SOFTWARE ENGINEERING

SOFTWARE DESIGN SPECIFICATION

Visitor Management System

GROUP MEMBERS

Raghav Agarwal	IIT2020047
Akanksha Kashyap	IIT2020076
Ekagra Sinha	IIT2020070
Bipul Kumar	IIT2020086

TABLE OF CONTENT

The Software Design Specification

1. Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behavior models, and other supporting requirement information.

1.1 Purpose of this document

This document will define the design of the one runway simulator. It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

1.2 Scope of the development project

We describe what features are in the scope of the software and what are not in the scope of the software to be developed.

In Scope:

a. Application for the visitor management system of IIIT Allahabad.

Out of Scope:

a. Management of visitors in real time.
b. There is direct communication between IIIT Authorities and the system.

1.3 Definitions, acronyms, and abbreviations

IEEE: Institute of Electrical and Electronics Engineers
SDS: Software Design Specification
VMS: Visitor Management System

1.4 References

1.4.1 R. S. Pressman, Software Engineering: A Practitioner's Approach, 5th Ed, McGraw-Hill, 2001.

1.4.2 IEEE SDS template

1.5 Overview of document

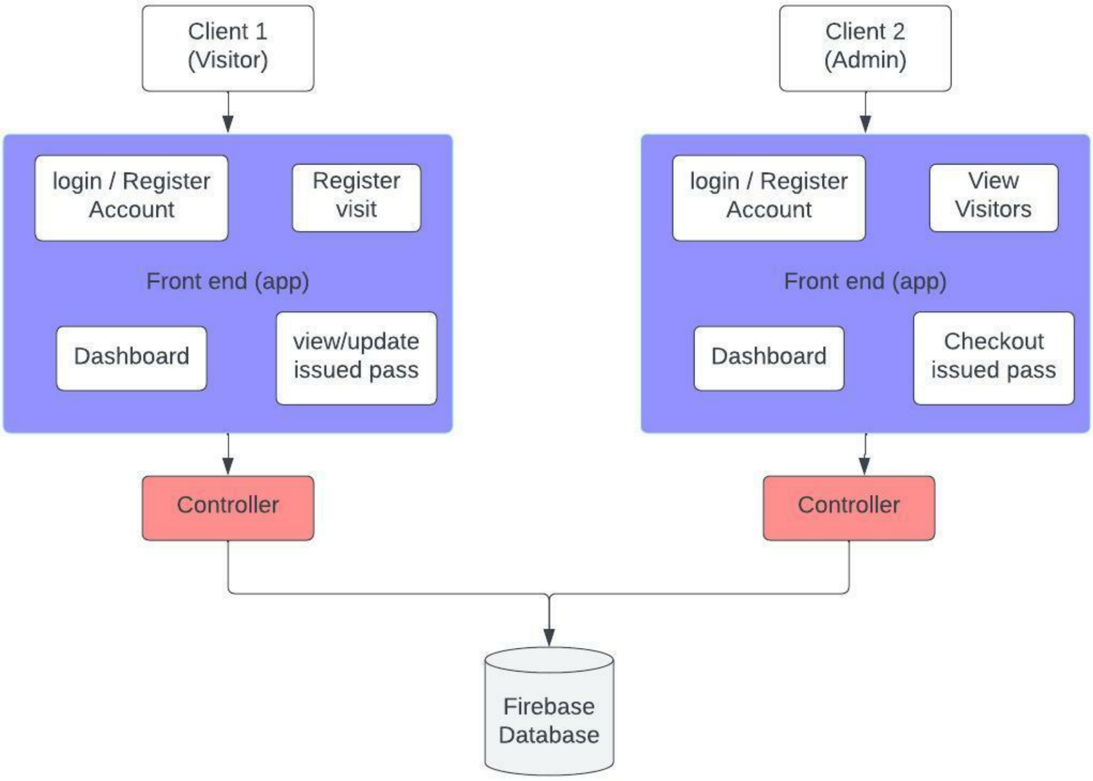
This SDS is divided into seven sections with various sub-sections. The sections of the Software Design Document are:

1. **Introduction**: describes the document, purpose, scope of development project definitions and abbreviations used in the document.
2. **Conceptual Architecture/Architecture Diagram:** describes the overview of components, modules, structure and relationships and user interface issues.
3. **Logical Architecture:** describes Logical Architecture Description and Components.

4. **Execution Architecture:** defines the runtime environment, processes, deployment view.
5. **Design Decisions and Trade-offs:** describes the decisions taken along with the reason as to why they were chosen over other alternatives.
6. **Pseudocode for components:** describes pseudocode, as the name indicates.
7. **Appendices:** describes subsidiary matter if any.

2. Conceptual Architecture/Architecture Diagram

Architecture Diagram:



2.1 Overview of modules / components

FrontEnd -

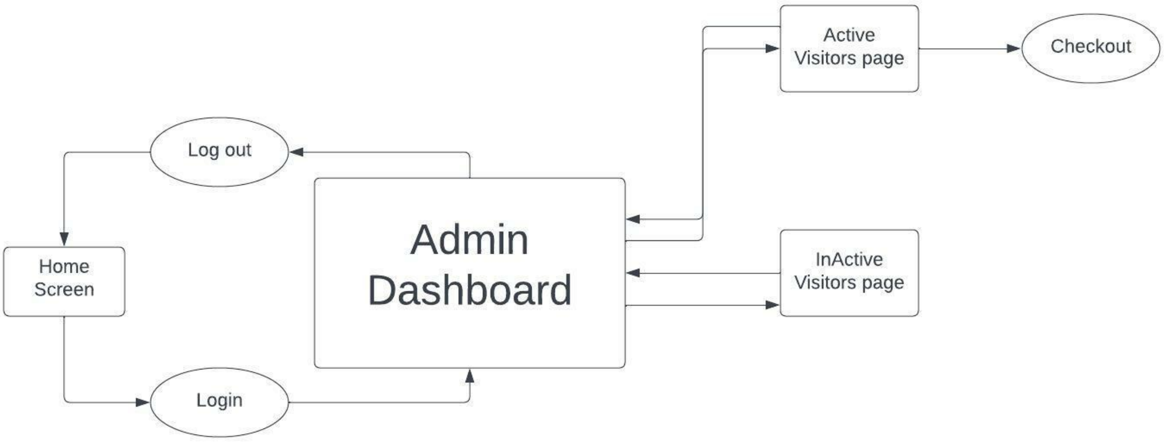
- Dart pages, containing both GUI logic and controller logic to communicate with database

Backend-

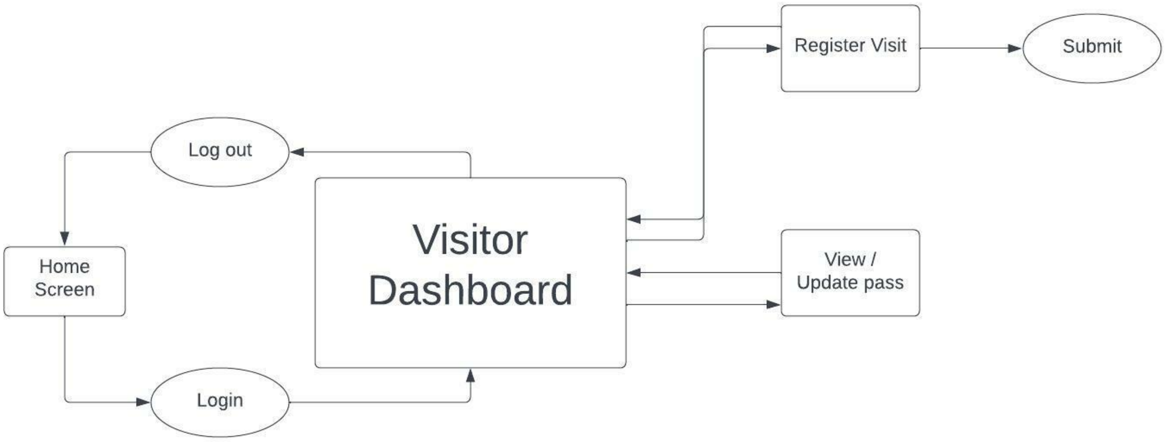
- Firebase

2.2 Structure and relationships

2.2.1 Admin's Side



2.2.2 Visitor's Side



NOTE:
The boxes represent individual screens.
The circles represent actions that do not have screens.
The arrows represent navigation between screens.

2.3 User interface issues

This section will address User Interface issues as they apply to the following hypothetical users of the Visitor Management System.

- User A is a 20-year-old female, a student of IIITA, IIIrd year, who is fairly comfortable with technology. She is proficient with using most common computer applications.

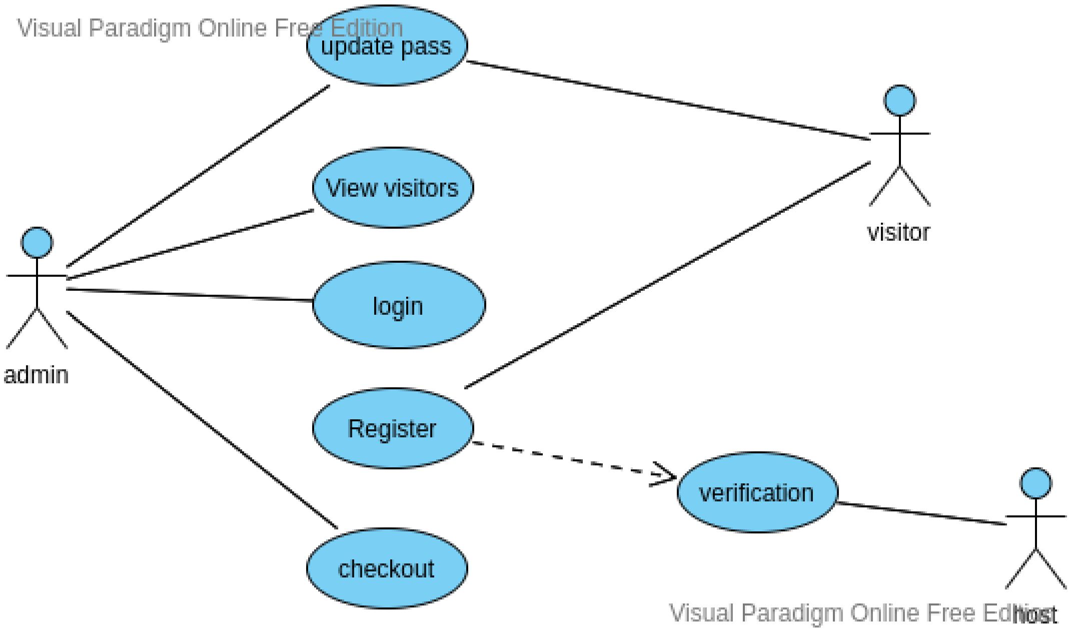
  Since User A is familiar with android applications, Admin will use common user interface conventions. For example, links between screens will use ordinary, easy to understand descriptions such as "Login", "Home", Register User" etc. To maintain consistency, any other links will also appear in the bottom half of the screen.

- User B is a 35-year-old male, A staff member of Security Department IIITA. He is comfortable using android applications, he has never used any portal which helps in managing visitors. He may or may not acknowledge if this android application is really helpful.

  Since User B is not fond of such a technology oriented, it is imperative that he be given clear on-screen directions. Consequently, the user interface has been tried to be such that it grabs the attention of the user. It has been taken care that users should not consider this application as a burden /difficult to use. On-clicks task implementation has been done in VMS, so that the admin (User B, in this case), has an ease in communication with students. Color combination has been chosen that allows the user to read all of the text on the screen in direct sunlight. Text size is reasonably larger and, therefore, more readable.

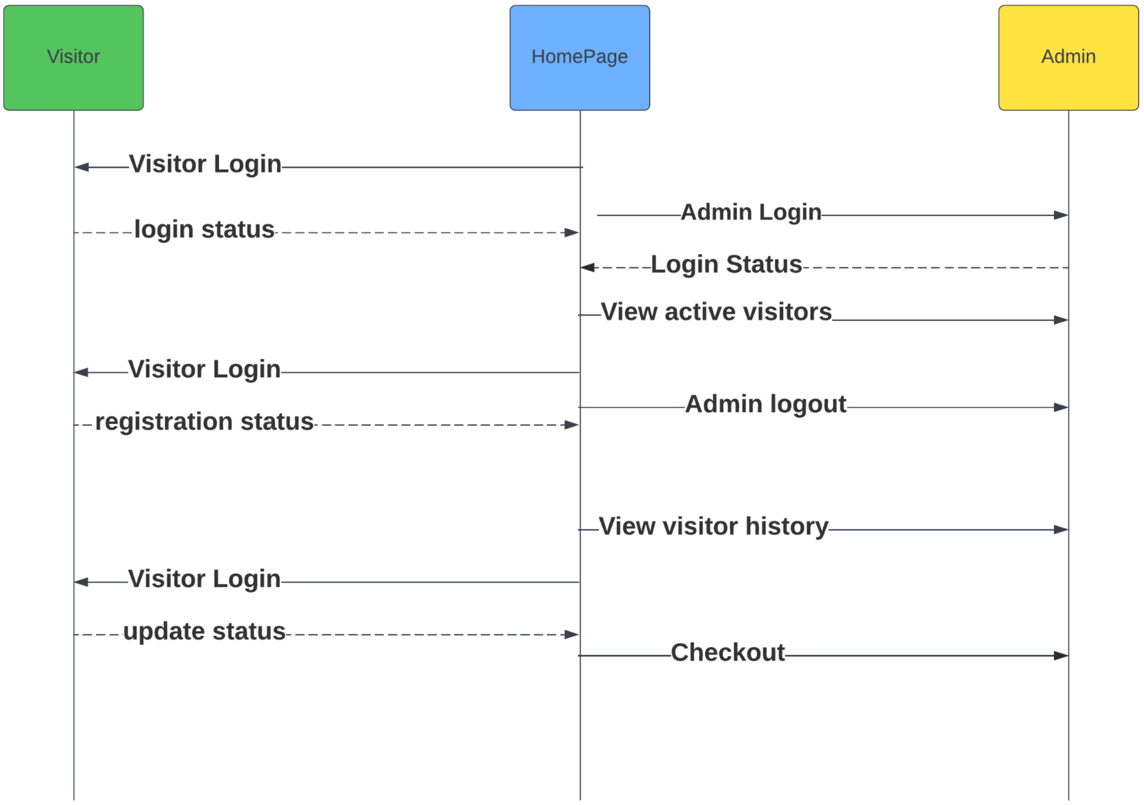
- User C is a 29-year-old female, A faculty member of IIITA. She is well versed with using android apps and therefore, VMS would be easily operated by her.

2. 4 Use case diagram



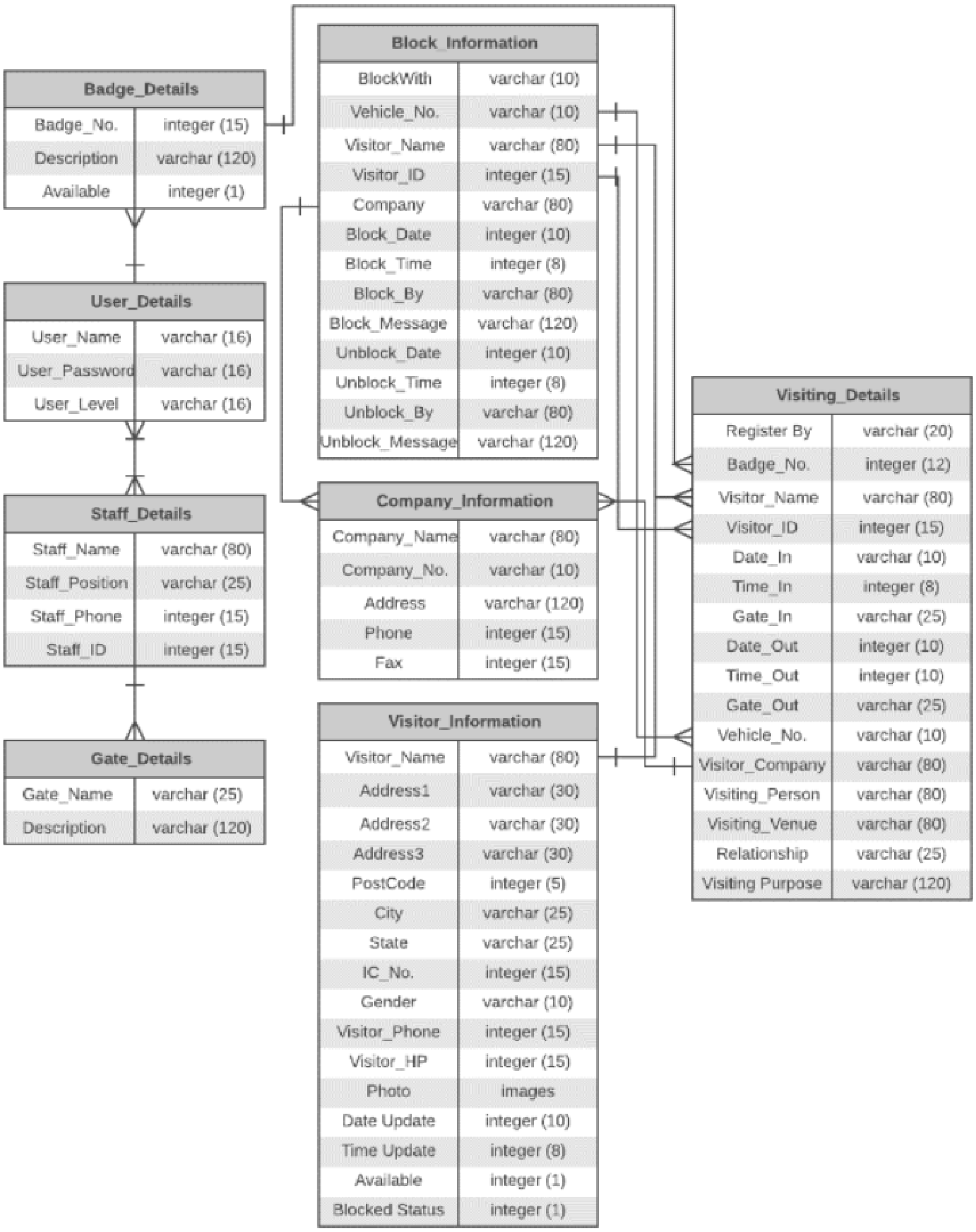
3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram) Class Diagram:

Sequence Diagrams:



3.1 Logical Architecture Description

3.1.1 Class Diagram:



Badge_Details	
Badge_No.	integer (15)
Description	varchar (120)
Available	integer (1)

User_Details	
User_Name	varchar (16)
User_Password	varchar (16)
User_Level	varchar (16)

Staff_Details	
Staff_Name	varchar (80)
Staff_Position	varchar (25)
Staff_Phone	integer (15)
Staff_ID	integer (15)

Gate_Details	
Gate_Name	varchar (25)
Description	varchar (120)

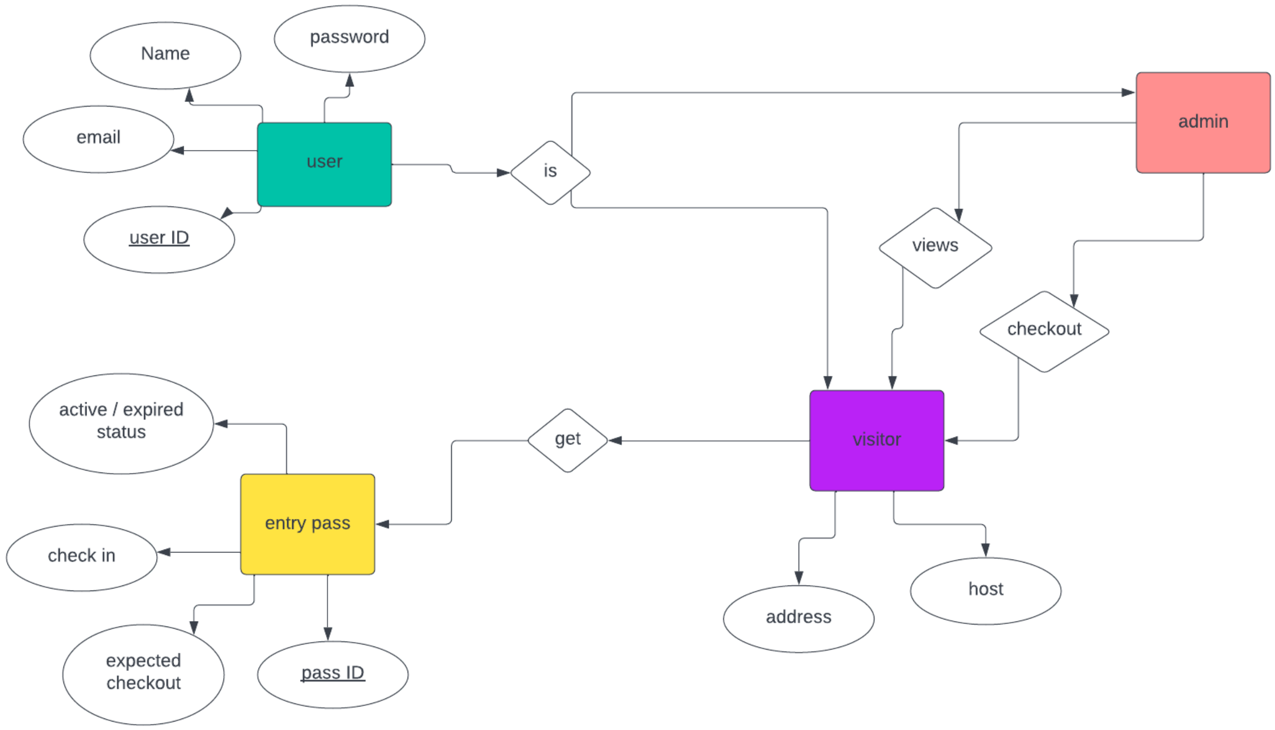
Block_Information	
BlockWith	varchar (10)
Vehicle_No.	varchar (10)
Visitor_Name	varchar (80)
Visitor_ID	integer (15)
Company	varchar (80)
Block_Date	integer (10)
Block_Time	integer (8)
Block_By	varchar (80)
Block_Message	varchar (120)
Unblock_Date	integer (10)
Unblock_Time	integer (8)
Unblock_By	varchar (80)
Unblock_Message	varchar (120)

Company_Information	
Company_Name	varchar (80)
Company_No.	varchar (10)
Address	varchar (120)
Phone	integer (15)
Fax	integer (15)

Visitor_Information	
Visitor_Name	varchar (80)
Address1	varchar (30)
Address2	varchar (30)
Address3	varchar (30)
PostCode	integer (5)
City	varchar (25)
State	varchar (25)
IC_No.	integer (15)
Gender	varchar (10)
Visitor_Phone	integer (15)
Visitor_HP	integer (15)
Photo	images
Date Update	integer (10)
Time Update	integer (8)
Available	integer (1)
Blocked Status	integer (1)

Visiting_Details	
Register By	varchar (20)
Badge_No.	integer (12)
Visitor_Name	varchar (80)
Visitor_ID	integer (15)
Date_In	varchar (10)
Time_In	integer (8)
Gate_In	varchar (25)
Date_Out	integer (10)
Time_Out	integer (10)
Gate_Out	varchar (25)
Vehicle_No.	varchar (10)
Visitor_Company	varchar (80)
Visiting_Person	varchar (80)
Visiting_Venue	varchar (80)
Relationship	varchar (25)
Visiting Purpose	varchar (120)

Entity relationship diagram



3.1.2 Sequence Diagram:

Arrow line signifies there is a send message taking place. Response is being shown by dotted arrows.

3.1.2.1 Register page: Visitor puts data on registor page regarding visitor details, which later can be viewed by admins on their view visitors page.
3.1.2.2 View Visitor page: Admin can view details of all visitors to IIITA campus.

3.1.2.3 Home: Has buttons to Login pages for admin and register page for visitor

3.1.2.4 Student Update Visitor pass: Vistor can update their details.

3.1.2.5 View Visitor pass: Visitors can view the visitor pass issued to them.

3.1.3.6 Admin checkout: Allows admin to checkout a visitor through their pass id

3.1.3.7 Login Admin: Allows user to enter credentials, which are being checked for authentication in the back-end. After being authenticated successfully, it lands up on Admin Dashboard.

3.1.3.8 Admin Dashboard: Has buttons to view passes and checkout visitors

4. Design tradeoffs

None

5. Pseudo code

Login Pseudo code-

```
Input username password
when submit is clicked, send username password to firebase database (through auth plugin)
if request is successful show admin dashboard
```

Register user pseudo code-

```
Input user details
when submit is clicked, send data to firebase.
firebase sends confirmation emails. And visitor pass on verification.
```

view visitors pseudo code-

```
Post a get request to firebase, using admin credentials.
display the data in list format.
```