Implementation Document

for

S.M.A.R.T.E.X

Version 2.0

Prepared by

Group #: 16 **Group Name:** BitEngine

Danish Mehmood	210297	danishm21@iitk.ac.in
Rikesh Sharma	180606	rikesh@iitk.ac.in
Anuj Kumar	200164	kanuj20@iitk.ac.in
Kumar Gourav	170353	songara@iitk.ac.in
Ajeet Meena	210078	ajeetm21@iitk.ac.in
Lohit P Talavar	210564	lohitpt21@iitk.ac.in
Avinash Saini	200232	avinashs20@iitk.ac.in
Pratiksha Dawane	210760	ppdawane21@iitk.ac.in

Course: CS253

Mentor TA: Rumit

Date: 22/04/2023

CONTENTS....... II
REVISIONS....... II

1 IMPLEMENTATION DETAILS....... 1

2 CODEBASE 2

3 COMPLETENESS....... 3

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
2.0	Danish Mehmood	Completeness of the implementation and future development plans. Final draft	22/04/2023

1 Implementation Details

The Serverless Student Entry-Exit System is designed to automatically track the entry and exit of students into a school or specific area. The system utilizes a combination of backend and frontend components to detect and record the movement of students in and out of a designated location. This document outlines the steps required to implement the Serverless Student Entry-Exit System.

Technology Used:

- **AWS Lambda with NodeJS 16**

  AWS Lambda is a good choice for implementing a serverless application like the Serverless Student Entry-Exit System for several reasons:

  1. **Cost-Effective**: AWS Lambda is a serverless computing service, meaning that you only pay for the actual execution time of your code. This can result in significant cost savings compared to running and maintaining your own servers.
  2. **Scalability**: AWS Lambda is designed to be highly scalable, automatically scaling up and down based on the number of incoming requests. This ensures that the system can handle sudden spikes in traffic without any performance issues.
  3. **Availability**: AWS Lambda is a fully managed service that is designed for high availability. It automatically replicates your code across multiple availability zones to ensure that it is always available to handle requests.
  4. **Ease of Development**: AWS Lambda is designed to make it easy for developers to build and deploy serverless applications. It supports a variety of programming languages, including Node.js, Python, Java, and more, and provides a simple way to integrate with other AWS services like DynamoDB and API Gateway.
  5. **Security**: AWS Lambda provides a secure runtime environment for executing your code, including automatic security updates and patches.

- **Amazon API Gateway**

- **AWS IAM**

  We selected AWS IAM because it helps to collaborate while building the backend services for the system and test the backend parallelly using the **POSTMAN** for making frontend request.

- **POSTMAN**

  1. **Versatility**: POSTMAN supports a wide range of API request methods, including GET, POST, PUT, DELETE, and more. It also allows you to set headers, parameters, and authentication credentials for your requests.

  2. **Collaboration**: POSTMAN provides collaboration features that allow multiple team members to work together on testing and development of the API. This includes the

ability to share collections, create workspaces, and integrate with other collaboration tools like GitHub

- **Amazon DynamoDB**

  Amazon DynamoDB is a good choice for the database of the Automatic Entry Exit System for several reasons:

  1. **Scalability**: DynamoDB is a highly scalable NoSQL database that can handle large amounts of data and high traffic loads. It can automatically scale up or down based on the demand, which ensures that the database can handle sudden spikes in traffic without any performance issues.
  2. **Performance**: DynamoDB provides fast and predictable performance, with single-digit millisecond latency at any scale. This ensures that the system can handle requests quickly and efficiently, even as the database grows.
  3. **Flexibility**: DynamoDB is a flexible database that can handle a wide variety of data types and structures. This allows you to store data in a way that is most efficient and effective for the needs of the Automatic Entry Exit System.
  4. **Cost-Effective**: DynamoDB is a cost-effective database, as you only pay for the amount of storage and throughput that you use. It is also serverless, which means that you don't need to manage any infrastructure or worry about scaling.
  5. **Security**: DynamoDB provides robust security features, including encryption of data at rest and in transit, fine-grained access control, and integration with other AWS security services like AWS Identity and Access Management (IAM).

- **HTML, CSS, and JavaScript**

  The frontend of the Automatic Entry Exit System is built using HTML, CSS, and JavaScript for several reasons:

  1. **Compatibility**: HTML, CSS, and JavaScript are the standard web development technologies that are supported by all modern browsers. This ensures that the frontend of the Automatic Entry Exit System will be accessible to the widest possible audience.
  2. **Flexibility**: HTML, CSS, and JavaScript provide a high degree of flexibility and customization for web development. This allows developers to create unique and engaging user interfaces that meet the specific needs and requirements of the Automatic Entry Exit System.
  3. **Interactivity**: JavaScript is a powerful programming language that allows for interactive and responsive user interfaces. This means that the frontend of the Automatic Entry Exit System can be designed to respond to user input and provide real-time feedback.
  4. **Speed**: HTML, CSS, and JavaScript are lightweight and fast-loading technologies that ensure quick and efficient rendering of web pages. This is especially important for the Automatic Entry Exit System, which requires fast and responsive user interfaces.
  5. **Development Tools**: There are a wide range of development tools available for HTML, CSS, and JavaScript, including libraries, frameworks, and editors. This makes it easy for developers to create, test, and deploy the frontend

- **Bootstrap CSS Framework**

  1. **Responsive Design**: Bootstrap provides a responsive design that ensures that the frontend of the Automatic Entry Exit System looks good and works well on different screen sizes and devices. This is important because users may access the system from a variety of devices, including smartphones, tablets, and desktop computers.

  2. **Consistency**: Bootstrap provides a consistent and standardized design language that ensures that the frontend of the Automatic Entry Exit System looks and behaves consistently across different pages and components. This makes it easier for users to navigate and use the system.

  3. **Customization**: Bootstrap provides a wide range of customization options, including themes, color schemes, and pre-built components. This allows developers to create unique and visually appealing designs for the Automatic Entry Exit System.

- **Git and GitHub**

  1. Git and GitHub are used for version control because they provide a robust and efficient way to manage version control and collaboration for software development projects.

2 Codebase

The link for the github repository contain all the front-end and back-end of the project SMARTEX is at the following link:

S.M.A.R.T.E.X

The whole web system has also been deployed on the following website: https://rikeshsharma.github.io/smartex/index.html

The navigation of the repository is in the readme file inside the repo itself.

3 Completeness

The part of the SRS that have been completed have been listed below:

External Interface Requirements

User Interfaces:

The user interface of the system will include both a physical component and a software component.

The physical component of the user interface includes the access control devices that individuals interact with when entering or exiting the facility. The access control devices will be located at the point of entry and exit and are used to verify the identity of individuals and authorize or deny access.

The software component of the user interface is typically accessed by the system administrator and other authorized users. The software provides a dashboard or control panel that allows users to manage the system settings, such as creating and managing user accounts, setting access levels and permissions, and generating reports.

One interface will be for student for filling the data automatically or physically (in case scanner doesn't work). The UI will have to be as user-friendly as possible by providing descriptive labels, icons, and step-by-step directions for doing common activities.

From the above the software part has been completed and the manual entry part has been completed too.

Hardware Interfaces :

For Hardware interface for backend there will be computer for admin, so that person can do his job and for student we will have scanner and computer both for registry of that gate.

For accessing the admin page by the authorized users, the hardware required will be internet enabled desktop.

A barcode scanner or a mobile device may be used for scanning the barcode on student's ID.

At the backend a database is needed to store the information.

The above part is being worked upon but is giving some errors and will be working soon.

Software Interfaces:

We aim to include the following elements in our software interface.

An administrative page to access different sections of the system, create manage user accounts, setup different access levels.

A minimal end-user interface for the entering an exit with ideally just scanning the student's ID card.

All the above software requirements are complete.

Functional Requirements

Authentication: The system must be able to authenticate the identity of individuals usingtheir student's ID card.

A navigation menu: Allows users to access different sections of the system, such asScan Page, Manual Entry, Daily Statistics, Logout.

An Admin Dashboard: Provide an overview of Daily Statistics status, changing passwords or adding new admin or guard.

Guard management by Admin: Allows admin to create and manage guard and admin accounts

Scanning/Manual entry: Allows guard/admin to help facilitate automatic entry/exit for student/non-students.

Status Check: Guard/Admin should be able to see the status of a student/non-student if he/she is inside or outside the campus.

Daily Statistics: Allows admin to get info on entry and exit daily statistics

Settings: Allows admin to change admin/guard login credentials or add new admin/guard.

Scanning screen: Provide the end-user to ideally scan and enter or exit with minimumother interaction.

Time Recording: The system should be able to record time of entry and exit just after scanning i-card.

Non-Student Registration: People who are not students or don't already have their data in the database should be able to register for the first time so that their data can be stored and their entry/exit can be easier.

History: Admin/Guard must be able to know complete information about entry/exit of anystudent by putting his details like roll number or user-id.

Editing Records: Admin/Guard should be able to edit or delete any entry from the history table in case of wrong information.

Logout: Admin/Guard should be able to logout of the system.

All the above functional requirements are complete.

Use Case Model:

All the use cases are complete and working, including the new ones that were added in the final version of SRS

Other Non-Functional Requirements:

Performance Requirements

Correctness: The system must be able to process many transactions and users in real-time, with minimal delays and errors.

Response time: The system should work with a few milliseconds of response-time and there should be not much variation in duration of authenticating multiple authentications subsequentially.

Reliability: The system must be reliable and have minimal downtime.

Safety and Security Requirements

Security: The system must be secure and protect against unauthorized access, data breaches, and other security threats.system should not be vulnerable to sql injection attacks.

Safety: Use static code analysis tools. Use of popular and well mainatained libraries and frameworks. All data should be encrypted in transit and at rest.

Software Quality Attributes

Usability: The system must be easy to use and navigate for both end-users and administrators. The system will be user-friendly means any user(new or infrequent) will be able to use product very easily and new users will be able to learn all the functions of product within few minutes. 4.3.2

Maintainability: The system must be easy to maintain and upgrade, with minimal disruption to operations.

Even after the completion of this system, if project member wants some required changes in the system then they can do. They can add some new additional features at any time.

To avoid problems in future we have to achieve, writing efficient and bug-free code with proper tests to cover all the cases or by improving the quality attributes such as high availability, maintainability and reliability

All the above non-functional requirements have been achieved and have been improved, although there is always space for more improvements.

For future versions:

Some new features can be added too to make the system much better:

A mobile application for students or non-student staff can be made which will keep them informed of their entry/exit status.

Late entry email prompts for students can be implemented so that they can be aware of their time outside the campus.