Attendance Management System using Face Recognition

Developed by

Shashank Kumar Trivedi

https://github.com/strivedi4u/amsProject

https://amsproject.azurewebsites.net

ANTI PLAGIARISM DECLARATION

1. I know that plagiarism means taking and using the ideas, writings, works or inventions of another as if they were one's own. I know that plagiarism not only includes verbatim copying, but also the extensive use of another person's ideas without proper acknowledgement (which includes the proper use of quotation marks). I know that plagiarism covers this sort of use material found in textual sources and from the Internet.

2. I acknowledge and understand that plagiarism is wrong.

3. I understand that my project work must be accurately referenced. I have followed the rules and conventions concerning referencing, citation and the use of quotations as set out in the Department Guide.

4. This assignment is my own work, or my group's own unique group assignment. I acknowledge that copying someone else's assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism.

5. I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.

Student Name

Shashank Kumar Trivedi

TABLE OF CONTENTS

Topics	Page No.
1.Aim and Objectives …….	8
2. Abstract ……..	9
3. Introduction ……	10
4. Software Requirement Specification ……..	12
5. System Planning & SDLC ……..	20
6. Constraints of use ……..	21
7. Technology used ……..	22
8. Feasibility Study ……..	29
9. Entity relationship diagram(ERD) ……	30
10. Data flow diagram (DFD) ……..	31
11. Gantt Chart ……..	33
12. Database designed ……	34
13. Unified Modeling Language (UML) Diagram ……..	37
14. Screenshots of Application ……..	40
15. Screenshots of Tools ……..	46
16. Source Codes ……	49
17. Future Scope of Work ……..	64
16. Conclusion ……..	65
17. Bibliography ……..	66

1.) AIM & OBJECTIVES

- The very first thing keeping in our mind we are developing this project is to gather all the college members, starting from administrative level to the basic clerical section into a single frame.
- To automate the manual work of taking attendance and replace the mythology of keeping the record of attendance in Hard copy to maintain it in soft-copy via. An software application
- Keep the security of application as first preference and identify the authorized person via. Software without any human support to machine.
- Shortening the time spend on taking attendance one by one in a sequence by taking attendance in bulk by recognizing the faces of authorized users.

2.) ABSTRACT

The purpose of Attendance Management System is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy assessing and manipulation of the same. The required software and hardware are easily available and easy to work with. AMS PORTAL, as described above, can lead to error free, secure, reliable and fast networking system. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

3.) INTRODUCTION

The “AMS PORTAL” has been developed to override the problems prevailing in the manual attendance work. This software is supported to eliminate the manual attendance work and in some cases reduce the hardships faced by this attendance work. Moreover this system is effective manner of attendance.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it provides it is user-friendly. AMS PORTAL, as described above, can lead to error free, secure, reliable and fast networking system. It can assist the student to concentrate on their attendance rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources.

Every institution, whether big or small, has challenges to overcome andmanaging the information’s of friends, users, shares, videos, photos. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future goals. These systems will ultimately allow you to better manage resources.

The project includes two main sections:

- Student:
    - In this system student is only priority for accessing the “Attendance Management System using Face Recognition” application.
    - Here student can create his/her own profile along with all the details and his/her profile photo.
    - Every student have individual & unique log in credential to go through the portal.
    - This website provides separate access for different sections of about, contact us, help, documentation & services etc.
    - Student can view their attendance anywhere and anytime.
    - Student can also edit their profile.

- Admin:
    - Admin can modify student details and delete the payment & attendance details.
    - Admin can add students and provide them student id and password without OTP confirmation & payment.
    - Admin can view the all student attendance.

In this system developer is a part for developing the application. Every developer has the permission to access all data of database of AMS PORTAL and also the permission for create, delete, update and many other features for betterment of security and future upgradation.

4.) SOFTWARE REQUIREMENT SPECIFICATION (SRS)

4.1 Introduction

a) Purpose

The main objective of this document is to illustrate the requirements of the project ***"Attendance Management System using Face Recognization"***. The document gives the detailed description of both functional and non-functional requirements proposed by the client. The purpose of this project is to build a connectivity between students and admin to reduce the manual attendance work for conveying any information regarding to the college. It tracks all the details about the Student Attendance & Personal Details etc.

b) Scope of Document Project

Attendance Management System using Face Recognition is a system that can be easily use by various users (viz. Academic, Administration, Training & Placement, Library, Finance, Examination sections).

This project is specially designed for the use of students.

- Student can view their attendance & profile with a click of mouse.
- Student can ask for any query with the help of Chat-Bot.
- Students can mark their attendance using the face without fingerprint and manual work.
- Admin view all Students Attendance, Payment Details & Personal Details of the student.
- Admin add student without payment because when student want registration on the AMS Portal the student pay some money but admin section payment not be considered.

It is especially useful for any educational institute where modifications in the content can be done easily according to requirements.

The project can be easily implemented under various situations. We can add new features as and when we require, making reusability possible as there is flexibility in all the modules.
The language used for developing the project is JAVA as it quite advantageous than other languages in terms of performance, tools available, cross platform compatibility and development process.

c) Definitions, Acronyms and Abbreviations

NOSQL -> Non Structured Query Language

ERD -> Entity Relationship Diagram

UML -> Unified Modeling Language

SRS -> Software Requirement Specification

SDLC -> Software Development Life Cycle

DFD -> Data Flow Diagram

4.2 Overall Descriptions

a) User Classes and Characteristics

The system provides different types of services based on the type of users. Here users are accessing the website after create his/her own account. When a user open his/her account then shows his/her attends and also show the profile. Users can also edit their profile along with all the details and his/her profile photo.

The features that are available to the Users are:

- Create his/her own profile along with all the details and his/her profile photo.
- Every users have individual & unique log in credential to go through theportal.
- After login Students can edit personal details & view the attendance.
- Admin have the Face Recognition access to mark the student attendance.
- Admin can view the student attendance & student personal details but not see the password because password saved on the database in encrypted way.
- Admin have the permission to add the student without any pay cost.
- Admin can view payment details, delete & update the student details.
- Admin & Student if any query then we go to the help section & Contact Us section to contact with the development team.

- This website provides separate access for different sections of college (viz. Academic, Administration, Training & Placement, Library, Finance, Examination sections).

b) Operating Environment

This application will be operating in Web pages through browser. The Attendance Management System using Face Recognition is a web application and can be operated through browser in web pages. The only requirement to use this website would be the internet connection and a device which can access high speed internet.

c) Assumptions and Dependencies

The assumptions are:

- The coding should be error free.
- The application should be user-friendly so that it is easy to use for theusers.
- Valid information of every users must be stored in database that is accessible by the website.
- The system should provide more storage capacity and provide fast accessto the database.
- The system should provide search facility and support quick transactions.
- Attendance Management System using Face Recognition is running 24 hours a day.
- Users may access from any browser that has Internet browsing capabilities and an Internet connection.
- Admin must have their correct usernames and passwords to enter into their AMS POERTAL accounts and do actions.

- The dependencies are:
    - The specific hardware and software due to which the product will be run.
    - On the basis of listing requirements and specification the project will bedeveloped and run.
    - The end users should have proper understanding of the product.
    - The information of all the users must be stored in a specific database that is easily accessible.
    - Any update regarding the student profile, attendance, face Recognition & other updation is to be recorded to the database and the data entered should be correct.

d) Requirement

Software Configuration:

This software package is developed using HTML, CSS, Bootstrap, RecatJS as front end, JavaScript, JQuery as client side validation, Ajax as server side validation, JAVA & NodeJs as business logic, MongoDB as to store the database, TanserFlow as Face Recognition, Razorpay as Payment Intrigation, JUnit as Java Testing, Postman as API Testing, SpringBoot as Java Fremwork, GitHub as creating Repository, Azure as deploying project, InteliJIDEA as IDLE for Backend codeing, VScode as Frontend Coding and Apache Tomcat as web server .

Operating System: Mac OS, Linux, Windows 10, Windows 11

Front end: HTML, CSS, Bootstrap

Client side Validation: JavaScript, jQuery

Server side Validation: Ajax

Business Logic: JAVA, NodeJs

Database: Mongo DB

Web Server: Apache Tomacat

Testing Tools: JUnit, Postman

Payment Intrigation: Razorpay

Face Recognition: TanserFlow

IDLE: InteliJIDEA, VSCode

Fremwork: ReactJs, SpringBoot

Deploy: Azure

Hardware Configuration:

Processor: Core i3, 1.5MHz

Hard Disk: 150 GB

RAM: 8GB

Resolution: 480 X 800

e) Data Requirement

The inputs consist of the query to the database and the output consists of the solutions for the query. The output also includes the user receiving the details of their accounts. In this project the inputs will be queries as fired by the users like create an account. Now the output will be visible when the user requests the server to get details of their own account and also accounts of the other members in the form of time, date.

4.3 External User Interface

a) GUI

This application provides good graphical interface for the user and the administrator can operate on the system, performing the request task such as create, update, delete and also view the every details.

- ❖ The user interface must be customizable by the administrator.
- ❖ All the modules provided with the software must fit into this graphical user interface and accomplish to the standard defined.
- ❖ The design should be simple and all the different interfaces should followa standard template.
- ❖ The user interface should be able to interact with the other users.
- ❖ Design of registration page, log in page and also OTP verification page iseasy to understand for every user.

4.4 System Features

This is possible by providing:-

- ❖ User accessibility available if he/she is a part of a particular collegecampus.
- ❖ Every users have individual & unique log in credential to go through the portal.
- ❖ User can register his/her profile along with all the details and his/herprofile photo.
- ❖ This website provides separate access for different sections of college (viz. Academic, Administration, Training & Placement, Library, Finance, Examination sections)
- ❖ Students can view attendance details & personal details including photo.
- ❖ Student can edit the personal details.
- ❖ Admin can view, update & delete the student details, attendance details & payment details.
- ❖ Admin have the permission to the recognizing the face of the students and add the students on the portal without payment.

4.4 Other Non Functional Requirement

a) Performance Requirement

The proposed system that we have developed will be used as the chief performance system within the different college campuses and also for users. Therefore, it is expected that the database would perform functionally all the requirements that are specified by the college and also for the users.

- ❖ The performance of the system should be fast and accurate.
- ❖ AMS PORTAL shall handle expected and non-expected errors in way that prevent loss in information and long downtime period. Thus it should have inbuilt error testing to identifysearch or data check/fetch.
- ❖ The system should be able to handle large amount of data. Thus it should accommodate large number of data entry of a particular college campus without any fault.

b) Satisfy Requirement

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required take the database backup.

c) Security Requirement

- System will use secured database.
- AMS PORTAL Application is very secure because the Java Programming language is used on backend that's never hack.
- System is very reliable and accuracy is very fast because the MongoDB and ReactJs is used.
- System is anywhere to access with the security because application is deployed on the azure that is more secure the any hosting providers.
- Normal users can just read instruction and use this system but they cannot edit or modify anything except their personal and some otherinformation.
- System will have two type of user and user has access constraints.
- Proper user authentication should be provided.
- No one should be able to hack user's details or any other information.
- There should be separate part for users that no users can access thedatabase and only admin has the rights to update the database.

d) Requirement Attributes

- There may be multiple admins creating the project, and all of them will have the right to create changes to the system. But the users cannot do changes.
- The project should be open source.
- The quality of the database is maintained in such a way so that it can be very user friendly to all the users of the database.
- The user be able to easily access this system from anywhere at any time.

e) Business Rule

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, make a decision, or infer new data from existing data. This includes the rules and regulations that the System users should abide by. This includes the cost of the project and the discount offers provided. The users should avoid illegal rules and protocols. Neither admin nor member should cross the rules and regulations.

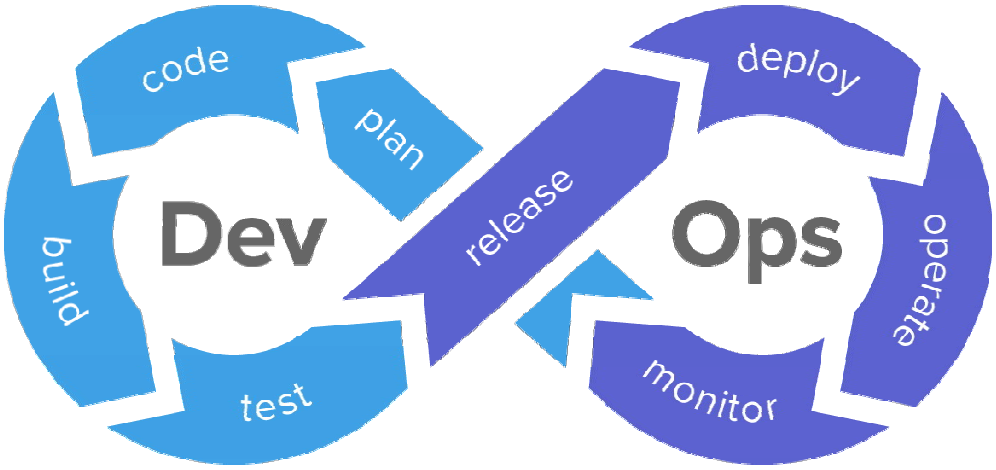
f) User Requirement

The users of the system are members of the “Attendance Management System using face Recognition”. The users are assumed to have basic knowledge of the computers and internet browsing. The administrators of the system should have more knowledge of the internals of the system and is able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to maintain the system. The proper user interface, user manual, online help and the guide to install and maintain the system must be sufficient to educate the users on how to use the system without any problems.

5).SYSTEM PLANNING & SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

We have chosen "**DevOps** Life Cycle Model" for developing this application, because an DevOps life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

Here is the diagram of the DevOps life cycle model which depicts its working flow.



Few advantages for choosing the SDLC are –

- In DevOps model we are building and improving the product step by step. Hence, we can track the defects at early stages. This avoids the downward flow of the defects.
- Testing and debugging in smaller iteration is easy.

- In DevOps model we can get the reliable user feedback. When presenting sketches and blueprints of the product to user for their feedback, we are effectively asking them to imagine how the product will work.
- Progress can be measured.
- In DevOps model less time is spent on documentation and more time is given for development.
- Risk are identified and resolved during iteration; and each iteration is an easily managed milestone. It supports changing requirements.

6.)CONSTRANTS OF USE

- Admin and Student must remember login id and password.
- User need to have a personal computer or cell phone with internet connection.

7.)TECHNOLOGY USED

7.1 Front-end Design:

7.1.1 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

7.1.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

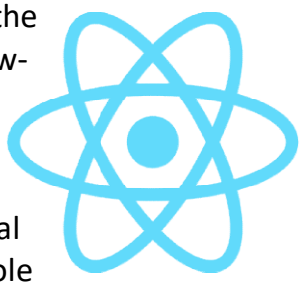
7.1.3 Bootstrap

Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

7.1.4 JavaScript

JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

7.1.5 React JS



React's primary role in an application is to handle the view layer of that application just like the V in a model-view-controller (MVC) pattern by providing the best and most efficient rendering execution. Rather than dealing with the whole user interface as a single unit, React.js encourages developers to separate these complex UIs into individual reusable components that form the building blocks of the whole UI. In doing so, the ReactJS framework combines the speed and efficiency of JavaScript with a more efficient method of manipulating the DOM to render web pages faster and create highly dynamic and responsive web applications.

7.1.6 Node JS

Node.js is an open source, cross-platform runtime environment and library that is used for running web applications outside the client's browser. It is used for server-side programming, and primarily deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services, but was originally designed with real-time, push-based architectures in mind. Every browser has its own version of a JS engine, and node.js is built on Google Chrome's V8 JavaScript engine.

7.2 Back-end Design:

7.2.1 Java Programming

Java is a widely-used programming language for coding web applications. It has been a popular choice among developers for over two decades, with millions of Java applications in use today.

Java is a multi-platform, object-oriented, and network-centric language that can be used as a platform in itself. It is a fast, secure, reliable programming language for coding everything from mobile apps and enterprise software to big data applications and server-side technologies.

7.2.2 SpringBoot

Java Spring Framework (Spring Framework) is a popular, open source, enterprise-level framework for creating standalone, production-grade applications that run on the Java Virtual Machine (JVM). Java Spring Boot (Spring Boot) is a tool that makes developing web application and microservices with Spring Framework faster and easier through three core capabilities:

Auto configuration

An opinionated approach to configuration

The ability to create standalone applications.

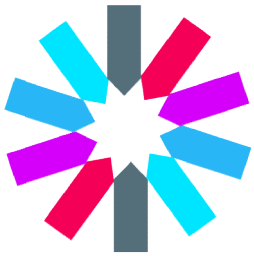
7.2.3 Spring Secuirty

Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements

7.2.4 Apache Maven

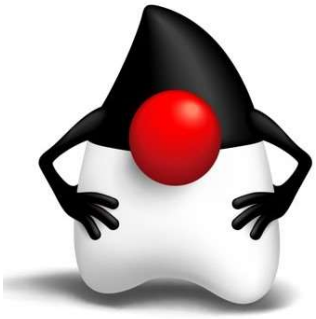
Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process like ANT. But it is too much advanced than ANT. In short terms we can tell maven is a tool that can be used for building and managing any Java-based project. maven make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

7.2.5 JSON Web Token (JWT)



JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.

7.2.6 Java Mail

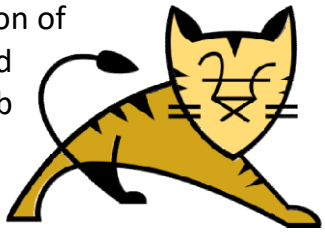


The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API is available as an optional package for use with the Java SE platform and is also included in the Java EE platform.

7.3 Database: Mongo DB

MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in your application code, making data easy to work with. Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data. MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use.

7.4 Web Server: Apache Tomacat



Apache Tomcat is a free and open-source implementation of the Jakarta Servlet, Jakarta Expression Language, and WebSocket technologies. It provides a "pure Java" HTTP web server environment in which Java code can also run. Thus it's a Java web application server, although not a full JEE application server.

7.5 : Face Recognition Library

7.5.1 TanserFlow.js

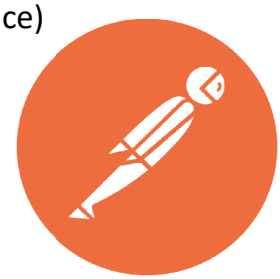
TensorFlow has a lot of machine learning libraries and is well-documented. It provides a few key functions and ways for doing so. TensorFlow is sometimes referred to as a “Google” product. A wide range of machine learning and deep learning algorithms are included. For handwritten digit classification, image recognition, word embedding, and the generation of other sequence models, TensorFlow can train and run deep neural networks.

7.6: Testing Tools

7.6.1 JUnit

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit. JUnit is linked as a JAR at compile-time

7.6.2 Postman



Postman is an API(application programming interface) development tool which helps to build, test and modify APIs. Almost any functionality that could be needed by any developer is encapsulated in this tool.It has the ability to make various types of HTTP requests(GET, POST, PUT, PATCH), saving environments for later use, converting the API to code for various languages(like JavaScript).

7.7: Payment Geteway

7.7.1 Razorpay

Razorpay helps you accept online payments from customers across Desktop, Mobile web, Android & iOS. Additionally by using Razorpay Payment Links, you can collect payments across multiple channels like SMS, Email, Whatsapp, Chatbots & Messenger.

7.8: Chat-Bot

7.8.1 DialogFlow

Dialogflow is a natural language understanding platform used to design and integrate a conversational user interface into mobile apps, web applications, devices, bots, interactive voice response systems and related uses.

7.9: Intregated Development Kit (IDE)

7.9.1 InteliJ IDEA

IntelliJ is one of the most powerful and popular Integrated Development Environments (IDE) for Java. It is developed and maintained by JetBrains and available as community and ultimate edition. This feature rich IDE enables rapid development and helps in improving code quality.

7.9.2 Visual Studio Code

Visual Studio Code is a free, lightweight but powerful source code editor that runs on your desktop and on the web and is available for Windows, macOS, Linux, and Raspberry Pi OS. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other programming languages (such as C++, C#, Java, Python, PHP, and Go), runtimes (such as .NET and Unity), environments (such as Docker and Kubernetes), and clouds (such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform).

7.10: Version Control Tool

7.10.1 Git

Git is a distributed version control system: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

7.10.2 GitHub

GitHub, Inc. is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

7.11: Deploy

7.11.1 Microsoft Azure

Microsoft Azure, often referred to as Azure is a cloud computing platform operated by Microsoft for application management via around the world-distributed data centers. Microsoft Azure has multiple capabilities such as software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools, and frameworks, including both Microsoft-specific and third-party software and systems.

7.11.1 FileZilla

FileZilla is a free and open-source, cross-platform FTP application, consisting of FileZilla Client and FileZilla Server. Clients are available for Windows, Linux, and macOS. Both server and client support FTP and FTPS, while the client can in addition connect to SFTP servers.

8.)FEASIBILITY STUDY

A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as economic, technological and operational. Project managers use feasibility studies to determine potential positive and negative outcomes of a project before investing a considerable amount of time and money into it.

During the stage of our feasibility study, we had to undergo the following steps as described under:

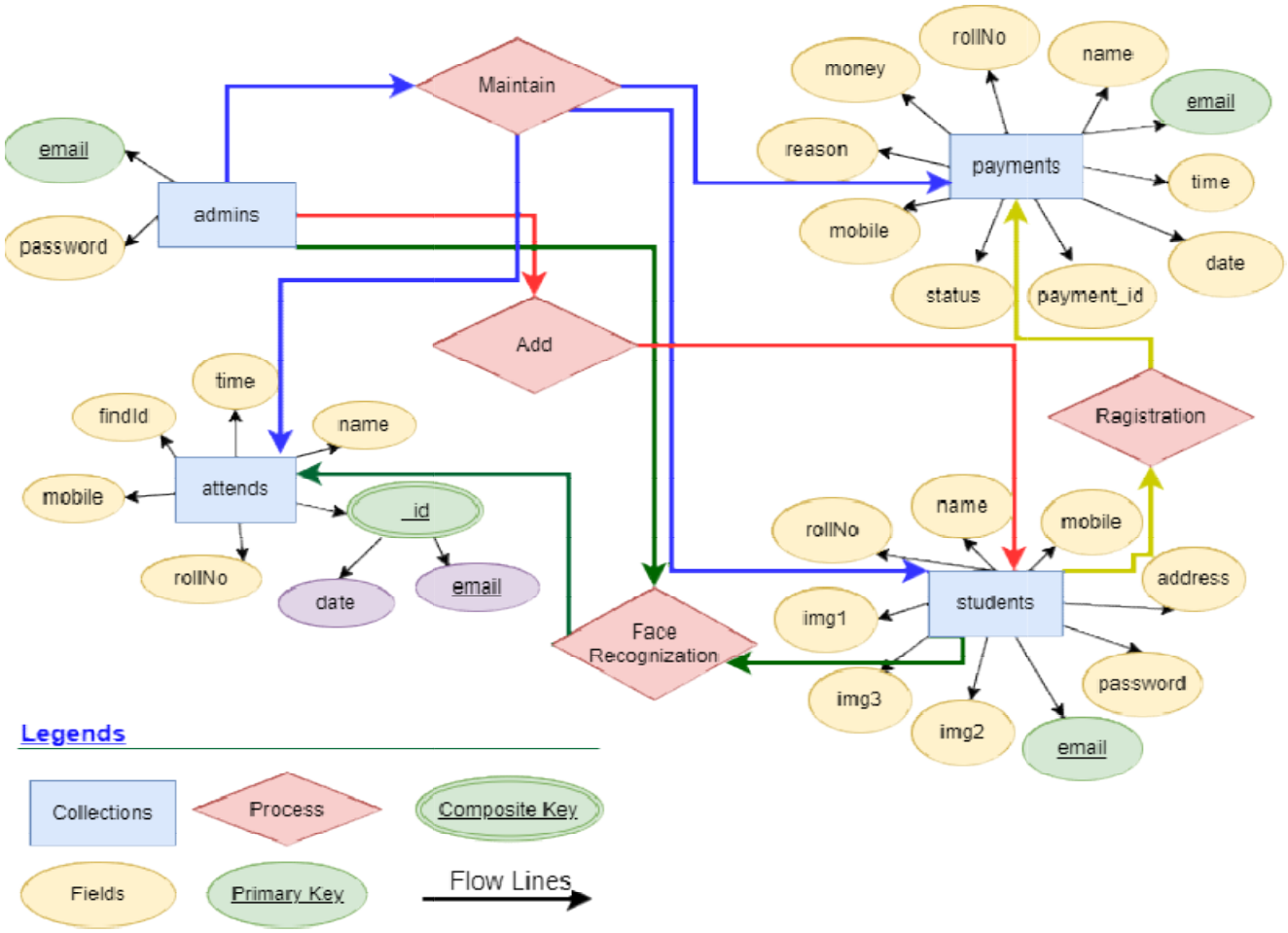
- Identify the origin of data at different levels of the system.
- Identify the expectation of end user from the finished product/system.
- Analyze the drawback(s) of the existing system.

- **Technical feasibility study:** It lays out details on how a good or service will be delivered, which includes transportation, business location, technology needed, materials and labour.

- **Financial feasibility study:** It is a projection of the amount of funding or startup capital needed, what sources of capital can and will be used and what kind of return can be expected on the investment.

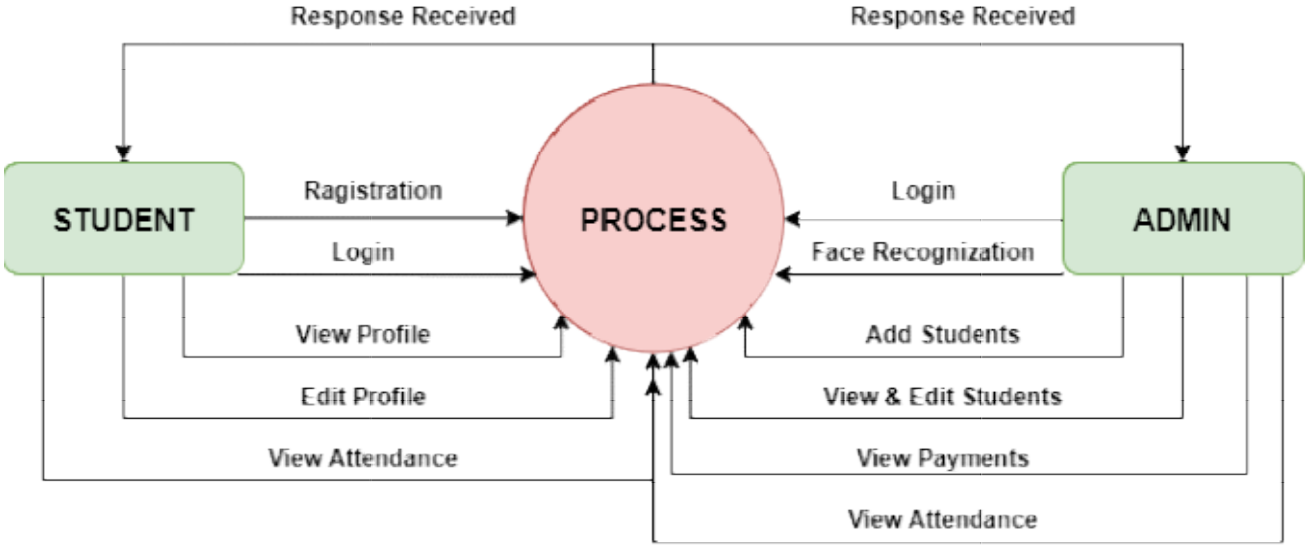
- **Organizational feasibility study:** It is a definition of the corporate and legal structure of the business; this may include information about the founders, their professional background and the skills they possess necessary to get the company off the ground and keep it operational.

9.)ENTITY RELATIONSHIP DIAGRAM(ERD)

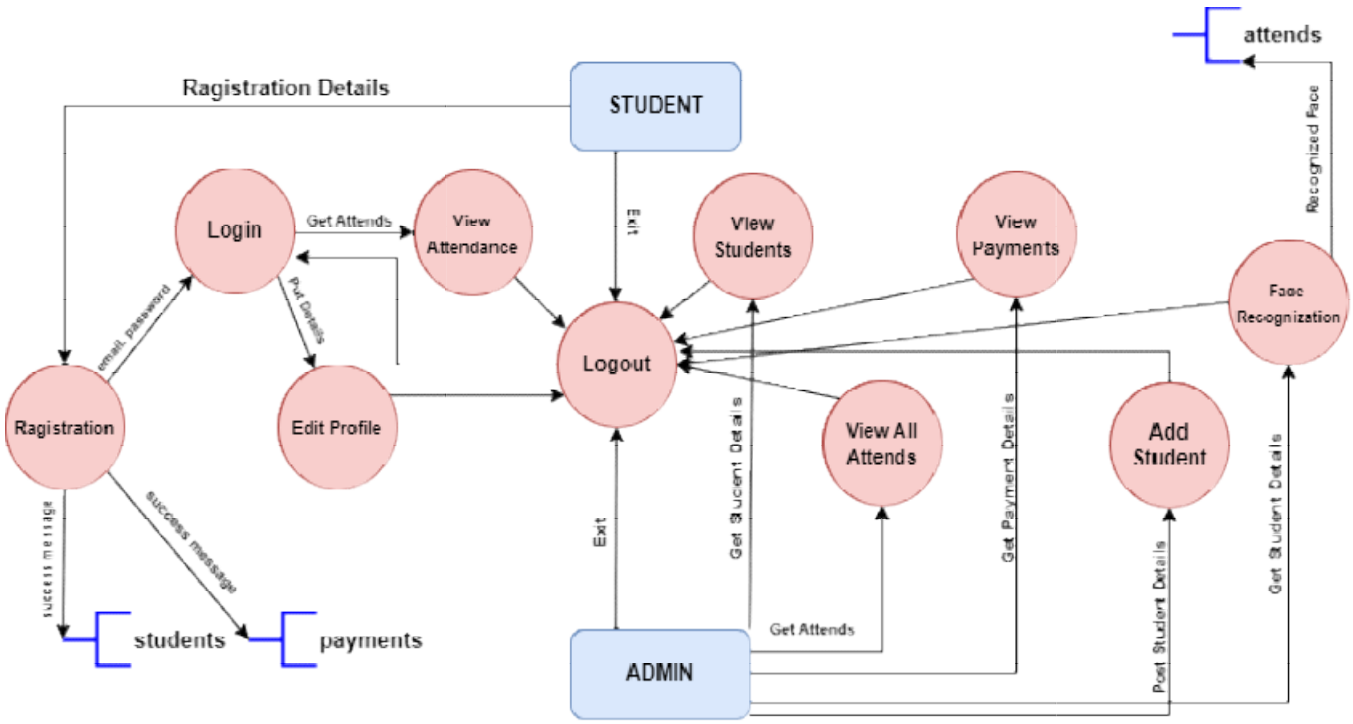


10.)DATA FLOW DIAGRAM(DFD)

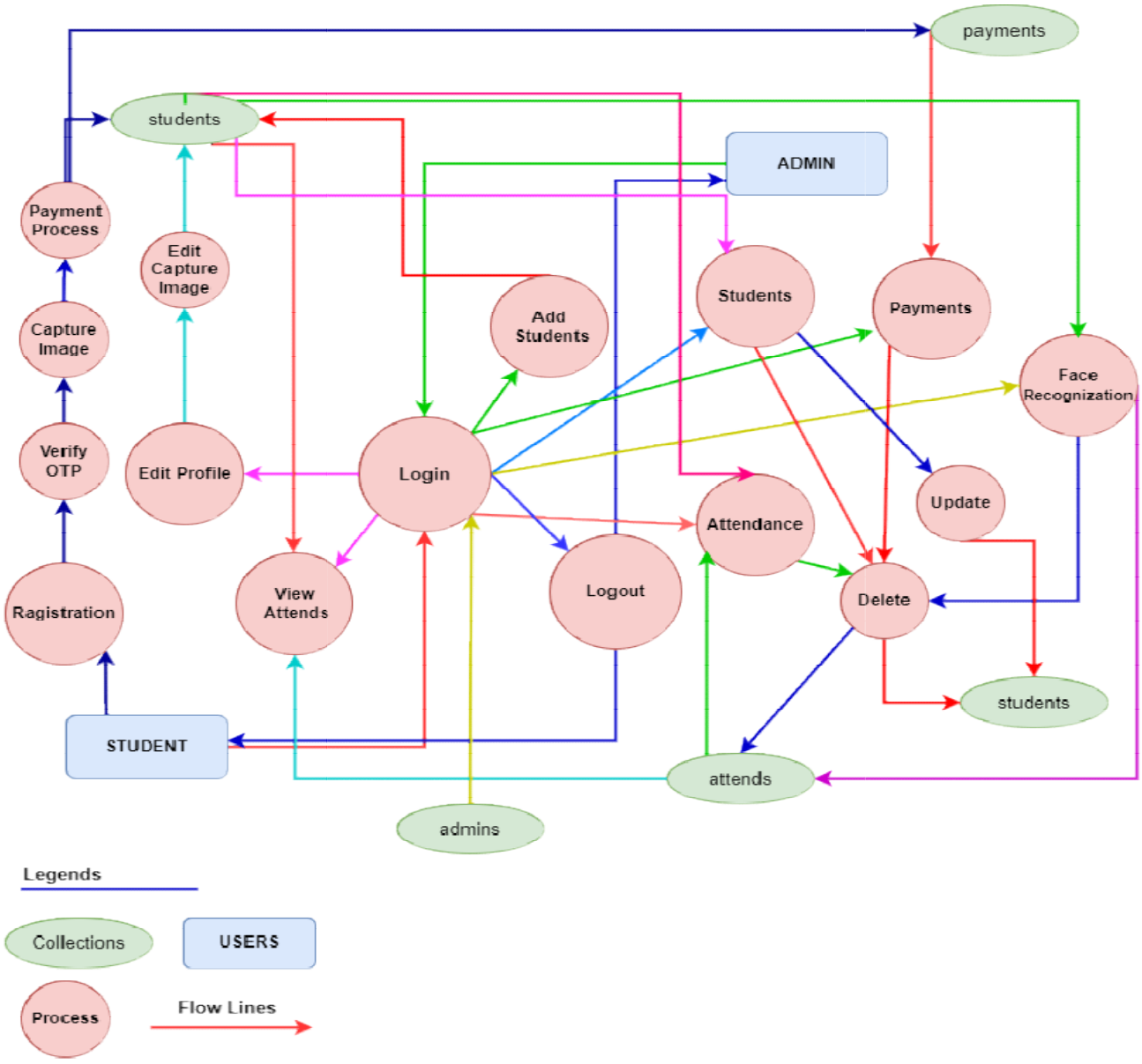
- **level 0**



- **level 1**



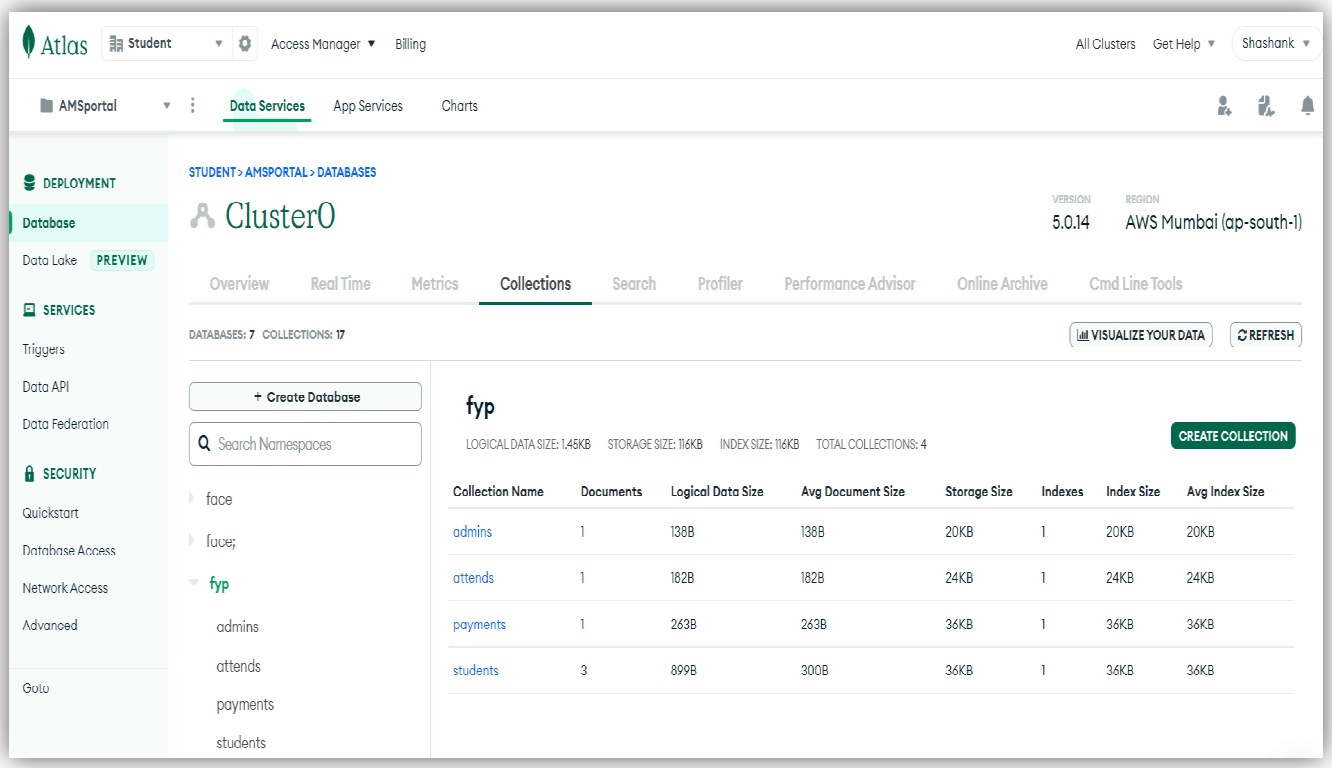
- **level 2**



11.) GANTT CHART

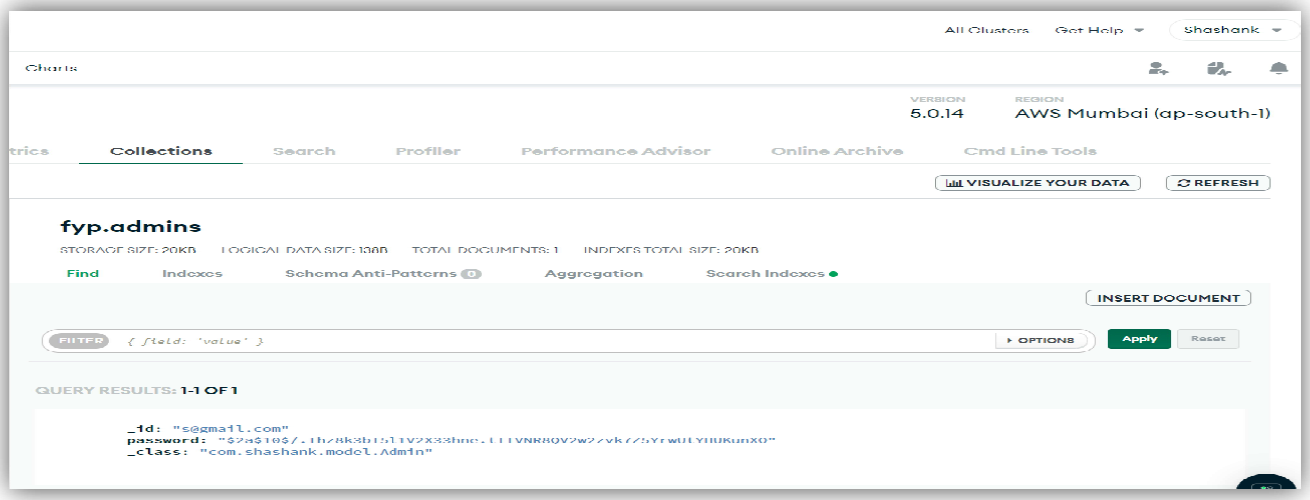
12.) DATABASE DESIGNED

12.1 Database: AMS PORTAL

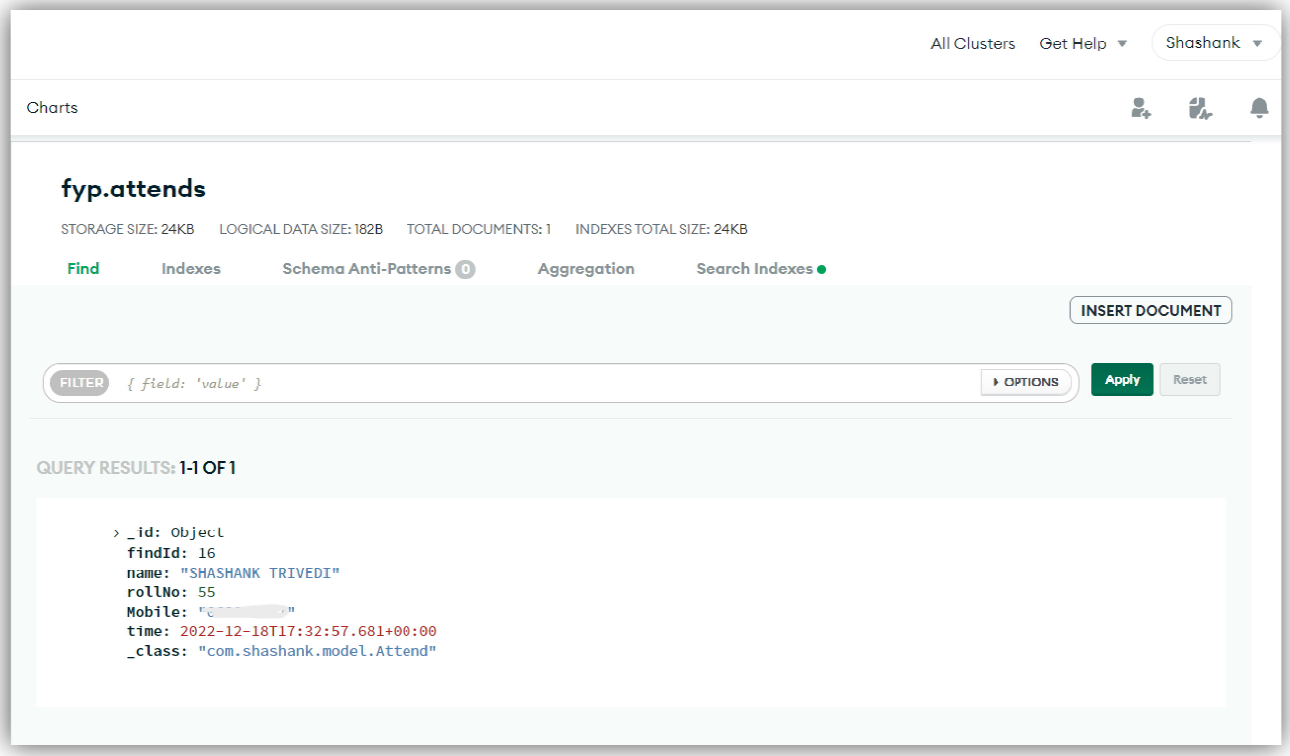


12.2 Database Table:

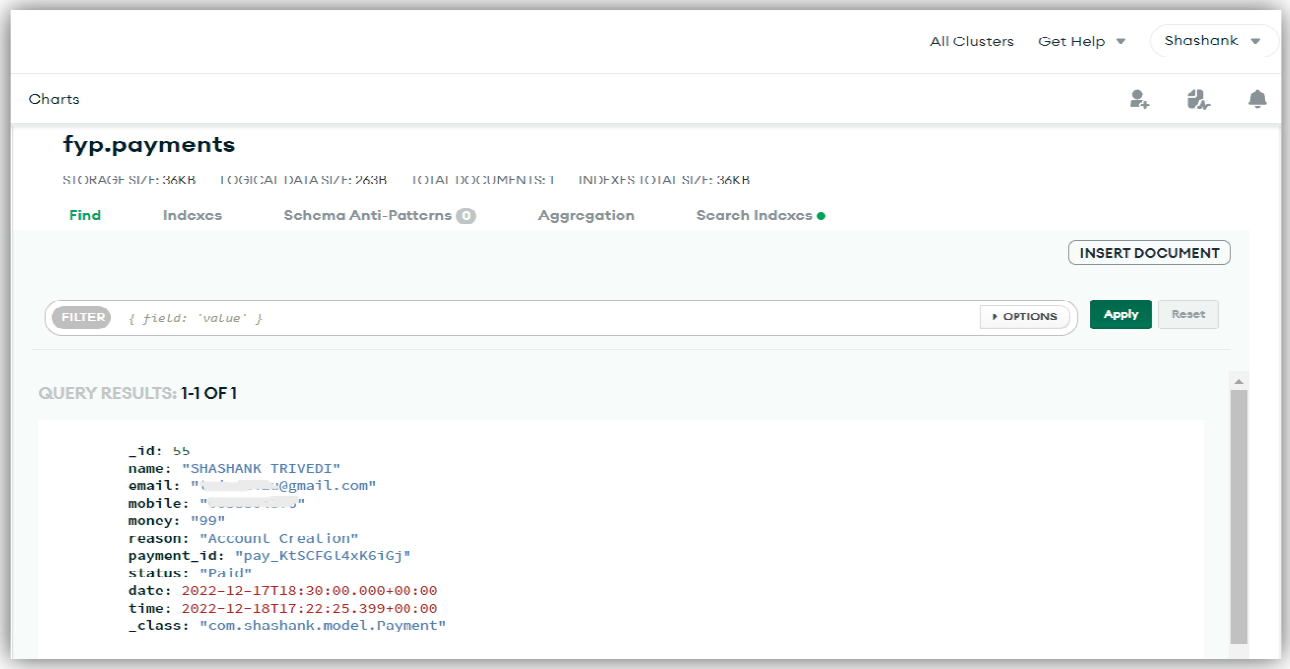
12.2.1 admins



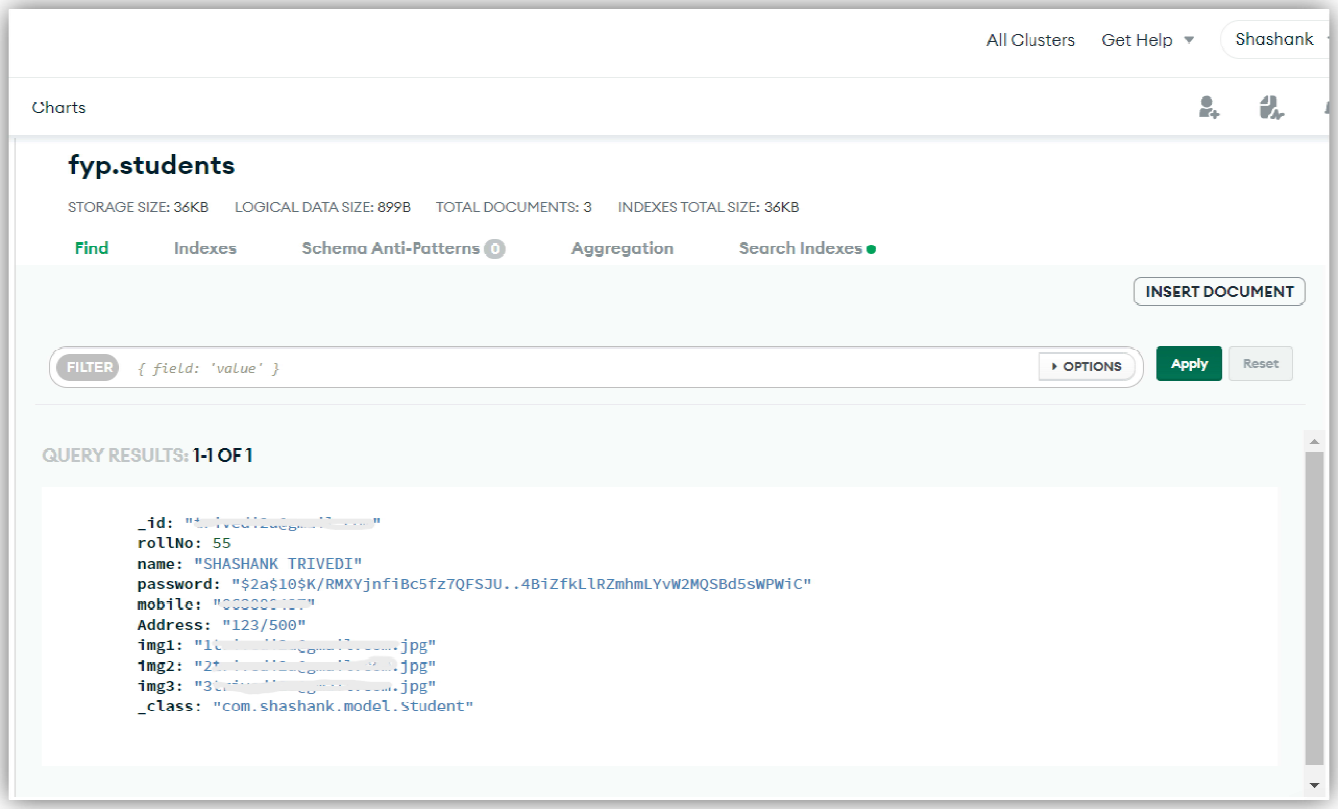
12.2.2 attends



12.2.3 payments



12.2.4 students



13.)UNIFIED MODELING LANGUAGE (UML) DIAGRAM

13.1 Use Case Diagram

- **Welcome Page**

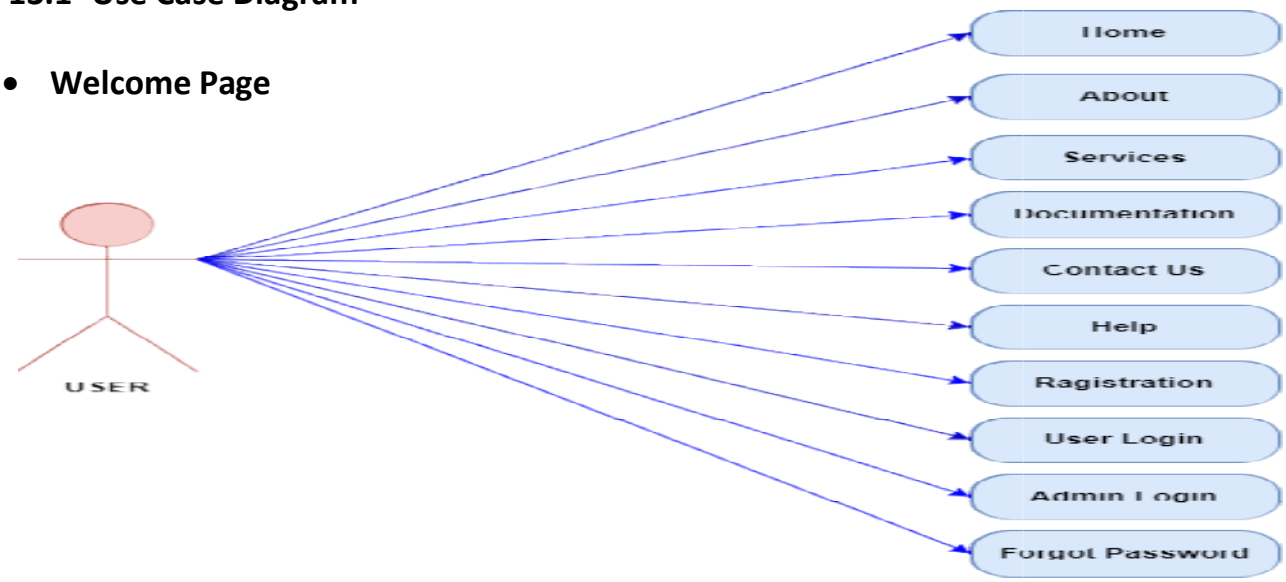


Fig: USE CASE DIAGRAM OF WELCOME

- **After Login Page**

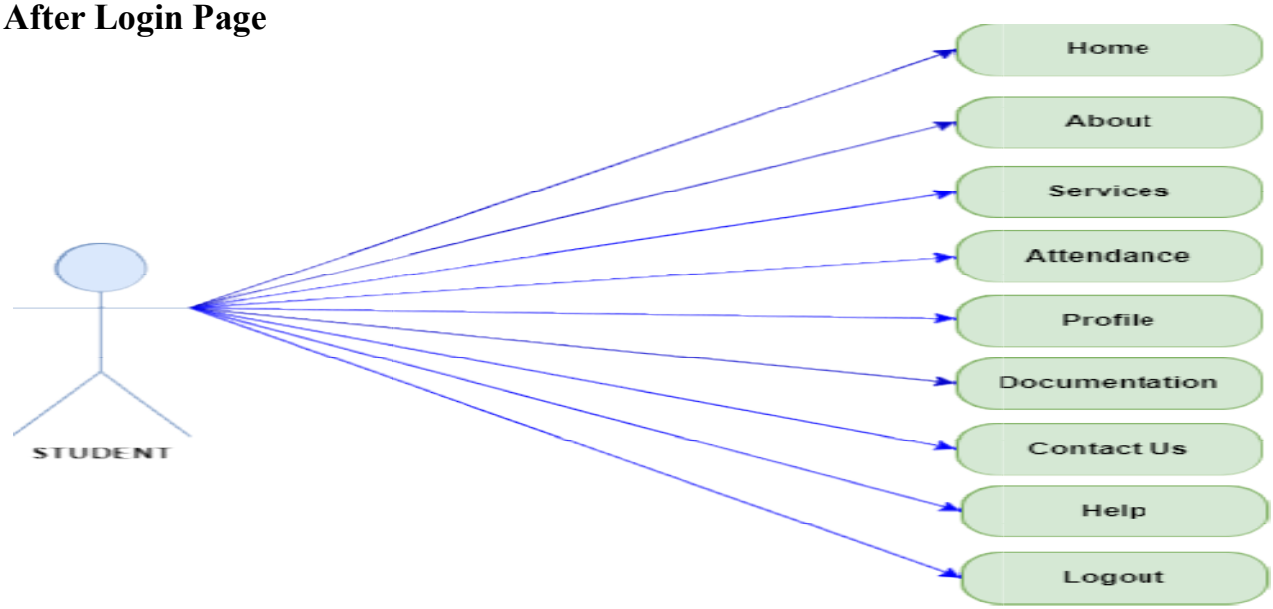


Fig: DIAGRAM OF AfFTER LOGIN PAGE FOR USER

- **Admin**

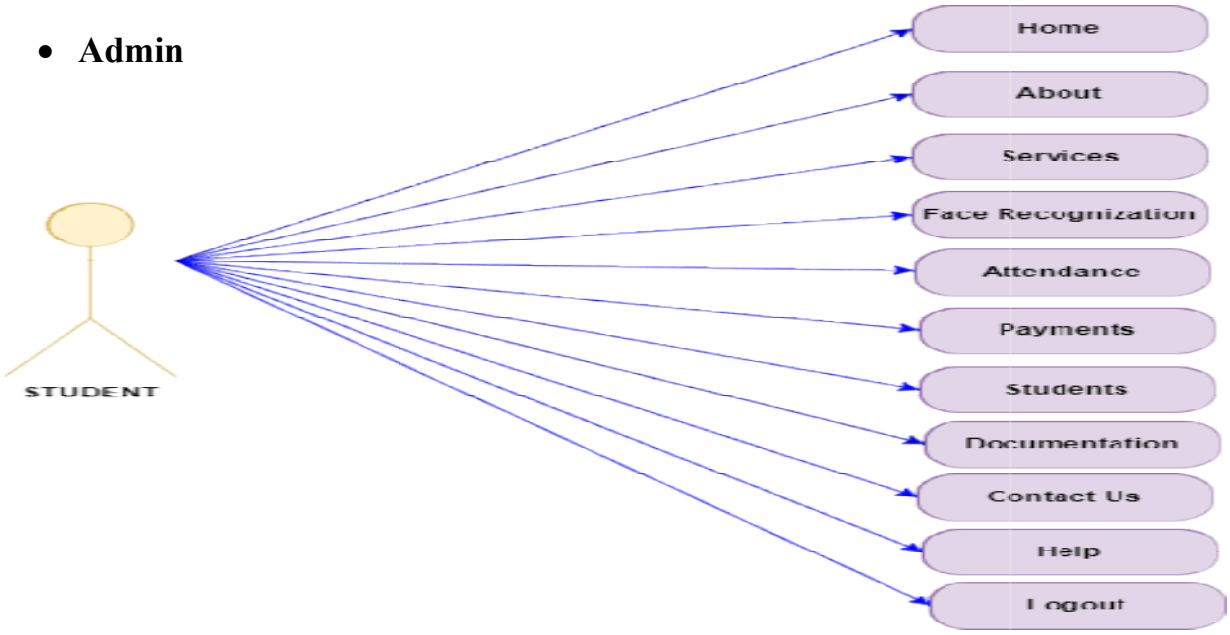
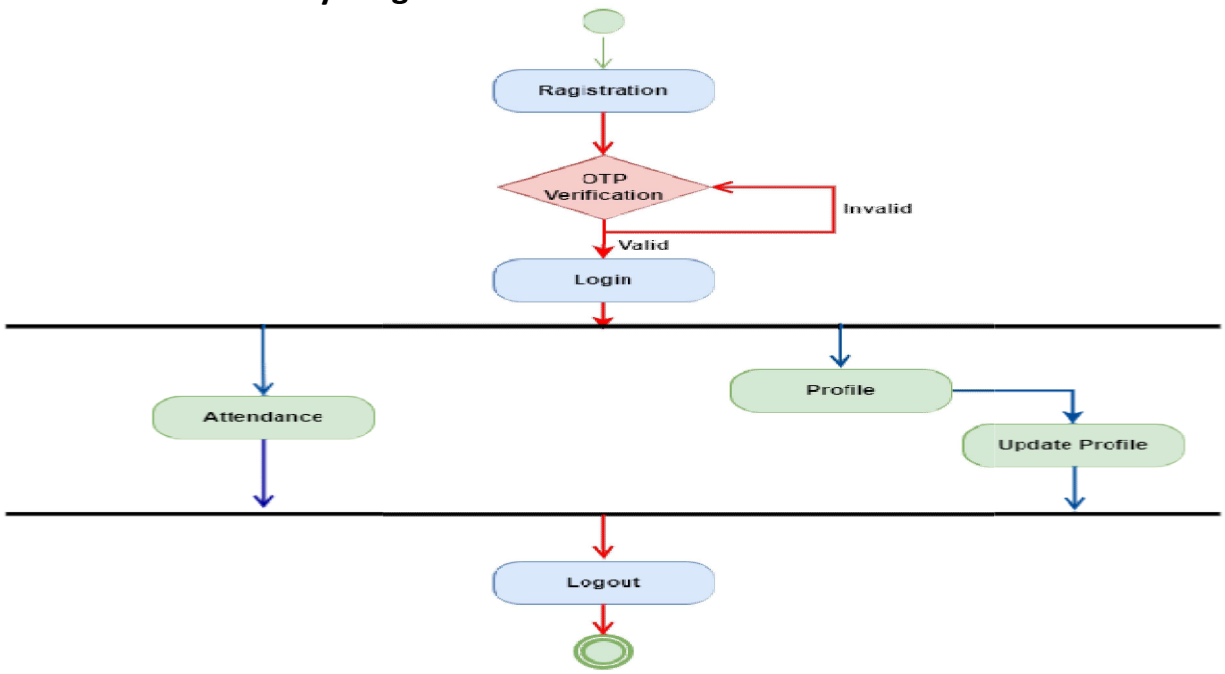


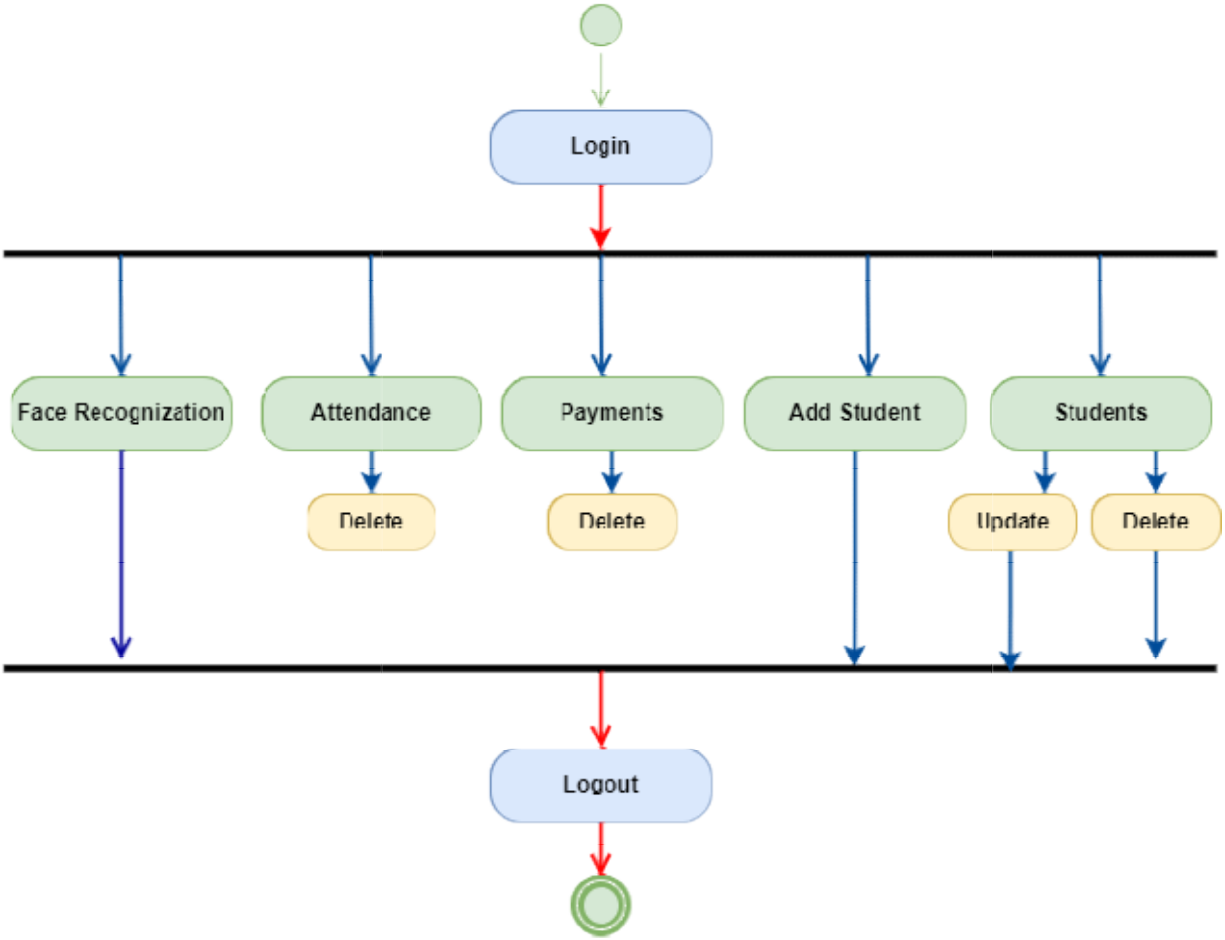
FIG: USE CASE DIAGRAM OF ADMIN PAGE

13.2 Activity Diagram

- **Student Activity Diagram**

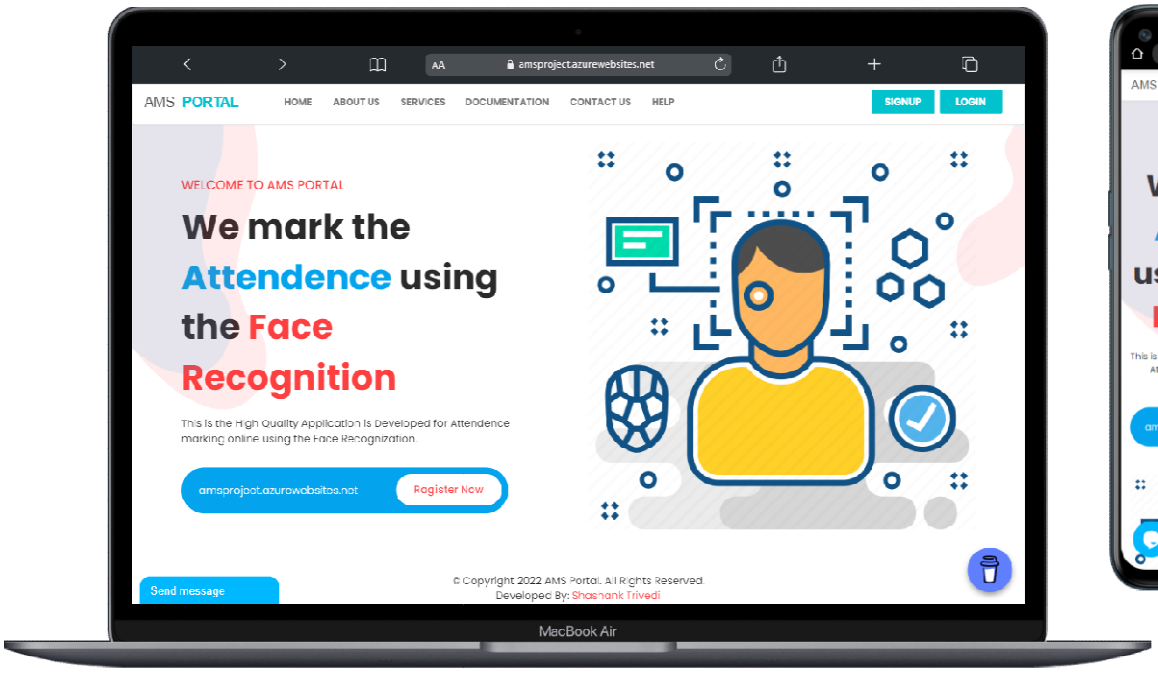


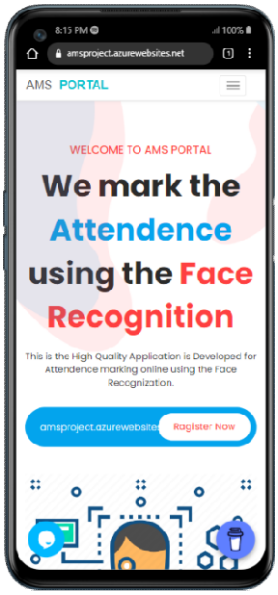
- **Admin Activity Diagram**



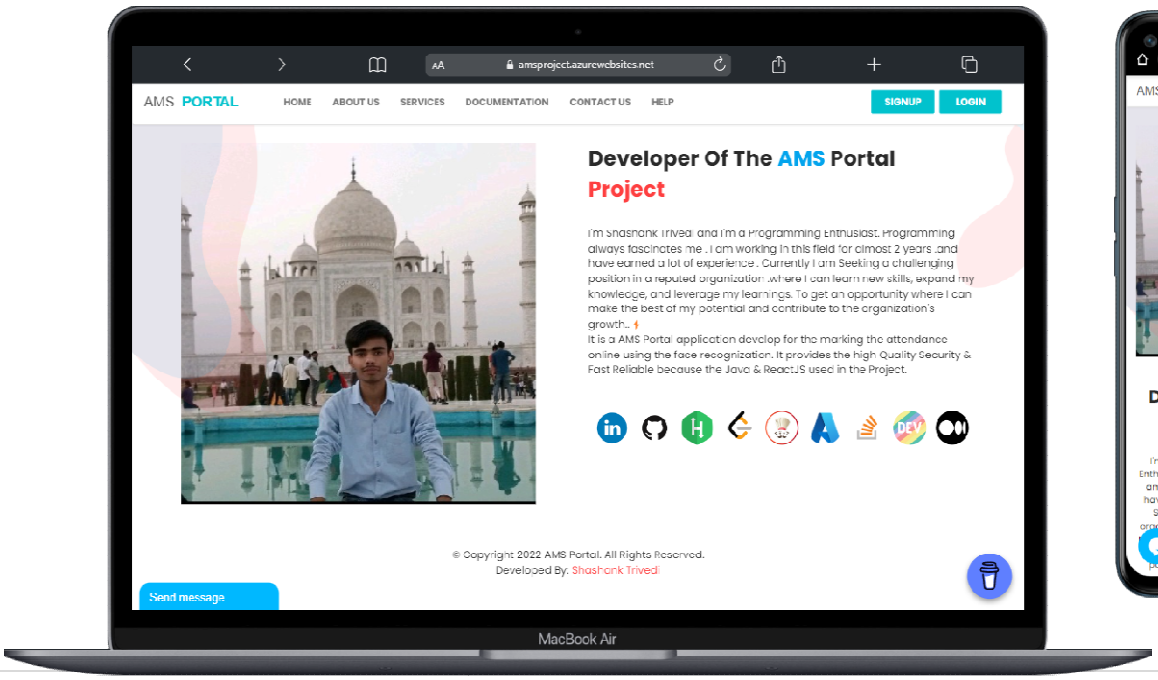
14.)SCREENSHOTS OF APPLICATION

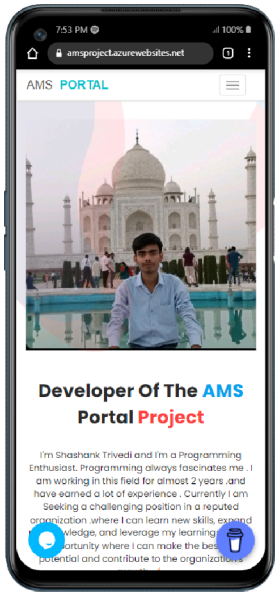
14.1 Home Page



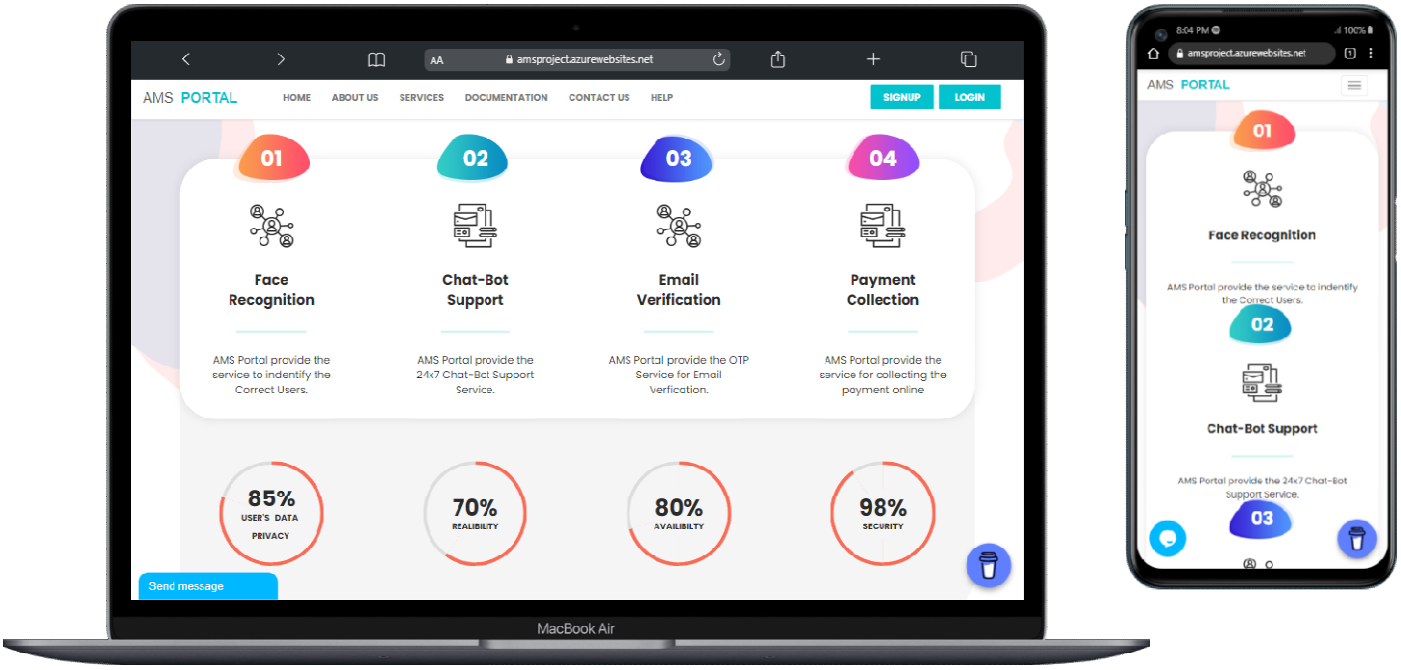


14.2 About Page

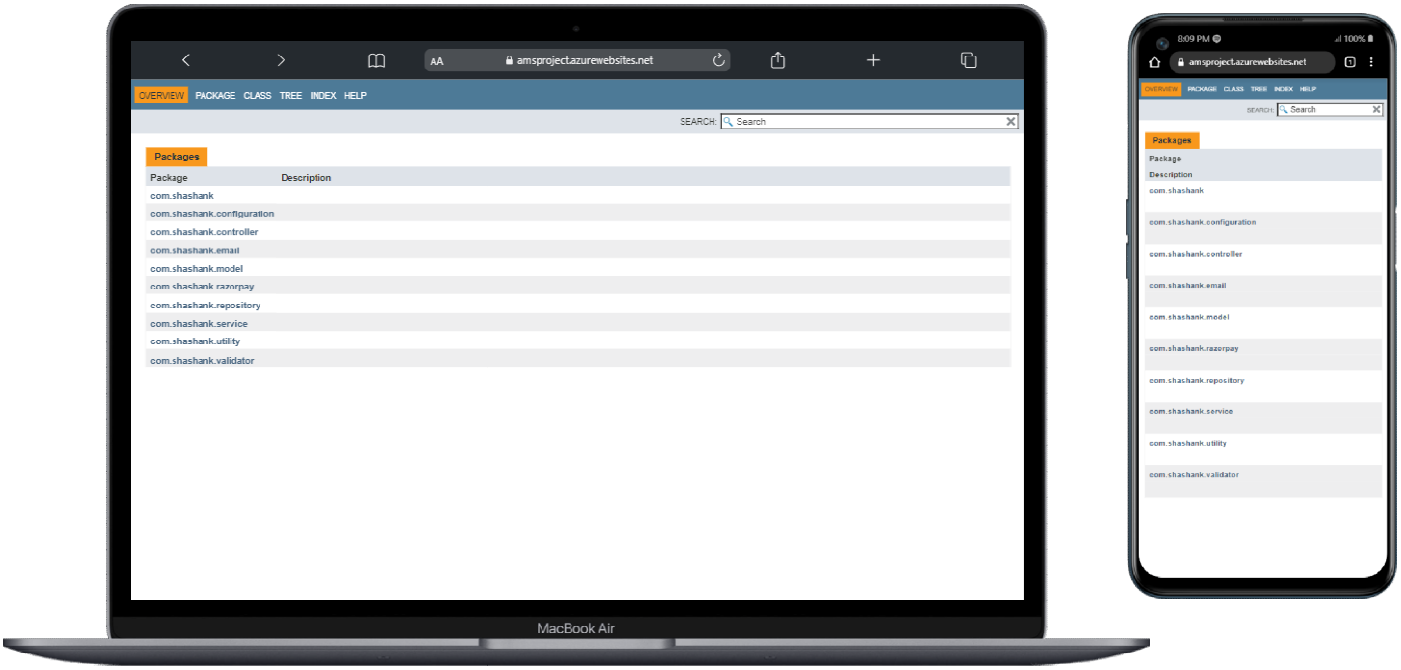




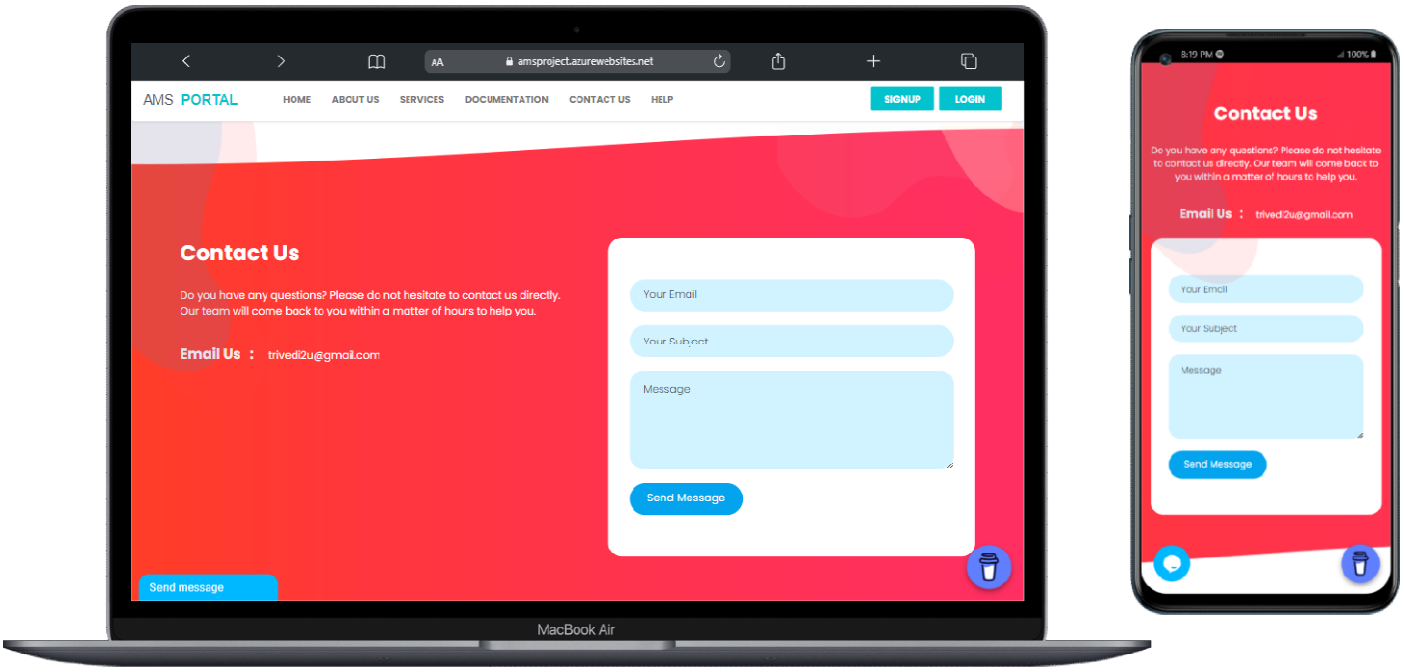
14.3 Service Page



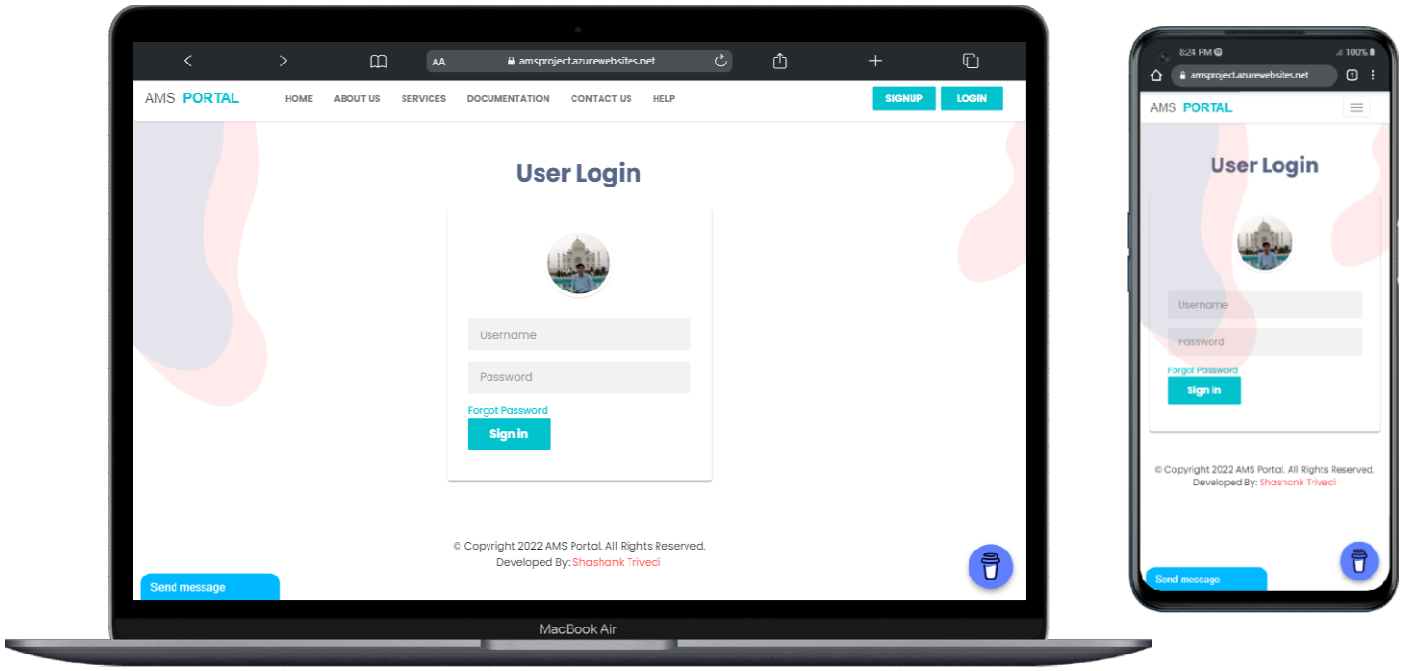
14.4 Documentation Page



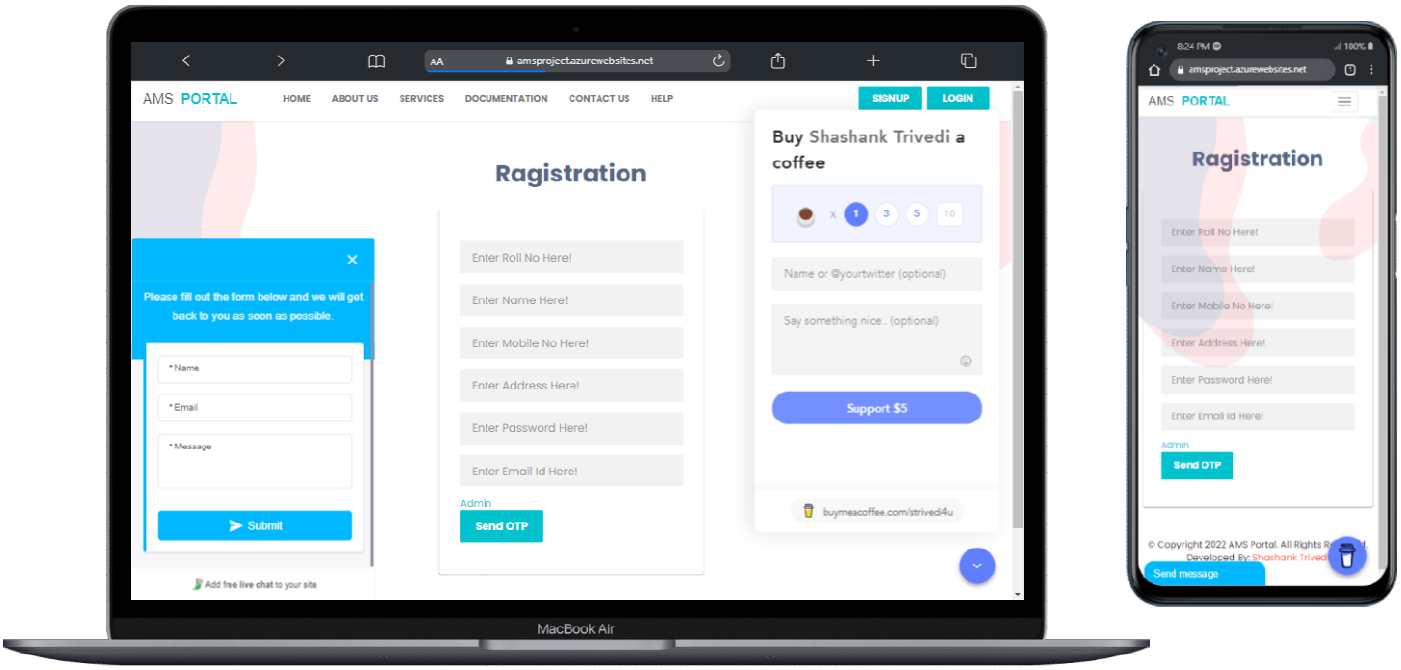
14.5 Contact Us Page



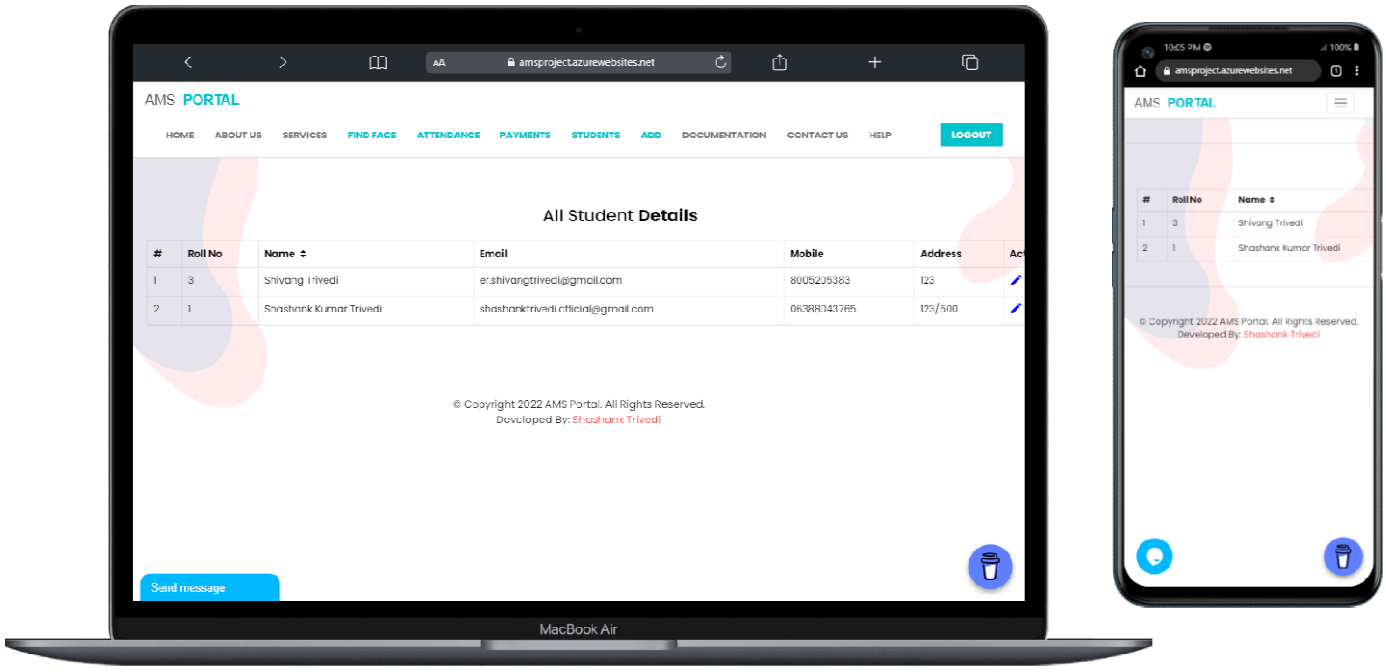
14.6 Login Page



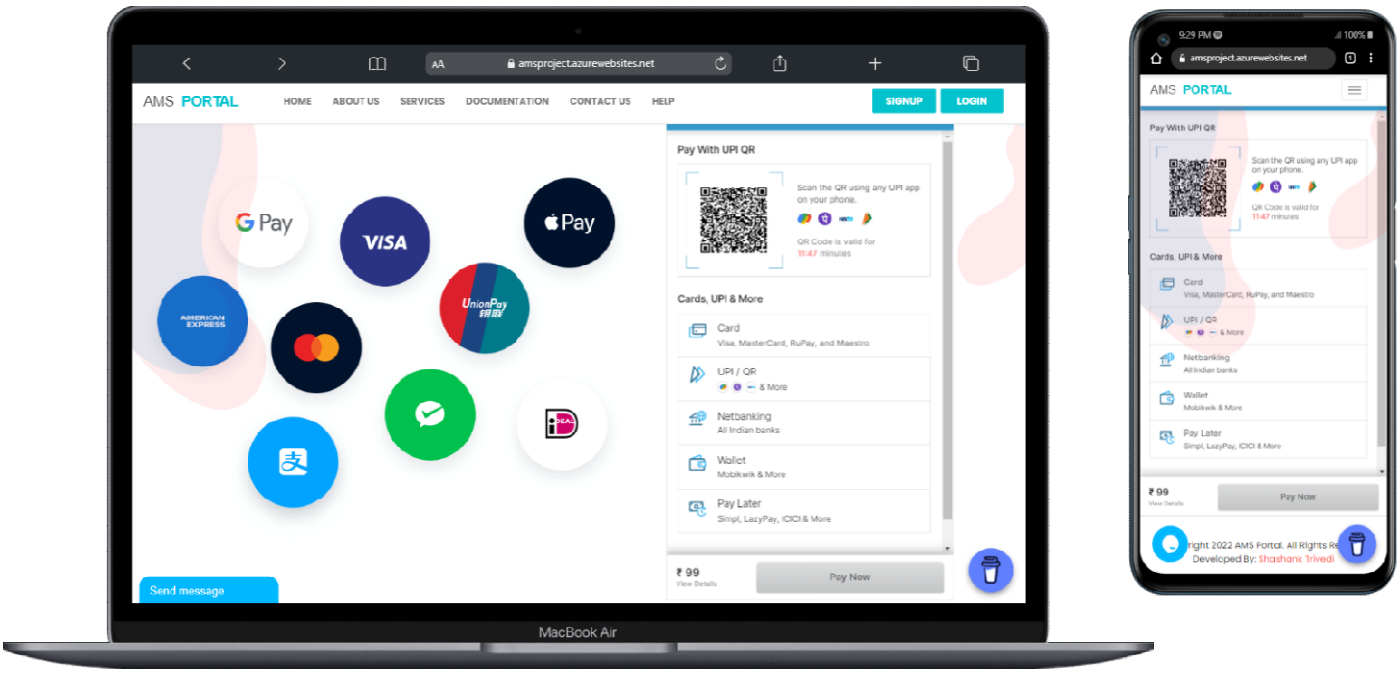
14.7 Registration Page



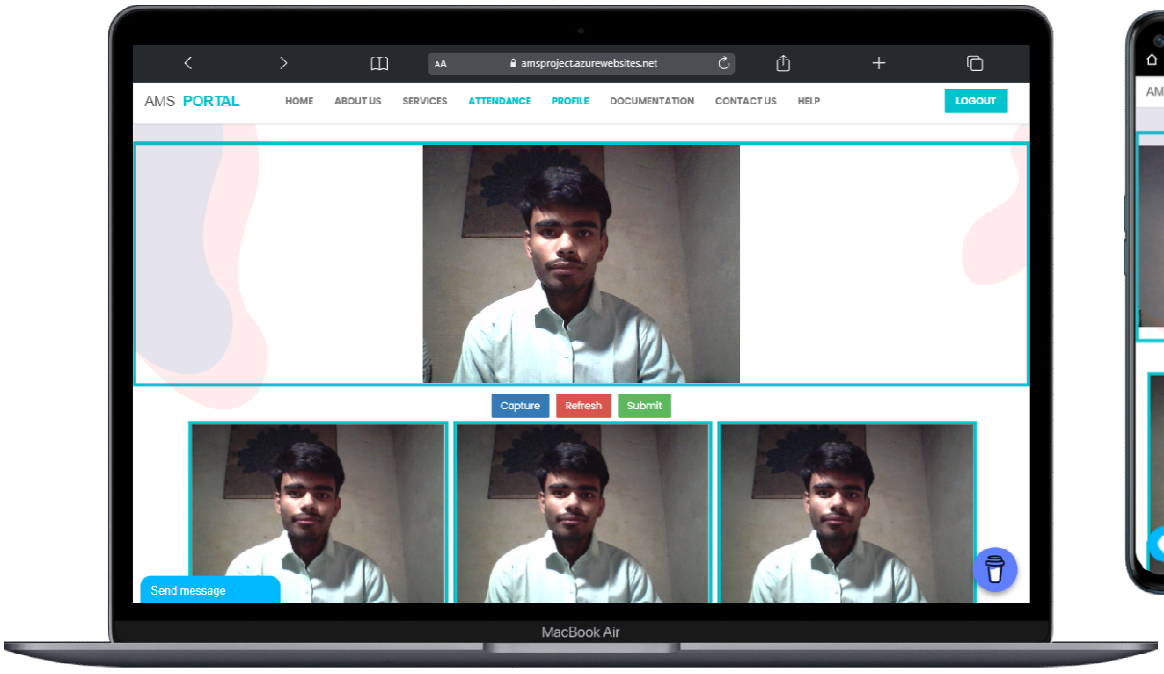
14.8 Student Page

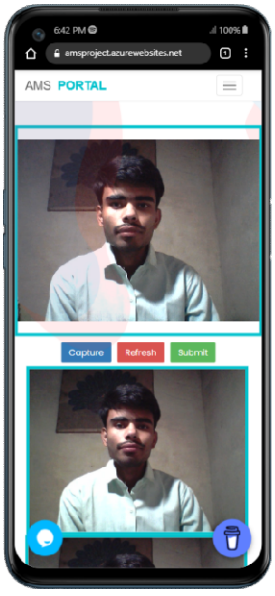


14.9 Payment Page

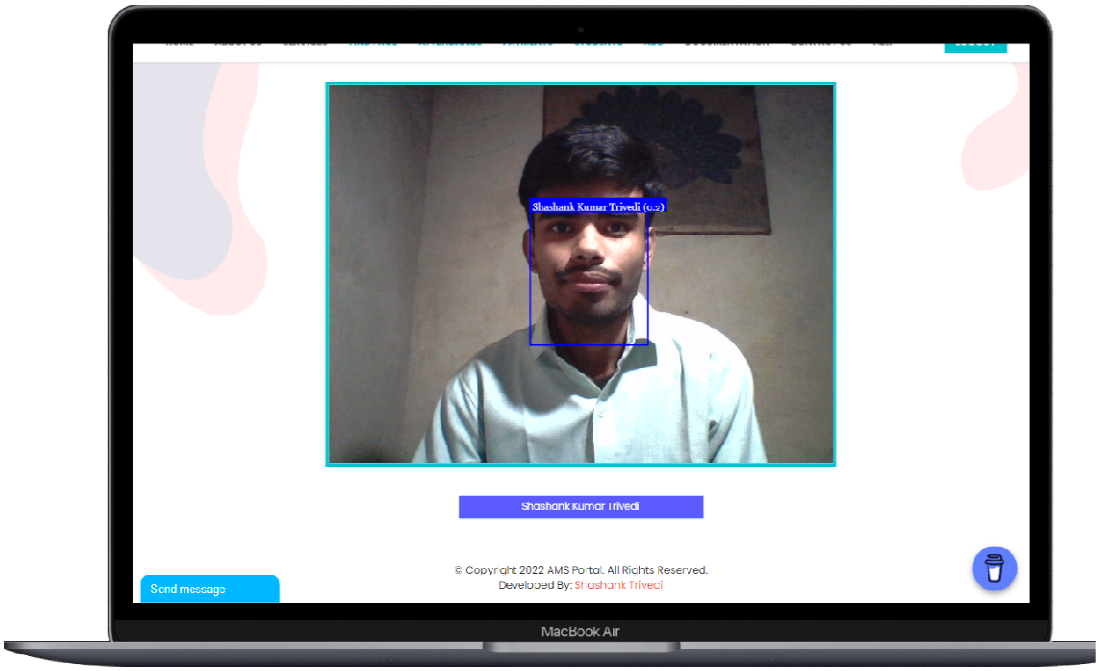


14.10 Image Capture Page

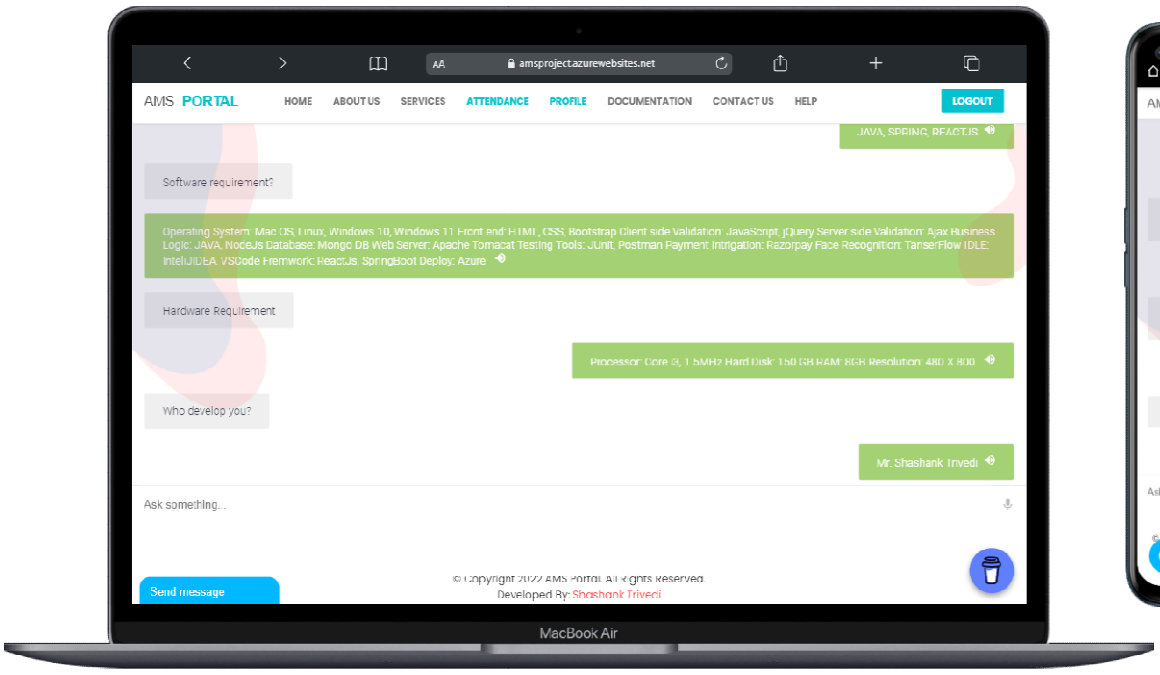




14.11 Face Recognition Page

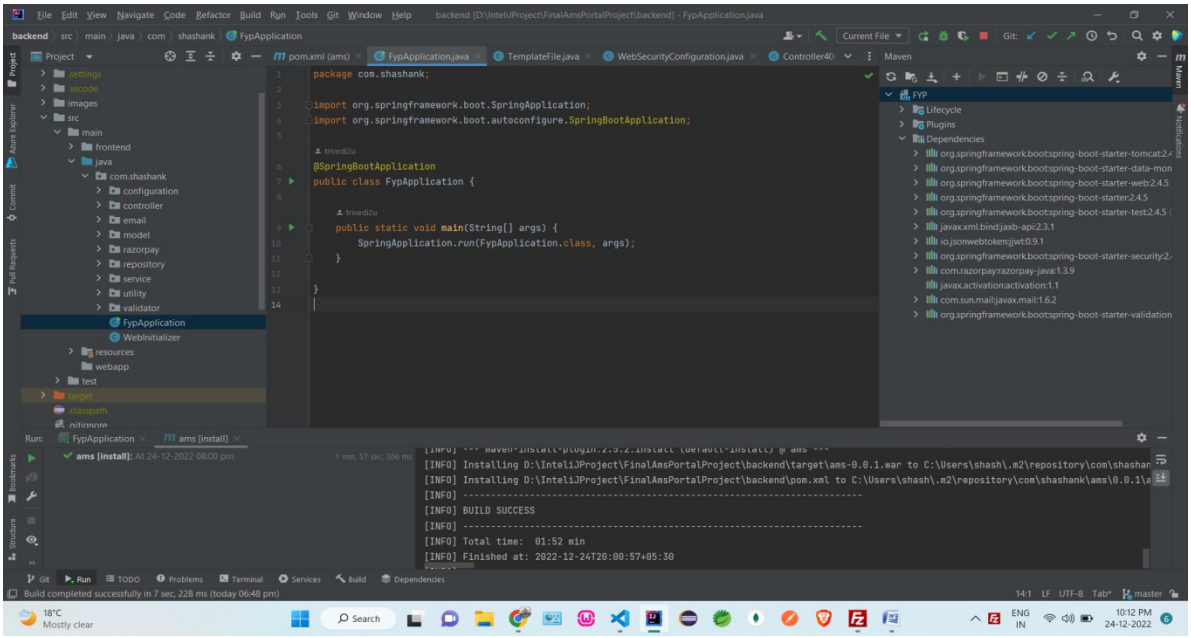


14.12 Help Bot Page

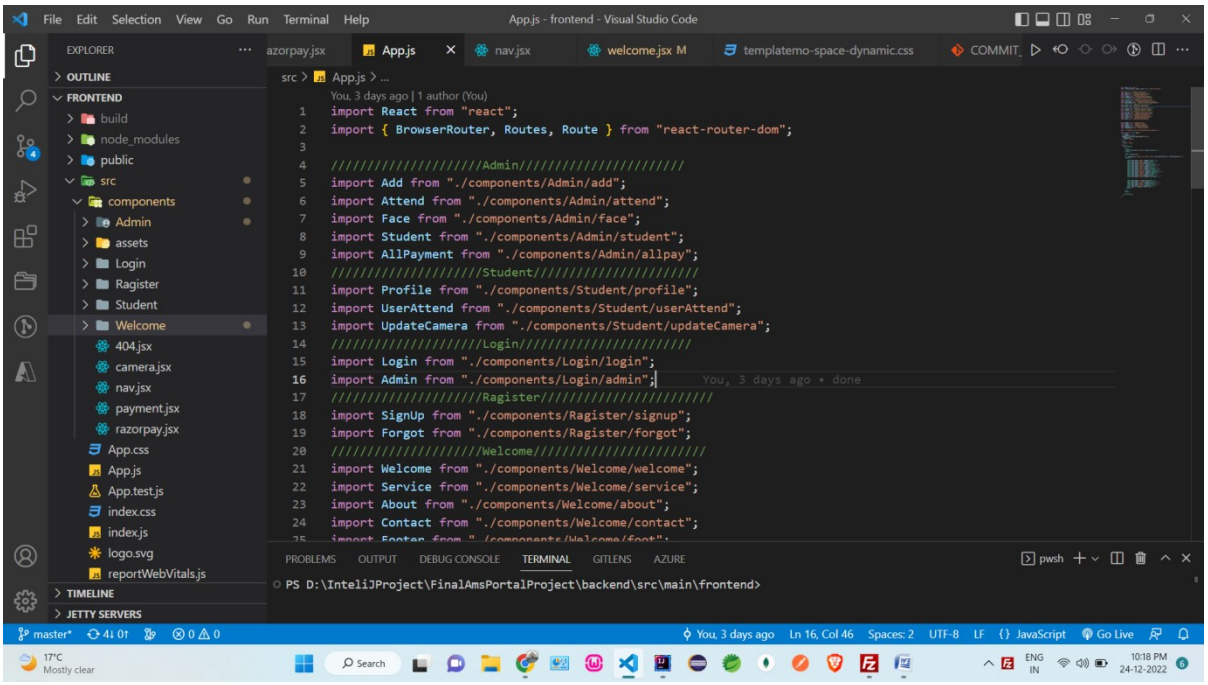


15.)SCREENSHOTS OF TOOLS

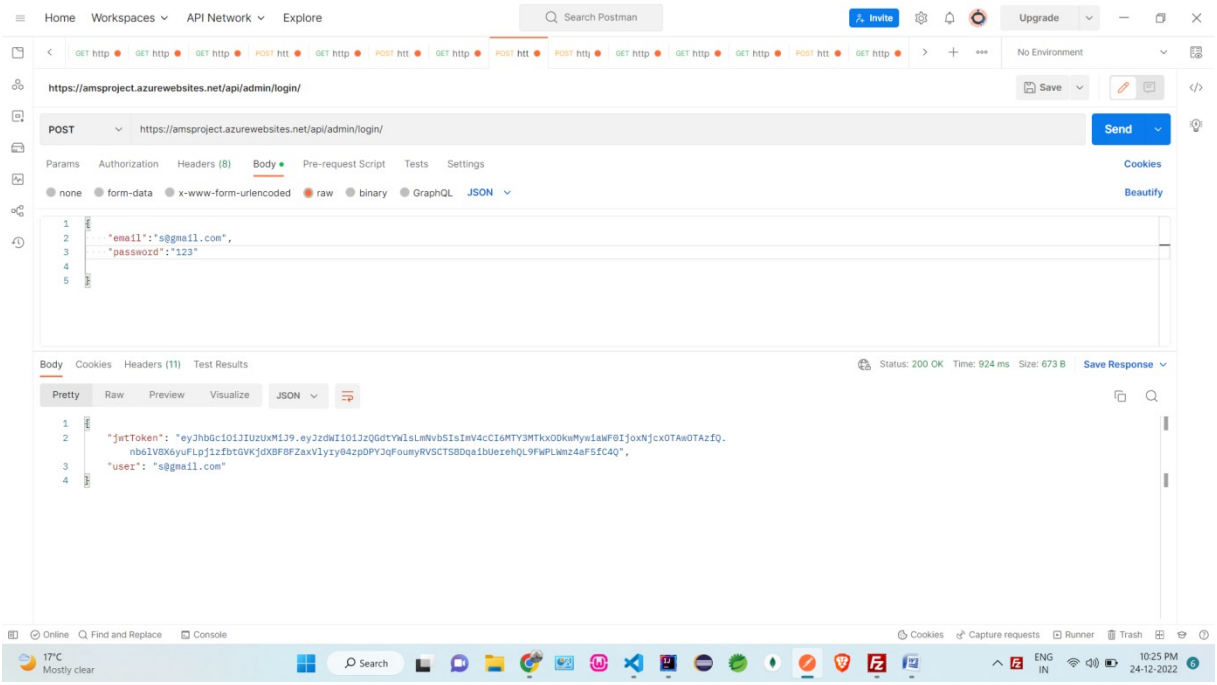
15.1 Backend on InteliJIDEA



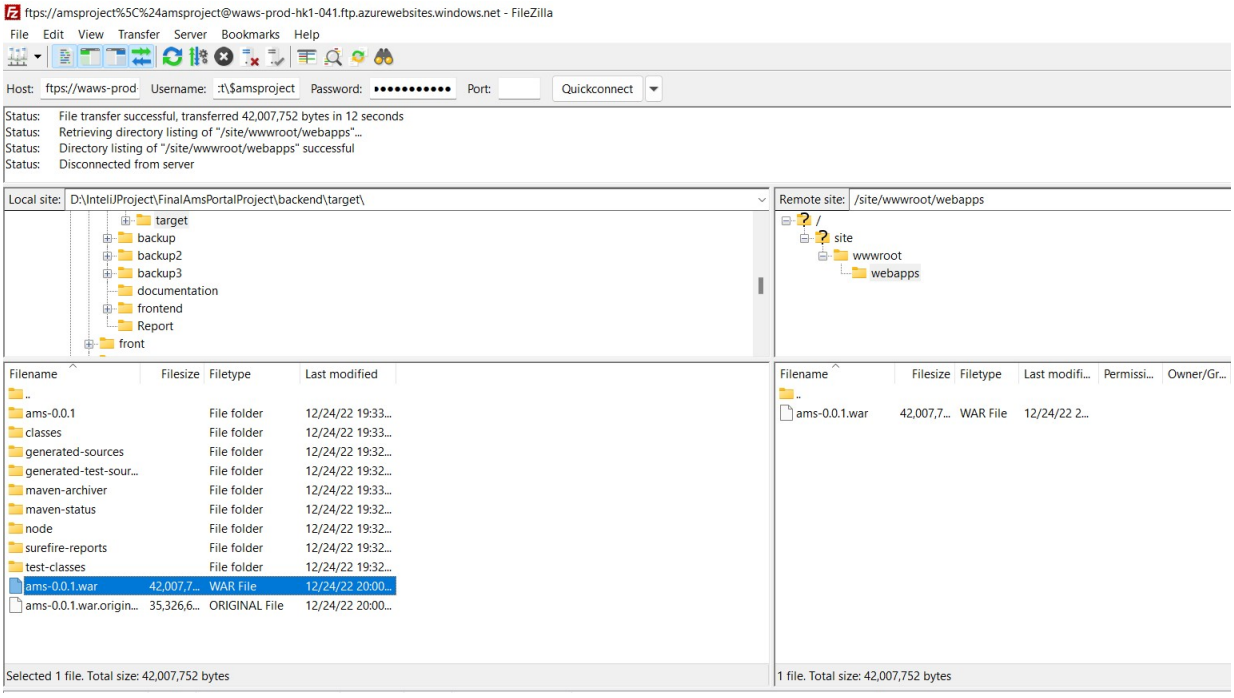
15.2 Frontend on VSCode



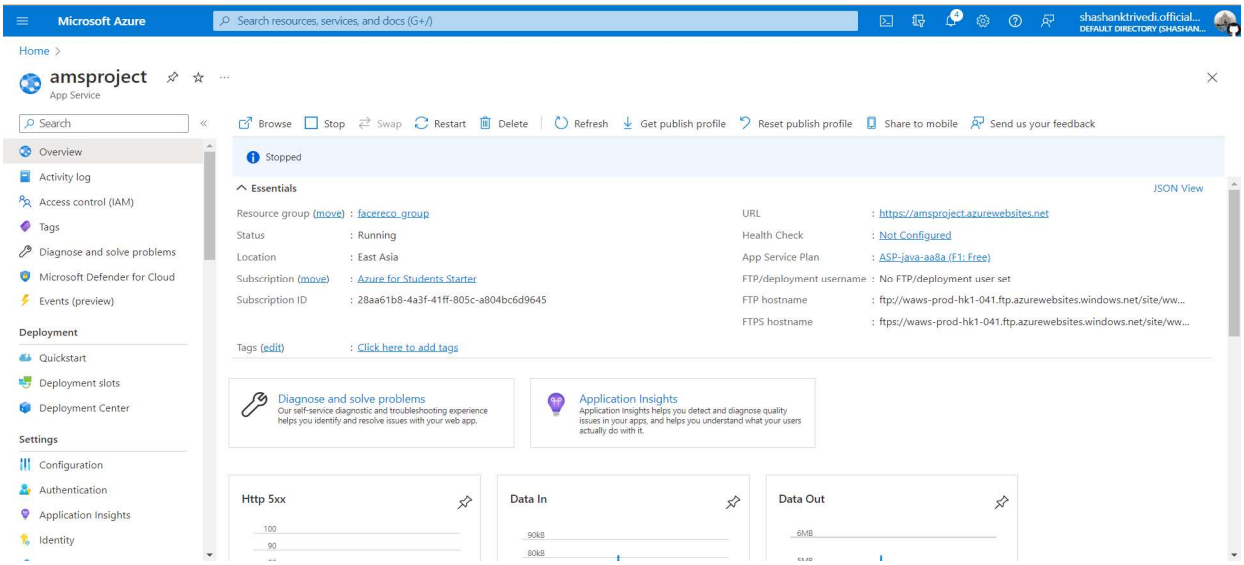
15.3 Testing on Postman



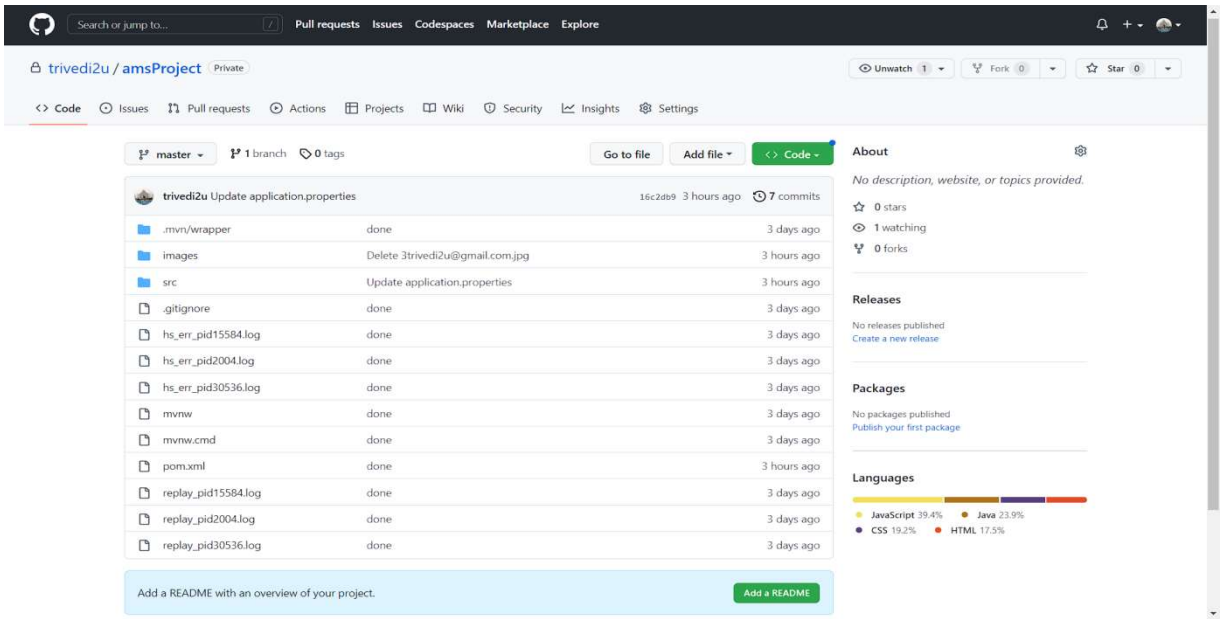
15.4 Deployment from FileZila



15.5 Deploy on Azure

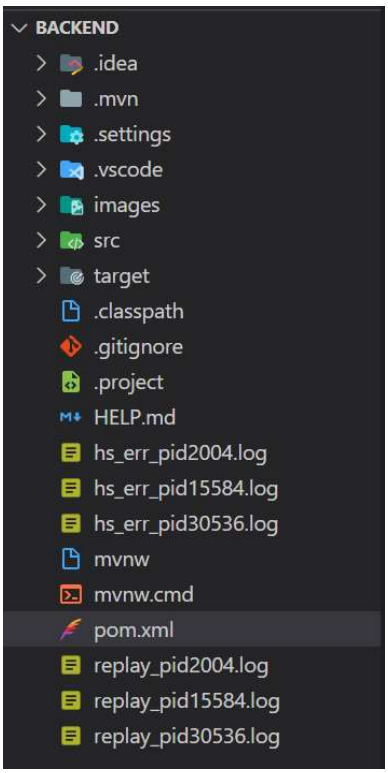


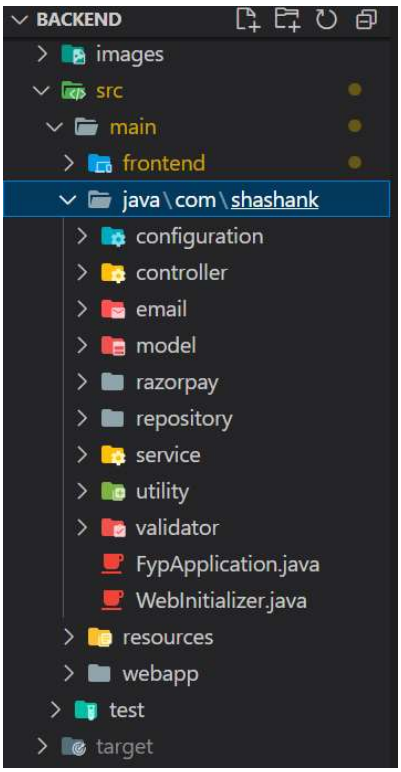
15.6 Backup on GitHub

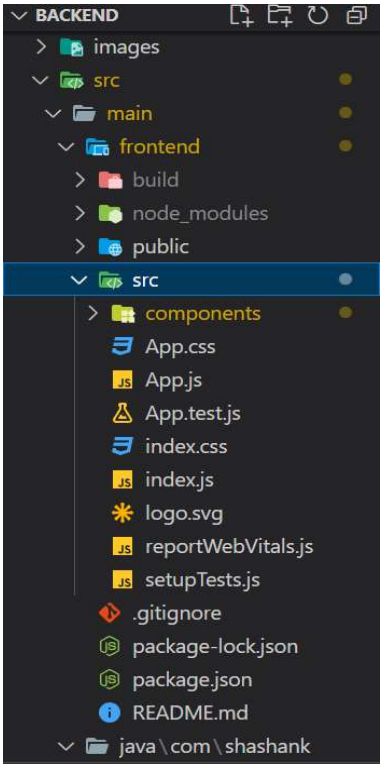


16.)SOURCE CODE

16.1 Directory Structure of the Project







16.2 FypApplication.java

```
package com.shashank;

@SpringBootApplication

public class FypApplication {

    public static void main(String[] args) {

        SpringApplication.run(FypApplication.class, args);

    }}
```

16.3 AdminController.java

```
package com.shashank.controller;
@RequestMapping("/api/admin/")
@CrossOrigin
public class AdminController {
    @Autowired
    AdminService adminService;
    @PostMapping("/")
    public String saveAdmin() throws Exception{
        adminService.save();
        return "Success";
    }}
```

16.4 AdminLoginController.java

```
package com.shashank.controller;
@RestController
@RequestMapping("/api")
@CrossOrigin
public class AdminLoginController {
    @Autowired
    private AdminLoginService adminLoginService;
    @PostMapping({"/admin/login/"})
    public JwtResponse createJwtToken(@RequestBody JwtRequest jwtRequest) throws
Exception {
        return adminLoginService.createJwtToken(jwtRequest);
    }}
```

16.5 Controller404.java

```
package com.shashank.controller;

@RestController

@CrossOrigin

public class Controller404 implements ErrorController {

    @RequestMapping("/error")

    public ModelAndView handleError(){

        ModelAndView modelAndView = new ModelAndView();

        modelAndView.setViewName("404.html");

        return modelAndView;

    }

    @Override

    public String getErrorPath() {

        return "/error";

    }}
```

16.6 PaymentController.java

```
package com.shashank.controller;
@RestController
@RequestMapping("/api/payment")
@CrossOrigin(origins = "http://localhost:3000")
public class PaymentController {
    @Autowired
    PaymentService paymentService;
    @GetMapping("/")
    private List<Payment> getAllPayment(){
        return paymentService.getAllPayment();
    }
    @PostMapping("/")
    private int savePayment(@Valid @RequestBody Payment payment) {
        paymentService.save(payment);
        return payment.getRollNo();
    }
    @DeleteMapping("/{id}")
    private void deletePayment(@PathVariable("id") int id){
        paymentService.delete(id);
    }
}
```

16.7 StudentController.java

```
package com.shashank.controller;
@RestController
@RequestMapping("/api/student")
@CrossOrigin
public class StudentController {
 @Autowired
 public StudentService studentService;
    @GetMapping("/image/")
    @PreAuthorize("hasRole('Admin')")
    public List<Student> getAllImage(){
        System.out.println("Get Call");
```

```
        System.out.println(studentService);
        return studentService.getAllImage();
    }
        @GetMapping("/")
        @PreAuthorize("hasRole('Admin')")
        public List<Student> getAllUStudent(){
            System.out.println("Get Call");
            System.out.println(studentService);
            return studentService.getAllStudent();
        }
        @PostMapping("/")
        public int saveStudent(@Valid @RequestBody Student student) {
            studentService.save(student);
            return student.getRollNo();
        }
        @PutMapping("/")
        public int updateStudent(@RequestBody Student student) {
            studentService.update(student);
            return student.getRollNo();
        }
        @DeleteMapping("/{email}")
        @PreAuthorize("hasRole('Admin')")
        public void deleteStudent(@PathVariable("email") String email){
            studentService.deleteStudent(email);
        }
        @GetMapping("/{name}")
        @PreAuthorize("hasRole('Admin')")
        public Student getAllStudentByName(@PathVariable("name") String
name){
            return studentService.getAllStudentByName(name);
        }
    @GetMapping("/email/{email}")
    @PreAuthorize("hasRole('User')")
    public Student getStudentByEmail(@PathVariable("email") String email){
        return studentService.getStudentByEmail(email);
    }
    @GetMapping("/checkEmail/{email}")
```

```
    public ResponseEntity<Student>
getStudentByEmails(@PathVariable("email") String
email){
        return studentService.checkStudentByEmail(email);
    }
    @Value("${project.image}")
    private String path;
    @PostMapping("/save/")
    public String createNewObjectWithImage(@Valid @RequestParam("model")
String model, @RequestParam("file1") MultipartFile file1,
@RequestParam("file2") MultipartFile file2, @RequestParam("file3")
MultipartFile file3) throws JsonMappingException, JsonProcessingException
{
        ObjectMapper mapper = new ObjectMapper();
        Student student = mapper.readValue(model, Student.class);
        String img1 = "1" + student.getEmail();
        String img2 = "2" + student.getEmail();
        String img3 = "3" + student.getEmail();
        student.setImg1(img1 + ".jpg");
        student.setImg2(img2 + ".jpg");
        student.setImg3(img3 + ".jpg");
        studentService.save(student);
        try {
            studentService.upload(img1, path, file1);
            studentService.upload(img2, path, file2);
            studentService.upload(img3, path, file3);
            System.out.println("Uploaded");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "Success";
    }}
```

16.8 TemplateFile.java

```
package com.shashank.controller;

@Controller

public class TemplateFile{

    @RequestMapping(value={"/login", "/admin", "/add", "/documentation/"
,"/attend" , "/student" , "/face" , "/camera", "/user", "/profile",
"/ucamera", "/signup", "/forgot" , "/student" , "/pay" , "/allpay",
"/service" , "/about" , "/student" , "/contact" , "/help", "/razorpay"  })

    public String HomePage() {

        return "";

    }

}
```

16.9 UserLoginController.java

```
package com.shashank.controller;

@RestController

@RequestMapping("/api")

@CrossOrigin

public class UserLoginController {

    @Autowired

    private UserLoginService userLoginService;

    @PostMapping({"/user/login/"})

    public JwtResponse createJwtToken(@RequestBody JwtRequest jwtRequest)
throws Exception {

        return userLoginService.createJwtToken(jwtRequest);

    }

}
```

16.10 AdminLoginService.java

```
package com.shashank.service;

@Service

public class AdminLoginService implements UserDetailsService {

    @Autowired

    private JwtUtil jwtUtil;
```

```
    @Autowired
    public AdminRepository adminRepository;
    public JwtResponse createJwtToken(JwtRequest jwtRequest){
        try {
            String userName = jwtRequest.getEmail();
            String userPassword = jwtRequest.getPassword();
            authenticate(userName, userPassword);
            UserDetails userDetails = loadUserByUsername(userName);
            String newGeneratedToken = jwtUtil.generateToken(userDetails);
            Admin admin = adminRepository.findByEmail(userName);
            return new JwtResponse(admin.getEmail(), newGeneratedToken);
        } catch (Exception e){
            return new JwtResponse(null, null);
        }

    }
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        Admin admin = adminRepository.findByEmail(username);
        if (admin != null) {
            return new org.springframework.security.core.userdetails.User(
                    admin.getEmail(),
                    admin.getPassword(),
                    getAuthority(admin)
            );
        } else {
            return null;
        }
    }
    private Set getAuthority(Admin user) {
        Set<SimpleGrantedAuthority> authorities = new HashSet<>();
        authorities.add(new SimpleGrantedAuthority("ROLE_" + "Admin"));
        return authorities;
    }
    private void authenticate(String userName, String userPassword) throws
Exception {
```

```
        try {
            Admin admin = adminRepository.findByEmail(userName);
            if (admin == null) {
                throw new BadCredentialsException("1000");
            }
            BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
            if (!encoder.matches(userPassword, admin.getPassword())) {
                throw new BadCredentialsException("1000");
            }
        } catch (DisabledException e) {
            throw new Exception("USER_DISABLED", e);

        } catch (BadCredentialsException e) {
            throw new Exception("INVALID_CREDENTIALS", e);
        } catch (Exception e){
        }
    }
}
```

16.11 AttendService.java

```
package com.shashank.service;
@Service
public class AttendService {
    @Autowired
    private AttendRepository attendRepo;
    public List<Attend> getAllAttend() {
        List<Attend> attends = new ArrayList<Attend>();
        attendRepo.findAll().forEach(stu -> attends.add(stu));
        return attends;
    }
    public void save (Attend attend) {
        LocalDate date = LocalDate.now();
        LocalTime time = LocalTime.now();
        Attend.CompositeKey key = new Attend.CompositeKey();
```

```
        String email = attend.getId().getEmail();
        key.setEmail(email);
        key.setDate(date);
        attend.setId(key);
        attend.setTime(time);
        attend.setFindId(counter++);
        attendRepo.save(attend);
    }
    public void delete (int id) {
        Attend attend = attendRepo.findId(id);
        attendRepo.delete(attend);
    }
    public Attend getAllAttendByName(String name) {
        Attend attend = new Attend();
        attendRepo.findAll();
        return attend;
    }
    public List<Attend> getAttendByRollNo(int rollNo){
        return attendRepo.findByRollNo(rollNo);
    }
}
```

16.12 PaymentService.java

```
package com.shashank.service;
@Service
public class PaymentService {
    @Autowired
    PaymentRepository paymentRepo;
    public List<Payment> getAllPayment() {
        List<Payment> payments = new ArrayList<Payment>();
        paymentRepo.findAll().forEach(stu -> payments.add(stu));
        return payments;
    }
    public void save (Payment payment) {
        LocalDate date = LocalDate.now();
```

```
        LocalTime time = LocalTime.now();
        payment.setDate(date);
        payment.setTime(time);
        paymentRepo.save(payment);
    }
    public void delete (int id) {
        paymentRepo.deleteById(id);
    }
}
```

16.13 StudentService.java

```
package com.shashank.service;
@Service
public class StudentService {
    @Autowired
    StudentRepository studentRepo;
    @Autowired
    PasswordEncoder passwordEncoder;
    public List<Student> getAllImage() {
        List<Student> students = new ArrayList<Student>();
        return studentRepo.findAllImage();

    }
    public List<Student> getAllStudent() {
        List<Student> students = new ArrayList<Student>();
        studentRepo.findAll().forEach(stu -> students.add(stu));
        return students;
    }
    public void save (Student student) {
        student.setPassword(getEncodedPassword(student.getPassword()));
        studentRepo.save(student);
    }
    public void update (Student student) {
        String email = student.getEmail();
        Student student1 = studentRepo.findByEmail(email);
```

```
        int rollNo = student.getRollNo();
        String name = student.getName();
        String mobile = student.getMobile();
        String address = student.getAddress();
        String password = student.getPassword();
        if(rollNo == 0){
            student.setRollNo(student1.getRollNo());
        }
        if(name == null){
            student.setName(student1.getName());
        }
        if(mobile == null){
            student.setMobile(student1.getMobile());
        }
        if(address == null){
            student.setAddress(student1.getAddress());
        }
        if(password == null){
            student.setPassword(student1.getPassword());
        }
        student.setPassword(getEncodedPassword(student.getPassword()));
        studentRepo.save(student);
    }

    public void delete (String email) {
        studentRepo.deleteById(email);
    }
    public Student getAllStudentByName(String name){
       return studentRepo.getAllStudentByName(name);
    }
    public Student getStudentByEmail(String email){
        return studentRepo.findByEmail(email);
    }
    public ResponseEntity<Student> checkStudentByEmail(String email){
        try {
            Student student = studentRepo.findByEmail(email);
            if((student.getEmail()).equals(email)){
```

```
                return new ResponseEntity<Student>(
                        student,
                        HttpStatus.OK);
            }else {
                return new ResponseEntity<>(
                        student,
                        HttpStatus.BAD_REQUEST);
            }
        }catch (Exception e){
            return new ResponseEntity<>(
                    HttpStatus.BAD_REQUEST);
        }
    }
    public void upload(String id, String path, MultipartFile file){
        String name = file.getOriginalFilename();
        String filePath = path + File.separator + id + ".jpg";
        Path paths = Paths.get(filePath);
        File f = new File(path);
        if(!f.exists()) {
            f.mkdir();
        }
        try {
            Files.delete(paths);
            System.out.println(id);
        }
        catch (Exception e){
            System.out.println(e);
        }
        try {
           Files.copy(file.getInputStream(), paths);
        }
        catch (Exception e){
            System.out.println(e);
        }

    }
    public void deleteStudent(String email)  {
```

```
        studentRepo.deleteById(email);
        Path path1 = Paths.get("./images/" + "1" + email + ".jpg");
        Path path2 = Paths.get("./images/" + "2" + email + ".jpg");
        Path path3 = Paths.get("./images/" + "3" + email + ".jpg");
        try {
            Files.delete(path1);
            Files.delete(path2);
            Files.delete(path3);
            System.out.println("File or directory deleted successfully");
        } catch (NoSuchFileException ex) {
            System.out.println("No such file or directory:");
        } catch (DirectoryNotEmptyException ex) {
            System.out.println("Directory %s is not empty");
        } catch (IOException ex) {
            System.out.println(ex);
        }
    }
    public String getEncodedPassword(String password) {
        return passwordEncoder.encode(password);
    }
}
```

16.14 UserLoginService.java

```
package com.shashank.service;
@Service
public class UserLoginService implements UserDetailsService {
    @Autowired
    private JwtUtil jwtUtil;
    @Autowired
    public StudentRepository studentRepository;
    public JwtResponse createJwtToken(JwtRequest jwtRequest){
        try {
            String userName = jwtRequest.getEmail();
            String userPassword = jwtRequest.getPassword();
            authenticate(userName, userPassword);
```

```
            UserDetails userDetails = loadUserByUsername(userName);
            String newGeneratedToken = jwtUtil.generateToken(userDetails);
            Student student = studentRepository.findByEmail(userName);
            return new JwtResponse(student.getEmail(), newGeneratedToken);
        }
        catch (Exception e){
            return new JwtResponse("Error", null);
        }
    }
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        Student student = studentRepository.findByEmail(username);
        if (student != null) {
            return new org.springframework.security.core.userdetails.User(
                    student.getEmail(),
                    student.getPassword(),
                    getAuthority(student)
            );
        } else {
            throw new UsernameNotFoundException("User not found with
username: " + username);
        }
    }
    private Set getAuthority(Student user) {
        Set<SimpleGrantedAuthority> authorities = new HashSet<>();
        authorities.add(new SimpleGrantedAuthority("ROLE_" + "User"));
        return authorities;
    }
    private void authenticate(String userName, String userPassword) throws
Exception {
        try {
            Student student = studentRepository.findByEmail(userName);
            if (student == null) {
                throw new BadCredentialsException("1000");
            }
            BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
            if (!encoder.matches(userPassword, student.getPassword())) {
```

```
                throw new BadCredentialsException("1000");
            }
        } catch (DisabledException e) {
            throw new Exception("USER_DISABLED", e);

        } catch (BadCredentialsException e) {
            throw new Exception("INVALID_CREDENTIALS", e);
        } catch (Exception e){
        }
    }
}
```

15.)FUTURE SCOPE OF WORK

The project has a very vast scope in future. The project can be implemented on organization and company. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner. The following are the future scope for the project.

- The scope of Attendance Management System is widening and today it offers a strong support to the college campus as well as any organization in terms of providing the much desired touch of security and identity with accuracy.
- The bright future prospect of maintaining records with data analysis is also proven with the fact that the technology is integrated in mobile phones as well.
- The project is developing with the power of complete support with AI bot and email.
- Since the PII and user data is highly confidential, hence mostly trusted secured language JAVA is used as backend while creating the application.
- By the huge data of attendance the data-analytics can be performed as to get the spike of attendance in which month and so on.

16.)CONCLUSION

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements. The objective of software planning is to provide a frame work that enables the manager to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

17.)BIBLIOGRAPHY

- ✓ https://docs.spring.io/spring-framework/docs/current/reference/html/
- ✓ https://docs.oracle.com/en/java/
- ✓ www.w3school.com
- ✓ www.stackoverflow.com
- ✓ www.tutorialspoint.com
- ✓ www.javapoint.com
- ✓ www.youtube.com
- ✓ www.wikipedia.com