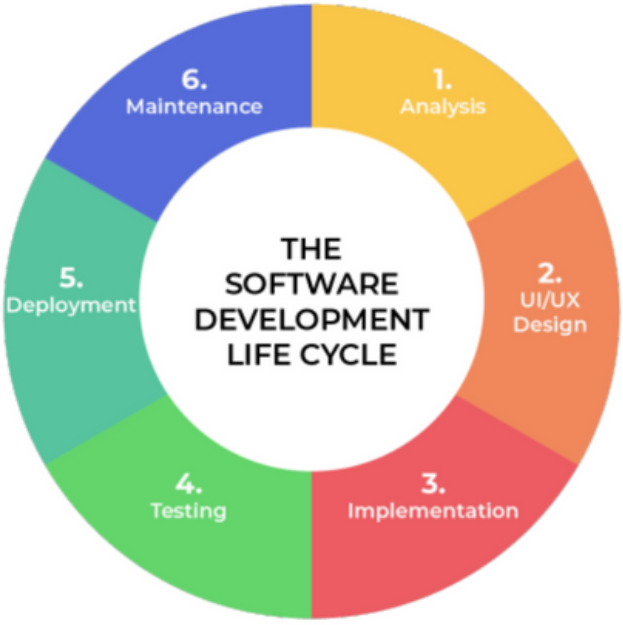
Software Development Life Cycle

SDLC :

The Software Development Life Cycle (SDLC) is a process followed by software development teams to design, build, test, and deploy high-quality software.

The key stages in SDLC are as follows:

1. **Planning:** Define the project scope, requirements, budget, and timeline.
2. **Analysis:** Gather and analyze user requirements to understand the system's functionality.
3. **Design:** Create the architecture and design the system's components based on the gathered requirements.
4. **Implementation:** Develop the actual code and integrate different components to form a complete system.
5. **Testing:** Conduct various tests to identify and fix bugs or issues in the software.
6. **Deployment:** Release the software for public use or deploy it within the organization.
7. **Maintenance:** Provide ongoing support, fix bugs, and implement updates or enhancements.



Iterative Model

The Iterative Model involves repeating cycles of development, testing, and refinement. Here are the key steps in the Iterative Model:

- **Requirements Gathering:**
1. Collect and document the initial set of requirements.
2. Identify the core features and functionalities needed for the software.

- **Design:**
1. Create a preliminary design based on the initial requirements.
2. Define the architecture and system components.

- **Implementation:**
1. Develop a partial, functional version of the software based on the initial design.
2. Implement a subset of features to create a working prototype.

- **Testing:**
1. Conduct testing on the implemented features to identify defects and issues.
2. Gather feedback from users and stakeholders.

- **Deployment:**
1. After completing all the phases, software is deployed to its work environment.

- **Feedback and Evaluation:**
1. Collect feedback from users, clients, and testing teams.
2. Evaluate the software's performance and usability.

- **Refinement:**
1. Based on feedback, refine and improve the design and implementation.
2. Add new features or modify existing ones.

- **Repeat:**
1. Repeat the cycle with each iteration, incorporating lessons learned from the previous cycle.
2. Continue to refine and expand the software in subsequent cycles.

- **Continuous Iterations:**
1. Continue the iterative process until the software meets the desired level of quality and functionality.
2. Each iteration brings the software closer to its final state.

Use of Iterative Model in The Project

- **Building Step by Step:**
  - We didn't try to build everything at once. Instead, we took small steps, like adding facial recognition, user authentication, and attendance tracking, one at a time.
- **Listening to Users:**
  - We kept asking users what they thought and made changes based on their feedback. This way, we could make the system more user-friendly and aligned with their needs.
- **Fixing as We Go:**
  - If something wasn't working quite right, we didn't wait until the end to fix it. We made improvements and fixed issues in each round of development.
- **Trying New Things:**
  - When we decided to use face-api.js for facial recognition, we knew it might have challenges. The iterative model allowed us to adapt and figure things out as we went along.
- **Making It Better:**
  - Each time we went through a development cycle, we tried to make things better. Whether it was accuracy in recognizing faces or adding features like viewing attendance reports, we kept improving.
- **Flexibility for Changes:**
  - Things changed as we developed, and that's okay. The iterative model gave us the flexibility to change our plans and adapt to new ideas and technologies.
- **Getting Better with Time:**
  - It's like making a recipe. You try it, taste it, and then adjust. Our attendance system got better and better with each iteration.
- **Testing Early and Often:**
  - We didn't wait until the end to see if everything worked. Testing was a regular part of the process, helping us catch issues early and fix them quickly.

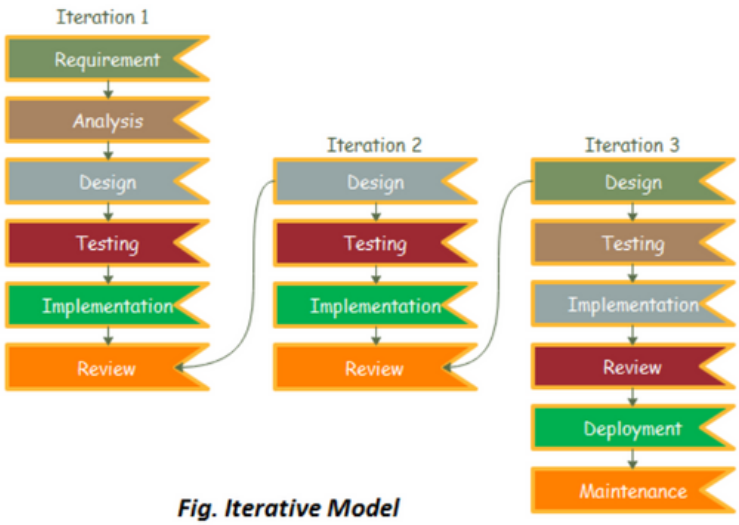
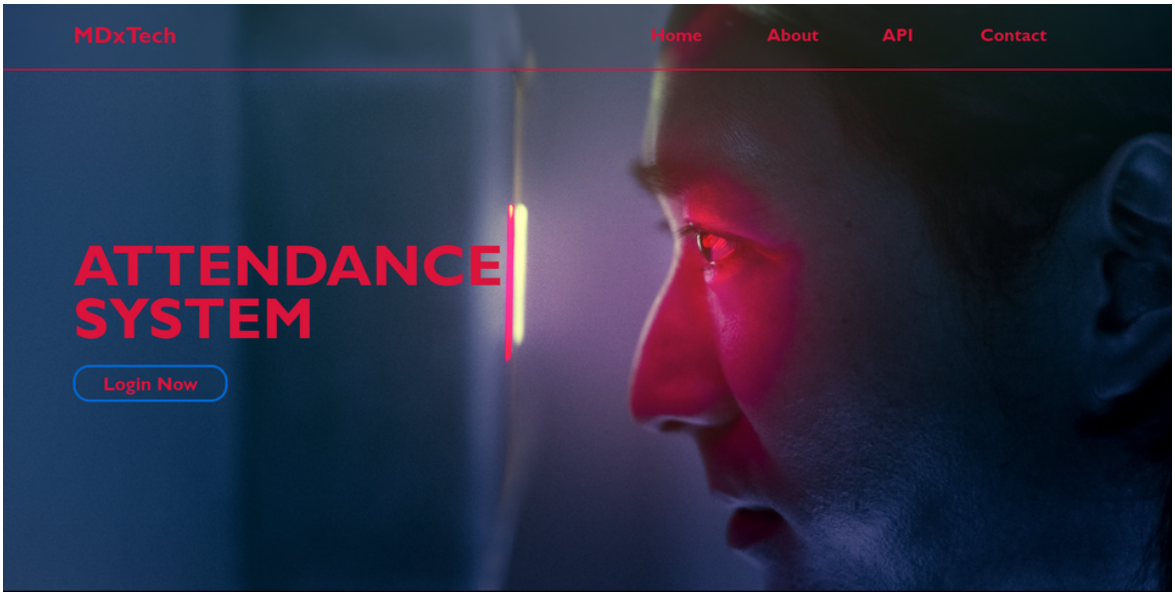


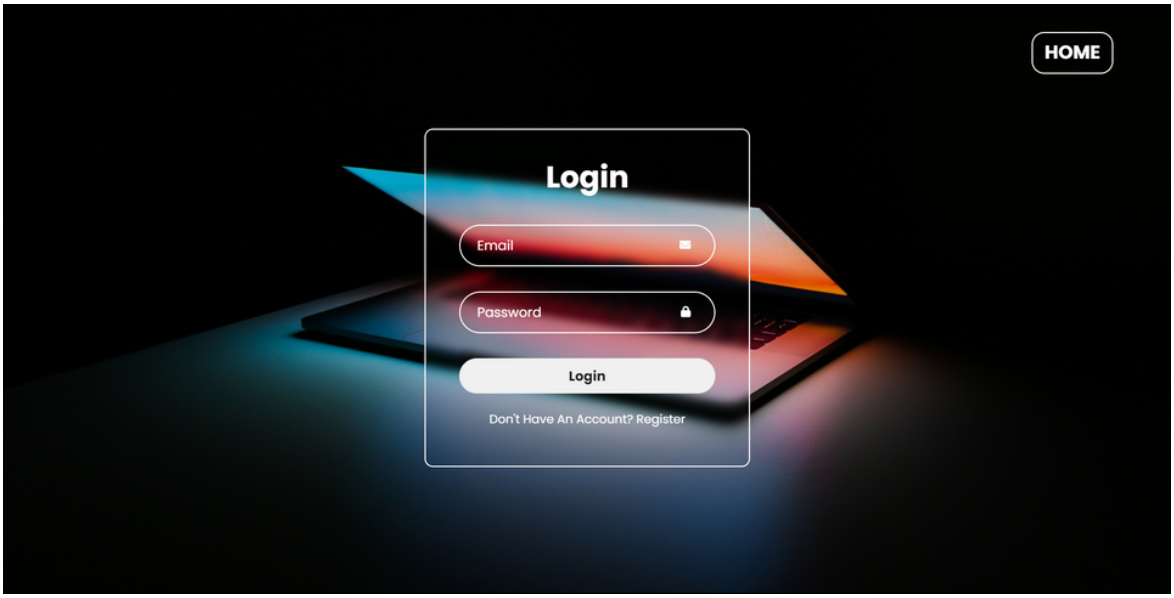
Fig. Iterative Model

Project Views

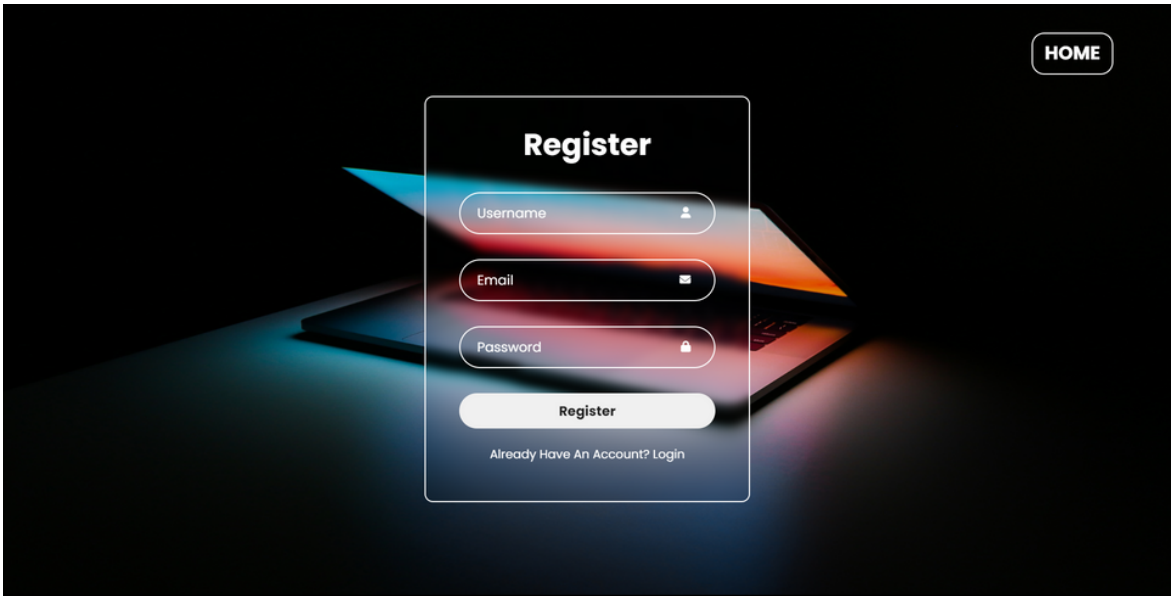
Landing Page



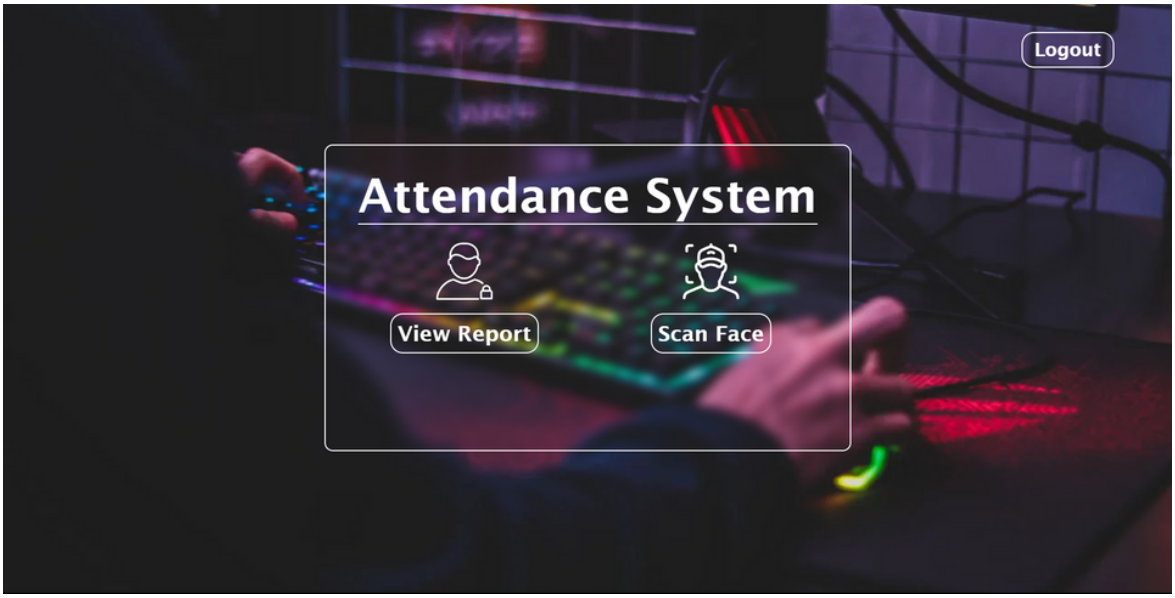
Login Page



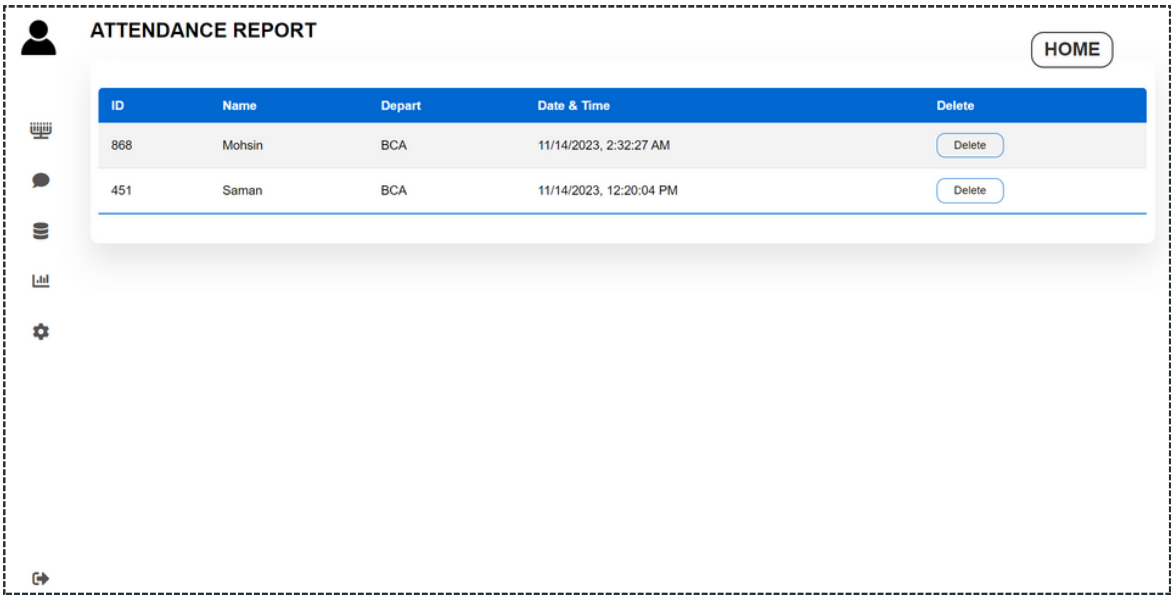
Register Page



Home Page



Attendance Report Page



Attendance Scanner Page

Testing the Attendance System :

Testing is the process of evaluating a system or application to identify any errors, gaps, or missing requirements in contrast to the actual requirements. The goal is to ensure that the software behaves as expected and meets the requirements.

1. **Unit Testing:**
   - We conducted thorough unit testing for individual components of the system, ensuring that functions, modules, and services worked as expected.
2. **Integration Testing:**
   - Integration testing was crucial to verify that different parts of the system, such as the facial recognition module, user authentication, and MongoDB database, worked seamlessly together.
3. **End-to-End Testing:**
   - We performed end-to-end testing to simulate real-world scenarios, checking the entire flow from facial recognition to data storage and report generation.
4. **User Testing:**
   - User testing played a significant role. We invited potential users to interact with the system, providing insights into its user-friendliness and identifying areas for improvement.
5. **Scalability Testing:**
   - To prepare for potential growth, we conducted scalability testing to assess how the system performed under varying loads, ensuring it could handle increased usage.
6. **Security Testing:**
   - Security was a top priority. We performed security testing to identify and address vulnerabilities, safeguarding user data and system integrity.
7. **Performance Testing:**
   - Performance testing helped optimize the system's responsiveness. We measured load times, response times, and resource utilization to ensure a smooth user experience.
8. **User Acceptance Testing (UAT):**
   - Before the official launch, we organized user acceptance testing sessions, allowing users to validate that the system met their requirements and expectations.
9. **Bug Tracking and Resolution:**
   - Bugs identified during testing were promptly logged, tracked, and resolved. This iterative process improved the system's stability and reliability.
10. **Adapting to User Suggestions:**
    - Users' suggestions and observations during testing influenced our development decisions. This iterative approach allowed us to align the system with user needs.

Deployment & Source Code

Deployment:
The Attendance System project has been successfully deployed using Cyclic, a platform for continuous deployment. This deployment method ensures that the latest changes to the project are automatically reflected in the live application, providing a seamless and up-to-date experience for users. The deployment on Cyclic simplifies the deployment process, making it efficient and allowing for quick updates.
Deployment Link : ClickHere

Source Code:
The complete source code for the Attendance System project is available on the GitHub repository with License. This repository serves as a centralized location for the project files, including server-side scripts, frontend code, Installation and configuration. Users and developers can access the source code, contribute to the project, or explore the implementation details. The GitHub repository enhances collaboration and transparency in the development process.
Source Code Link: ClickHere

Conclusion

In conclusion, the Attendance System project leverages facial recognition technology to create an efficient and modern approach to attendance management. The use of face-api.js, along with Node.js and MongoDB, forms a robust foundation for the system. The iterative development model facilitated a systematic and adaptive process, allowing for continuous improvements and bug fixes.

The project's deployment on Cyclic ensures seamless updates, while the availability of the source code on GitHub encourages collaboration and transparency. The user-friendly interface, real-time attendance tracking, and downloadable reports contribute to the system's effectiveness. Overall, the Attendance System represents a successful integration of cutting-edge technologies to streamline and enhance the traditional process of attendance tracking.

Thank You