HOSTEL GATE PASS MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

AMAN KUMAR SINGH	**(620820104004)**
ARVIND KUMAR SAH	**(620820104009)**
RAJAN KUMAR MAHTO	**(620820104078)**
RAMPRAVESH KUMAR YADAV	**(620820104081)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

GNANAMANI COLLEGE OF TECHNOLOGY

NAMAKKAL - 637 018

ANNA UNIVERSITY:: CHENNAI 600 025

May 2024

HOSTEL GATE PASS MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

AMAN KUMAR SINGH	**(620820104004)**
ARVIND KUMAR SAH	**(620820104009)**
RAJAN KUMAR MAHTO	**(620820104078)**
RAMPRAVESH KUMAR YADAV	**(620820104081)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

GNANAMANI COLLEGE OF TECHNOLOGY

NAMAKKAL - 637 018

ANNA UNIVERSITY:: CHENNAI 600 025

May 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“HOSTEL GATE PASS MANAGEMENT SYSTEM”** is the bonafide work of “**AMAN KUMAR SINGH (620820104004), ARVIND KUMAR SAH (620820104009), RAJAN KUMAR MAHTO (620820104078) and RAMPRAVESH KUMAR YADAV (620820104081)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.R.UMAMAHESWARI, Ph.D.,	**Mr.M.S.SABARI, AP/CSE**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Head Of the Department,	Assistant Professor,
Computer Science and Engineering,	Computer Science and Engineering,
Gnanamani College of Technology,	Gnanamani College of Technology,
Namakkal-637 018.	Namakkal-637 018.

Submitted for the University project viva voce Examination held on __ / __ /____

Internal Examiner External Examiner

ACKNOWLEDGEMENT

We would like to express our deep sense of heartiest thanks to our beloved Chairman **Dr.T.ARANGANNAL** and Chairperson **Smt.P.MALALEENA,** Gnyanamani Educational Institutions, Namakkal, for giving an opportunity to do and complete this project. **Ms.MADHUVANTHINIE ARANGANNAL,** Vice Chairman, Gnyanamani Educational Institutions, Namakkal, for her encouragement during our project work.

We would like to express our sincere gratitude to our Chief Administrative Officer **Dr.P.PREMKUMAR,** Gnyanamani Educational Institutions, Namakkal, for providing us with indefinable support.

We would like to express our Principal, Gnanamani College of Technology, Namakkal, for creating a beautiful atmosphere deep sense of gratitude and profound thanks to **Dr.T.K.KANNAN,** which inspired us to take over this project.

We take this opportunity to convey our heartiest thanks to **Dr.R.UMAMAHESWARI,** Professor & Head Department of Computer Science and Engineering, pillar of support for the successful completion of the project.

We are thanks to my Project Coordinator **Mrs. A. SANGEETHA,** Assistant Professor of Computer Science and Engineering department, for providing good suggestions and help me to make this project a success.

We are extremely grateful to our Guide, **Mr. M.S SABARI,** Assistant Professor of Computer Science and Engineering department, for giving this opportunity with full encouragement to complete this project.

We express my sincere words of thanks of my parents, friends and all staff members of Department of computer science engineering, Gnanamani College of Technology, blessing to complete the project successfully.

[AMAN KUMAR SINGH] [ARVIND KUMAR SAH] [RAJAN KUMAR MAHTO] [RAMPRAVESH KUMAR]

INSTITUTE

VISION

Emerging as a technical institution of high standard and excellence to produce quality Engineers, Researchers, Administrators and Entrepreneurs with ethical and moral values to contribute the sustainable development of the society.

MISSION

We facilitate our students

MI1: To have in-depth domain knowledge with analytical and practical skills in cutting edge technologies by imparting quality technical education.

MI2: To be industry ready and multi-skilled personalities to transfer technology to industries and rural areas by creating interests among students in Research and Development and Entrepreneurship.

DEPARTMENT

VISION

To evolve as a Centre of Excellence to produce the most competent software professionals, researchers, entrepreneurs and academicians with ethical values in Computer Science and Engineering.

MISSION

We facilitate our students

MD1: Imparting quality education through latest technologies to prepare students as software developer and system analyst.

MD2: Inculcating the technological transformations for the sustainable development of society.

MD3: Promoting excellence towards higher education, research, employability and entrepreneurship.

ABSTRACT

The HOSTEL GATE PASS MANAGEMENT SYSTEM (HGMS) is an objective and the scope of this paper is to record the details, various activities of the students.

Gate Pass management system is the software application, which is used to manage the entry and exit of the students on short term and long term leave. Gate Pass Management system is an efficient system, to keep track of each student’s entry and exit.

The gate pass application form is subjected to approval by Hostel warden and Parents and finally the gate pass is issued by the hostel warden. The system is easy to use and manage, as it will simplify the task and reduce the paper work.

The Hostel Gate Pass Management System (HGMS) is a comprehensive software solution designed to streamline and automate the process of managing gate passes for hostel residents in educational institutions, corporate hostels, or similar facilities. This system aims to replace traditional paper-based gate pass issuance methods with an efficient, digital approach.

HGMS facilitates the generation, issuance, tracking, and verification of gate passes through a user-friendly interface accessible to both hostel administrators and residents. Administrators can create and maintain resident profiles, manage gate pass requests, and monitor hostel occupancy in real-time. Residents can conveniently submit gate pass requests, track their status, and receive notifications on approval or rejection.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	**iii**
	LIST OF FIGURES	**vi**
	LIST OF ABBREVIATIONS	**viii**
1	**INTRODUCTION**	**8**
	1.1 PROJECT OVERVIEW	8
2	**LITRATURE SURVEY**	**11**
	2.1 RELATED WORKS	
	2.1.1 Campus Management Systems	11
	2.1.2 Front End-Dream aver	12
	2.1.3 Dreamweaver	12
	2.1.4 Requirement Analysis	12
	2.1.5 System Design	12
	2.1.6 Development System	13
	2.1.7 Application of HGMS System	13
	2.1.8 Language Support	14

3	**PROJECT DESCRIPTION**	**17**
	3.1 PROPOSED SYSTEM	17
	3.1.1 About Proposed system	17
	3.1.2 Proposed Architecture	19
4	**HARDWARE COMPONENTS**	**20**
	4.1 HARDWARE REQUIREMENTS	20
	4.2 LAPTOP / COMPUTER	23
5	**SOFTWARE COMPONENTS**	**25**
	5.1 SOFTWARE REQUIREMENT	25
	5.2 JAVASCRIPT	25
	5.3 NODE HTTP SERVER	26
	5.4 REACT	26
	5.5 MONGODB	27
	5.6 EXPRESS	27
	5.7 BOOTSTRAP	27
6	**WORKFLOW DIAGRAMS**	28
	6.1 WORKFLOW DIAGRAM	29
7	**CONCLUSION & FUTURE ENHANCEMENT**	29
	7.1 CONCLUSION	29
	7.2 FUTURE ENHANCEMENT	29
	APPENDIX 1: SOURCE CODE	30
	APPENDIX 2: SCREENSHOTS	40
	REFERENCES	43

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
2.1.8.1	Recruitment Process flow chart	15
2.1.8.2	Context Level DFD	15
2.1.8.3	Student Module	15
2.1.8.4	Registration Process	16
2.1.8.5	Administrator Module	16
3.1.2.1	User Requirement process	19
4.1.1.1	Server Processing Model	20
4.1.2.1	Data Storage process	21
4.1.3.1	Required networking Infrastructure	22
4.1.4.1	workstation of HGMS	23
4.1.5.1	Backup and Disaster Recovery	24
6.1.1	Flow Diagram	28

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSIONS
SQL	Structural Query Language
HTML	Hyper Text Markup Language
Gemini	Google Gemini
ChatGPT	OpenAI
KB	Kilo Bytes
GNU	Ground
IDE	Integrated Development Environment
MHz	Mega Hertz
NTC	National Technology Center
TTL	Time To Live

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

A Hostel Gate Pass Management System is a software solution designed to streamline the process of issuing and managing gate passes for individuals entering or leaving a hostel premises. Here's an overview of its functionality and features: Hostel Gate Pass Management System (HGMS).

In Hostel Gate Pass System, we have seen various industries or institutes, where there is a compulsion of taking a gate pass to enter the premises as well as while leaving the premises before time. The gate pass is generally either a written chit of paper or a receipt sheet. This sheet remains of no use once, out of the premises and there is wastage of paper. Our system proposes to give the "Paper Saving Idea/Paper less office" by the Use of Technology for the same.

It consists of a design where the students can enter the required details with student's ID, Name, Contact number, reason to enter/leave, the type of student (sport/normal) and type of leave (day out/ night stay). The higher authority would contain the details of the students from database. If this is true then details of student would be saved into database and once itis done, the permission is granted, then the respected student will get a confirmation the/she is allowed to enter/leave the premises before time through Message/SMS.

The gate pass application form is subjected to approval by Hostel warden and Parents and finally the gate pass is issued by the hostel warden. The system is easy to use and manage, as it will simplify the task and reduce the paper work:

User Registration: The system allows the registration of hostel residents, visitors, and staff members. Each user is assigned a unique identifier for tracking purposes.

Gate Pass Issuance: Authorized personnel can issue gate passes to hostel residents and visitors for various purposes such as temporary leave, emergencies, or special events. Gate passes may include details like the purpose, duration, and destination of the visit.

Approval Workflow: Gate pass requests typically go through an approval process. The system facilitates this by routing requests to the appropriate authorities for review and approval.

Access Control: The system ensures that only authorized individuals are granted access to the hostel premises. It may integrate with access control systems to manage gate entry and exit points**.**

Real-time Monitoring: Hostel administrators can monitor the movement of individuals in real-time using the system. This helps maintain security and track the whereabouts of hostel residents and visitors.

Hostel Gate Pass Management System enhances security, improves operational efficiency, and provides better control over access to hostel premises. It simplifies the process of issuing gate passes while ensuring compliance with hostel regulations and safety standards.

Reporting and Analytics: The system generates reports on gate pass activities, including the number of passes issued, approved, and denied, as well as trends over time. These insights help in assessing security risks and optimizing the gate pass management process.

Integration: The system may integrate with other hostel management systems such as student databases, visitor management systems, and security systems to streamline operations and ensure data consistency**.**

Overall, a Hostel Gate Pass Management System enhances security, improves operational efficiency, and provides better control over access to hostel premises. It simplifies the process of issuing gate passes while ensuring compliance with hostel regulations and safety standards.

Security: Data security and privacy are paramount in a hostel gate pass management system. The system employs robust authentication and encryption mechanisms to safeguard sensitive information and prevent unauthorized access.

CHAPTER 2

LITRATURE SURVEY

2.1 RELATED WORKS

2.1.1 Campus Management Systems

A literature survey of the Hostel Gate Pass Management System would involve reviewing existing academic papers, articles, and other relevant literature that discuss various aspects of such systems. Here's a general outline of how you might conduct a literature survey on this topic.

Identifying Relevant Keywords: Begin by identifying keywords related to the hostel gate pass management system, such as "hostel management", "gate pass system", "access control", "visitor management", etc.

Search Academic Databases: Use academic databases like PubMed, IEEE Xplore, ACM Digital Library, Google Scholar, and others to search for relevant research papers, conference proceedings, and articles.

Data security is another critical consideration in assessing the status of cold storages within HGMS. Given the sensitive nature of the information stored within these facilities, ensuring robust security measures is imperative. This includes implementing encryption protocols, access controls, intrusion detection systems, and regular security audits to mitigate the risk of unauthorized access, data breaches, or tampering.

Moreover, compliance with privacy regulations and legal frameworks governing data storage and handling is essential. HGMS must adhere to strict guidelines to protect individual privacy rights while fulfilling its mandate to identify and mitigate security threats. This requires ongoing monitoring and updates to ensure alignment with evolving regulatory requirements.

2.1.2 FRONT END-DREAMWEAVER

Adobe Dreamweaver CC is a web design and development application that uses both a visual design surface known as Live View and a code editor with standard features such as syntax highlighting, code completion, and code collapsing as well as more advanced features such as real-time syntax checking and code introspection for generating code hints to assist the user in writing code. Combined with an array of site management tools, Dreamweaver allows for its users design, code and manage websites, as well as mobile content

2.1.3 DREAMWEAVER

Dreamweaver is an Integrated Development Environment (IDE) tool. You can live preview of changes for the frontend. Dreamweaver is positioned as a versatile web design and development tool that enables visualization of web content while coding.

2.1.4 REQUIREMENT ANALYSIS:

Understand the needs and requirements of the hostel regarding gate pass management. This involves identifying stakeholders, gathering user stories, and defining functional and non-functional requirements.

2.1.5 SYSTEM DESIGN:

- **Architecture**: Determine the overall architecture of the system, including client-server architecture, database design, and integration with other systems if applicable.
- **User Interface Design**: Design intuitive and user-friendly interfaces for different user roles such as administrators, hostel staff, and residents.
- **Database Design**: Create a database schema to store information related to users, gate passes, access logs, etc.
- **Workflow Design**: Define the workflow for gate pass issuance, approval, and monitoring within the system.

2.1.6 DEVELOPMENT SYSTEM

- **Frontend Development**: Implement the user interfaces using appropriate web technologies such as HTML, CSS, and JavaScript, ensuring responsiveness and accessibility.

- **Backend Development**: Develop the server-side logic using a suitable programming language and framework (e.g., Node.js with Express.js) to handle requests, process data, and interact with the database.

- **Integration**: Integrate the frontend and backend components to create a cohesive system. Also, integrate with any external systems or APIs as needed.

- **Testing:** Conduct unit testing, integration testing, and system testing to ensure the reliability, security, and performance of the system.

2.1.7 APPLICATION OF HGMS SYSTEM IN MNC COMPANIES

Implementing a Hostel Gate Pass Management System in Multi-National Companies (MNCs) can enhance security, streamline operations, and improve overall efficiency. Here are several ways such a system could be applied in MNCs.

- **Employee Accommodation Management**: MNCs often provide accommodation facilities for employees who relocate for work. A Hostel Gate Pass Management System can streamline the process of allocating, monitoring, and managing accommodation for employees, ensuring smooth check-in/check-out procedures and accurate record-keeping.

- **Visitor Management**: MNCs frequently host clients, partners, and other visitors. Implementing a Hostel Gate Pass Management System enables efficient registration, approval, and monitoring of visitors entering the company's accommodation premises. It helps ensure that only authorized individuals have access and provides a record of visitor activity for security purposes.

- **Security and Access Control**: Enhanced security measures are crucial for MNCs, especially in accommodation facilities where employees reside. A Gate Pass Management System can integrate with access control systems to regulate entry and exit, monitor guest movements, and enforce security policies. It can also generate alerts for any unauthorized access attempts or security breaches.

Overall, implementing a Hostel Gate Pass Management System in MNCs contributes to a safer, more secure, and well-organized environment for employees and visitors while optimizing operational efficiency and compliance with regulatory requirements.

2.1.8 LANGUAGE SUPPORT

EXPRESS JS is free software released under the EXPRESS License, EXPRESS JS can be deployed on most webservers and also as a standalone shell on almost every operating system and platform, free of charge. EXPRESS JS variables, data types and programming constructs variables names start with $and can include characters, letters, numbers, and, No other special characters are permitted, Can’t start with a number.

In this project we have designed and implemented Raspberry Pi which works as a sensor node for the fruit and vegetable storage house as well as central base station is connected to cloud where MySQL open-source database server to support data storage functionalities. The sensor values are stored in the cloud and sent to the base station by connecting to database using its IP address. Thus aggregated several inputs as temperature, humidity and averages it to produce single consolidated output based on which the future decisions could be made. facilitate user interaction and connect through IoT based system that is station/gateway and the internet.

HOSTEL GATE PASS MANAGEMENT SYSTEM FIGURE

Fig 2.1.8.1 Recruitment Process flow chart

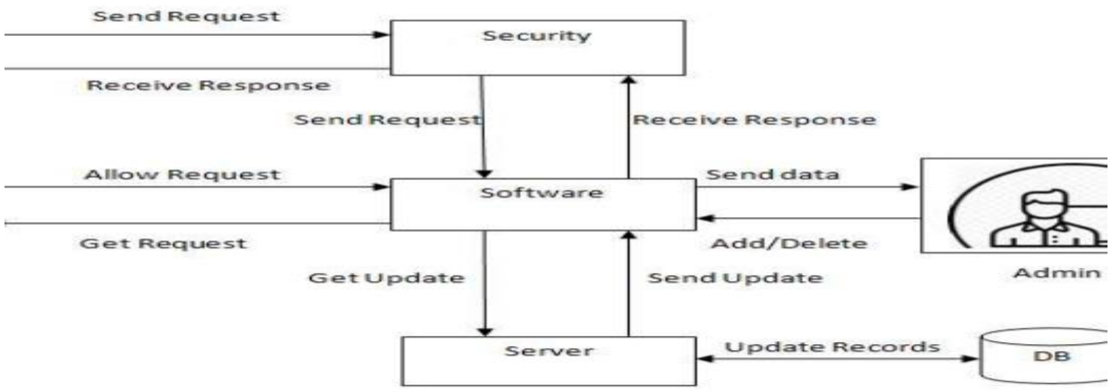


Fig 2.1.8.2 Context Level DFD

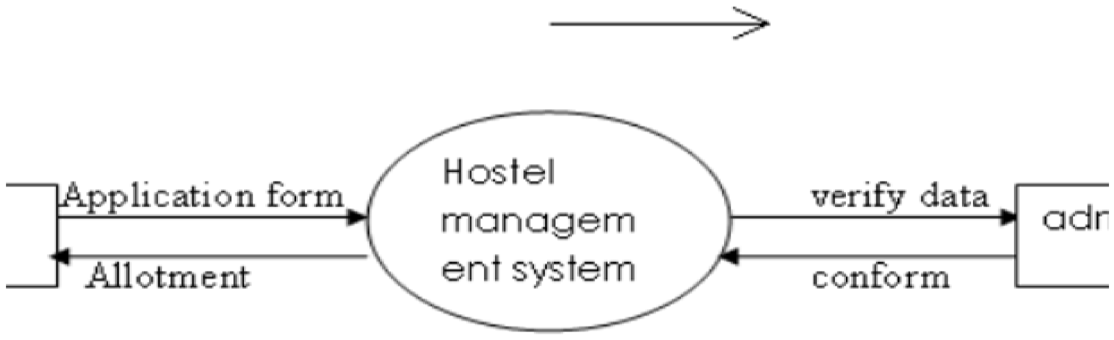


Fig 2.1.8.3 Student Module

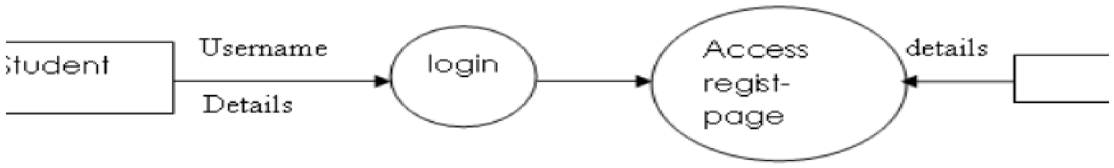


Fig 2.1.8.4 Registration Process

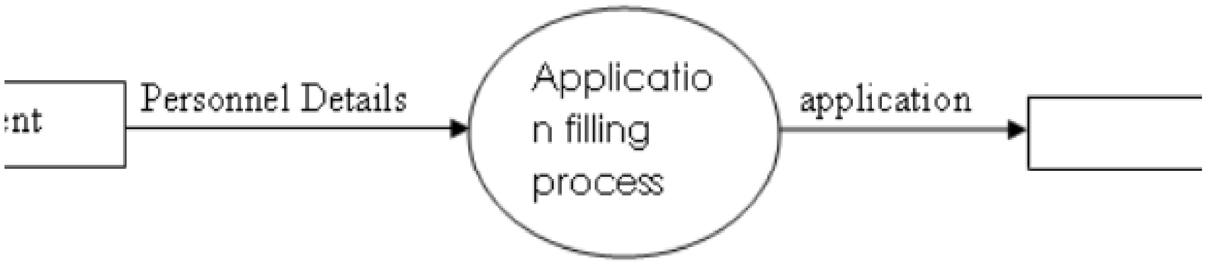
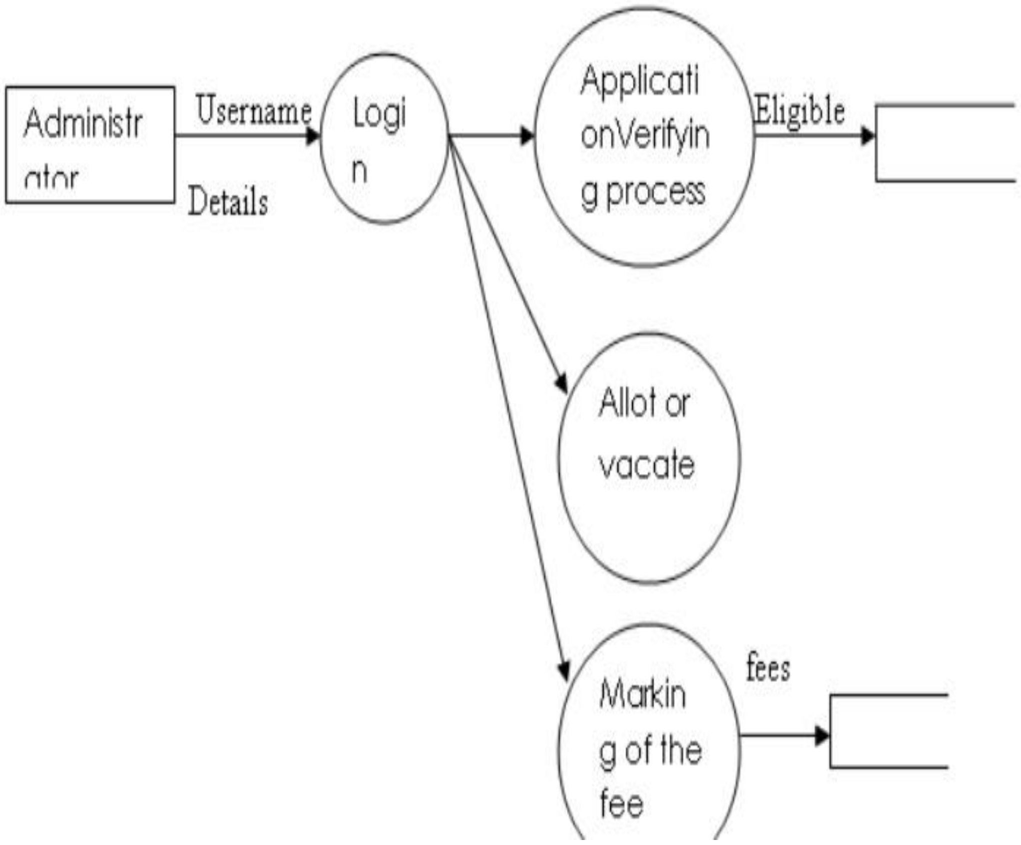


Fig 2.1.8.5 Administrator Module



CHAPTER 3

PROJECT DESCRIPTION

3.1 PROPOSED SYSTEM

The proposed system in the HGMS aims to revolutionize talent acquisition through innovative features, advanced algorithms, and seamless integration for enhanced recruitment outcomes.

3.1.1 About Proposed System

Our proposed system manages the entry and exit of the students on short term and long term leaves. Gate Pass Management System is an efficient system to keep track of each student's entry and exit. Management System is available for use and it is easy to access and manage.

It will simplifies the task and reduce the paper work. Our proposed system provides complete road map for students take one day leave and long leaves formalities. Also our proposed system shows the hierarchy of approval for students.

This project associated with the website where it is constructed with the modules including stock management, monitoring of units and additional marketing module. The overall process is monitored, stored, and marketed through this website. It associates with the three persons, admin, farmer, and customers. The admin has associated with all the three modules. The process at software level occurs parallel with the above IoT process. The admin is responsible for including stock details in the website, creating account of the farmers and maintaining of the stocks.

This proposed system aims to streamline the process of managing gate passes for hostel residents, staff, and security and accountability. Additional features and functionalities can be added based on specific requirements and preferences.

- **User Registration:**
    - Hostel residents, staff members, and visitors need to register with the system to use its features.

- Each user will have a unique identification such as an ID number or username.

- **Access Levels:**
    - Different access levels will be assigned to users based on their roles (e.g., administrator, resident, staff, visitor).
    - Administrators will have full access to manage the system, while residents may have limited access to request gate passes.

- **Gate Pass Request:**
    - Residents or visitors who want to enter or exit the hostel premises outside of regular hours or without their usual ID will request a gate pass.
    - They will provide details such as reason for visit, date, time, duration, and any accompanying guests.

- **Approval Process:**
    - Gate pass requests will be sent to designated administrators or staff members for approval.
    - Administrators can review and approve/deny requests based on predefined criteria (e.g., availability of staff, purpose of visit, security concerns).

3.1.2 ARCHITECTURE OF PROPOSED SYSTEM

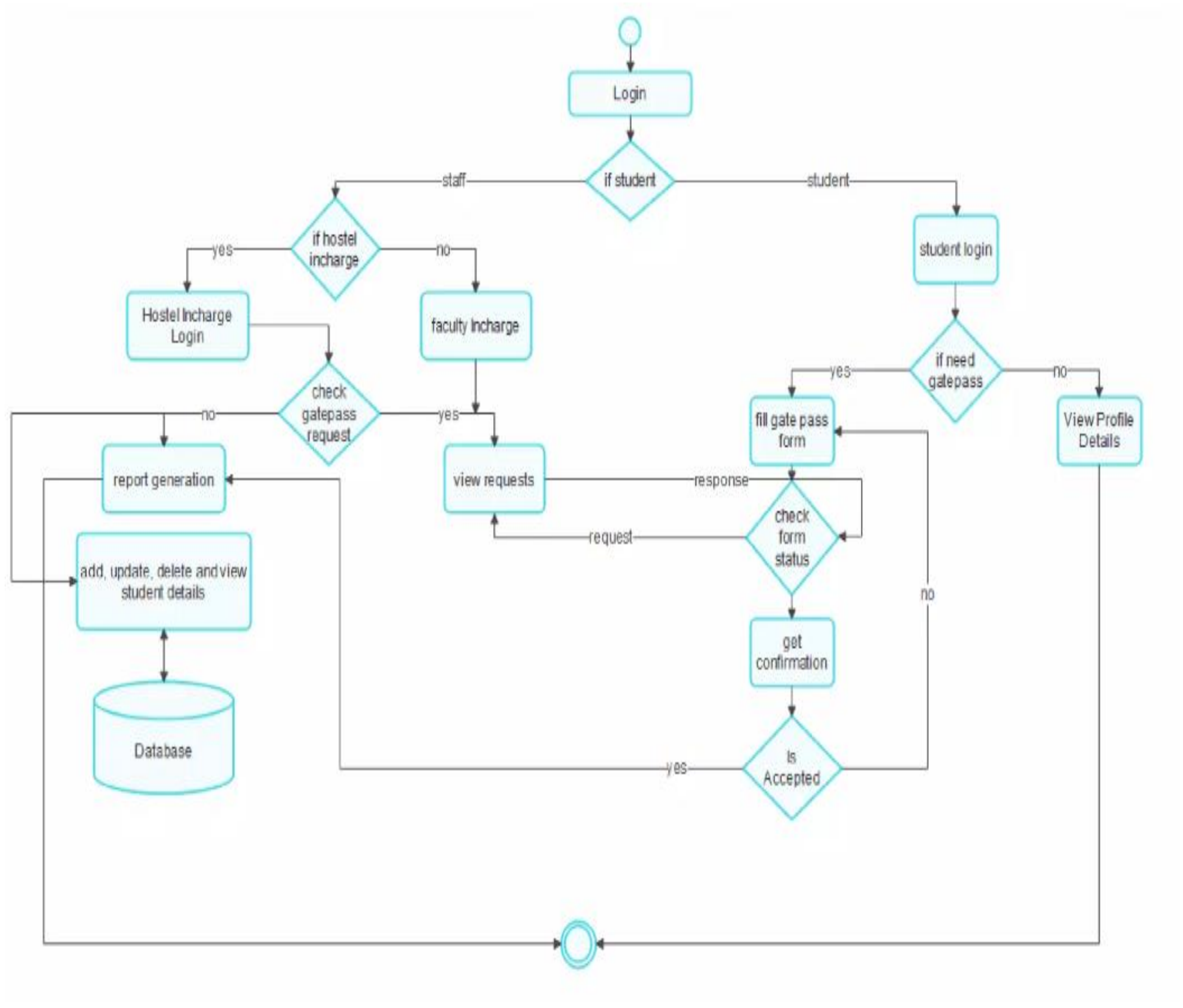


Fig 3.1.2.1 User Requirement process

CHAPTER 4

HARDWARE COMPONENTS

4.1 HARDWARE REQUIREMENTS

- SERVERS
- STORAGE
- NETWORKING INFARASTRUCTURE
- WORKSTATION
- BACKUPN AND RECOVERY

4.1.1 SERVERS:

High-performance servers are essential for hosting the HGMS software and handling the processing load generated by candidate data, algorithms, and analytics. These servers should have sufficient processing power, memory, and storage capacity to support concurrent user access and large-scale data processing.

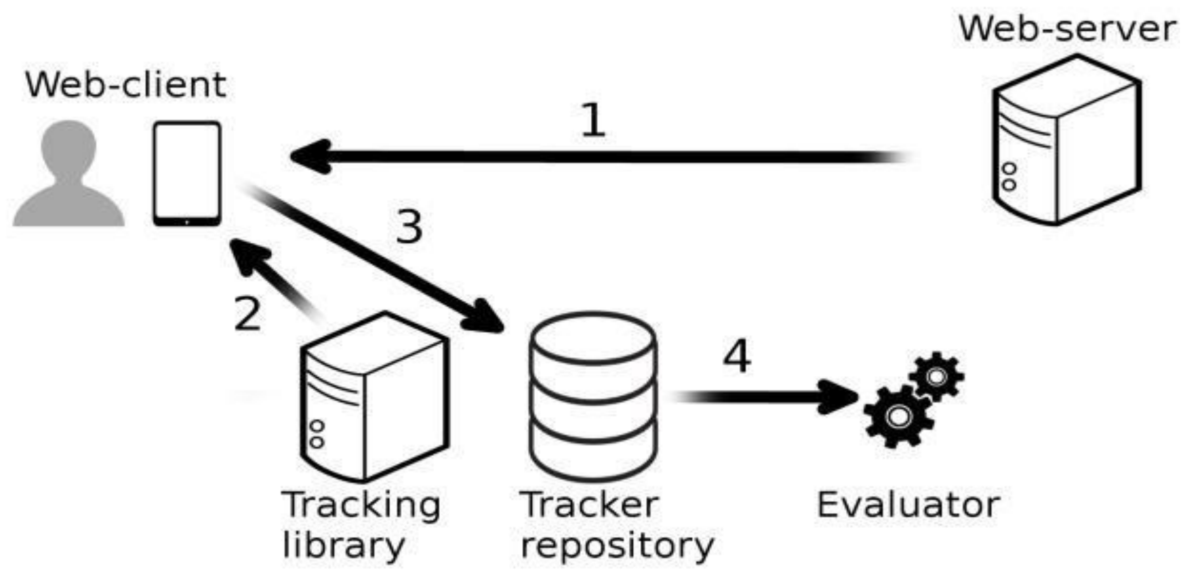


Fig 4.1.1.1 Servers processing model

4.1.2 STORAGE SYSTEM:

Robust storage systems are required to store vast amounts of candidate data, resumes, applications, and historical records. This includes both primary storage for active data and archival storage for historical data, backups, and audit trails. Storage solutions should be scalable, fault-tolerant, and capable of handling data growth over time.

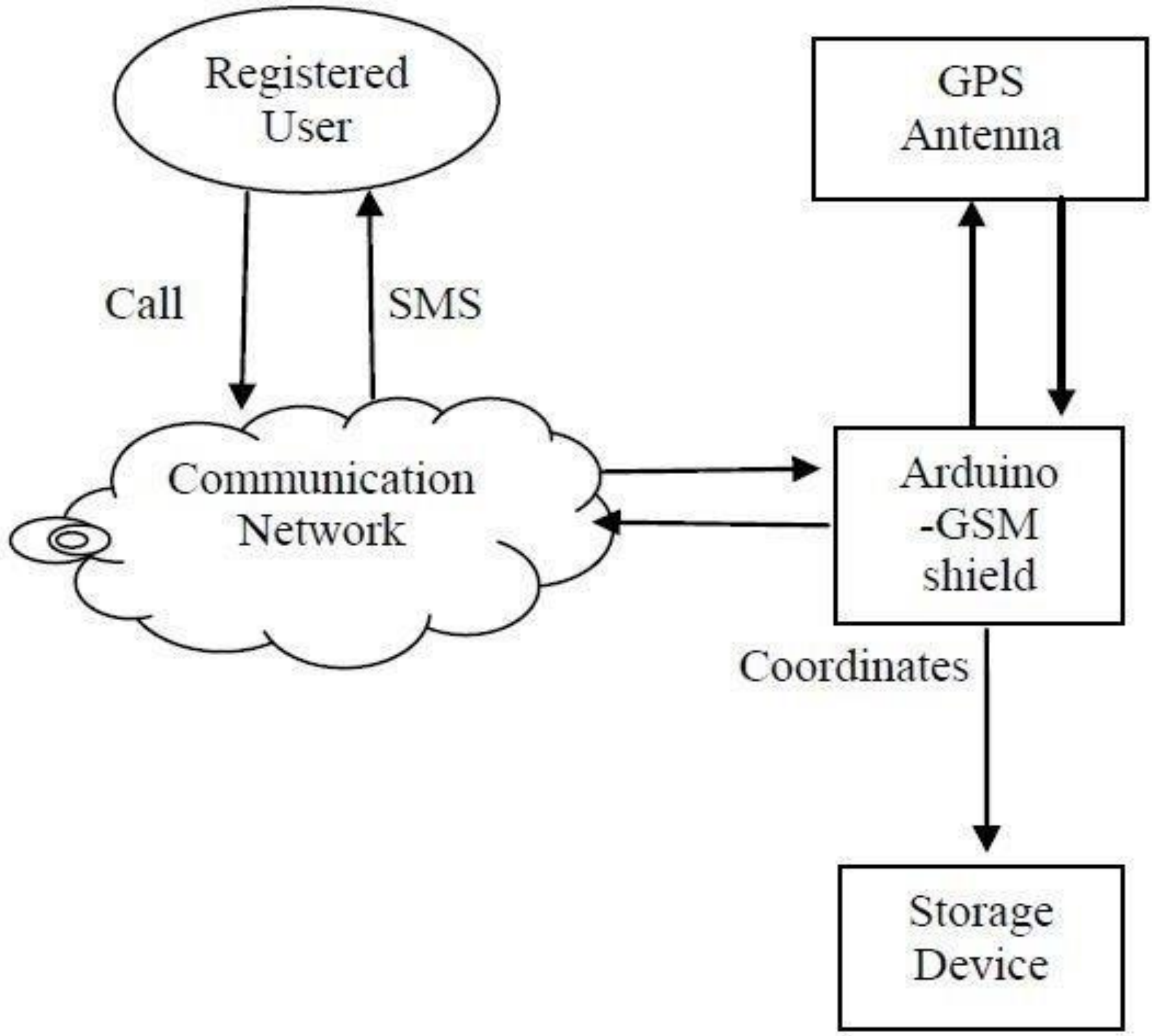


Fig 4.1.2.1 Data Storage process

4.1.3 NETWORK INFRASTRUCTURE:

A reliable and high-speed networking infrastructure is essential for facilitating communication between servers, clients, and external systems. This includes switches, routers, firewalls, and network security appliances to ensure data integrity, confidentiality, and availability.

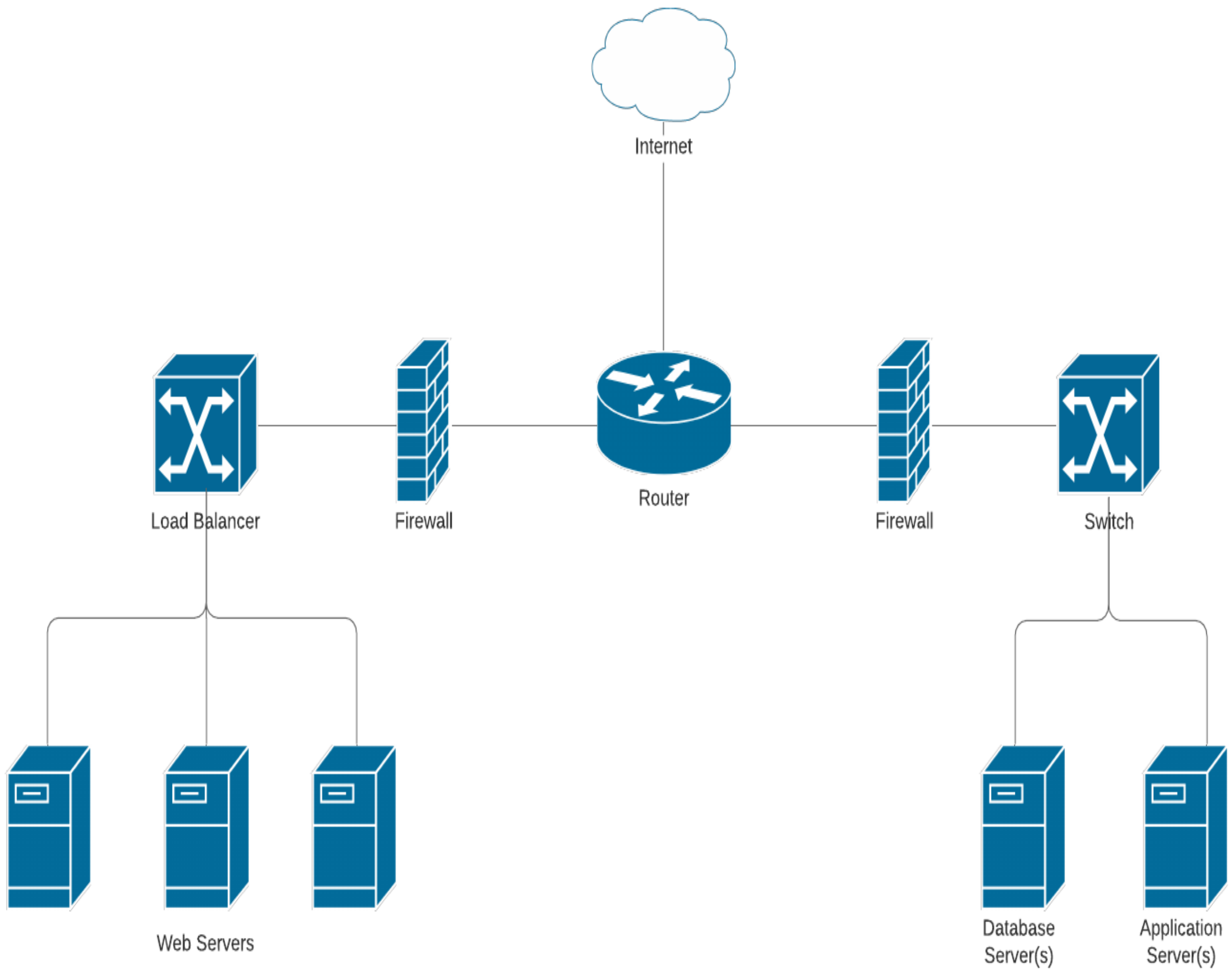
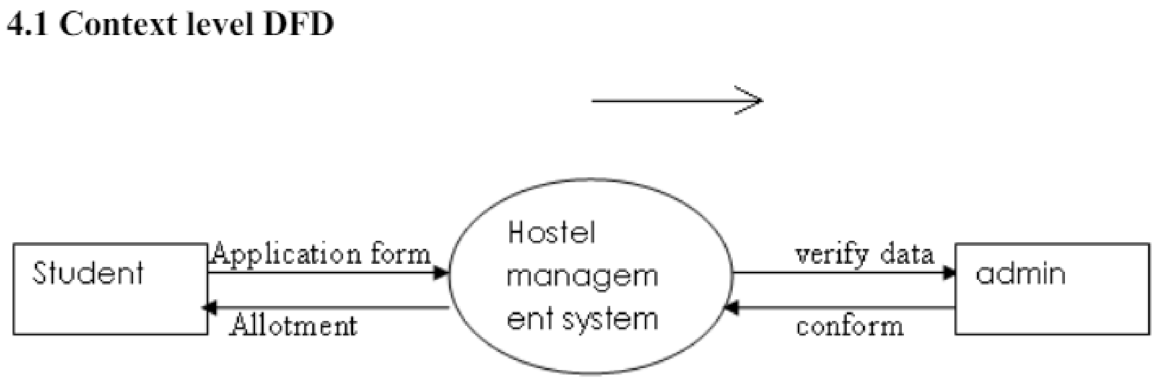


Fig 4.1.3.1 Required networking Infrastructure

4.1.4 WORKSTATIONS:

Workstations for recruiters, HR personnel, and administrators should be equipped with sufficient processing power, memory, and network connectivity to access and interact with the HGMS software efficiently. This includes desktop computers, laptops, or thin clients depending on user requirements and mobility needs.

Fig 4.1.4.1 workstation of HGMS



4.1.5 BACKUP AND DISASTER RECOVERY:

Backup systems and disaster recovery solutions are essential for protecting HGMS data against loss, corruption, or unauthorized access. This includes redundant storage, backup servers, and off-site replication to ensure data availability and continuity in the event of hardware failure or disaster.

Laptops play a vital role in developing and implementing Applicant Tracking Systems (HGMS). Their portability and computing capabilities empower developers to write, test, and optimize algorithms for efficient candidate management. With laptops, teams collaborate seamlessly, sharing code and insights to enhance HGMS functionality. Developers utilize the versatility of laptops to integrate HGMS seamlessly into existing workflows, ensuring a smooth transition and improved recruitment processes. Laptops also facilitate real-time data analysis, enabling recruiters to make informed decisions and streamline hiring procedures. Overall, laptops serve as indispensable tools in the creation and utilization of HGMS, revolutionizing the way organizations manage their talent acquisition efforts

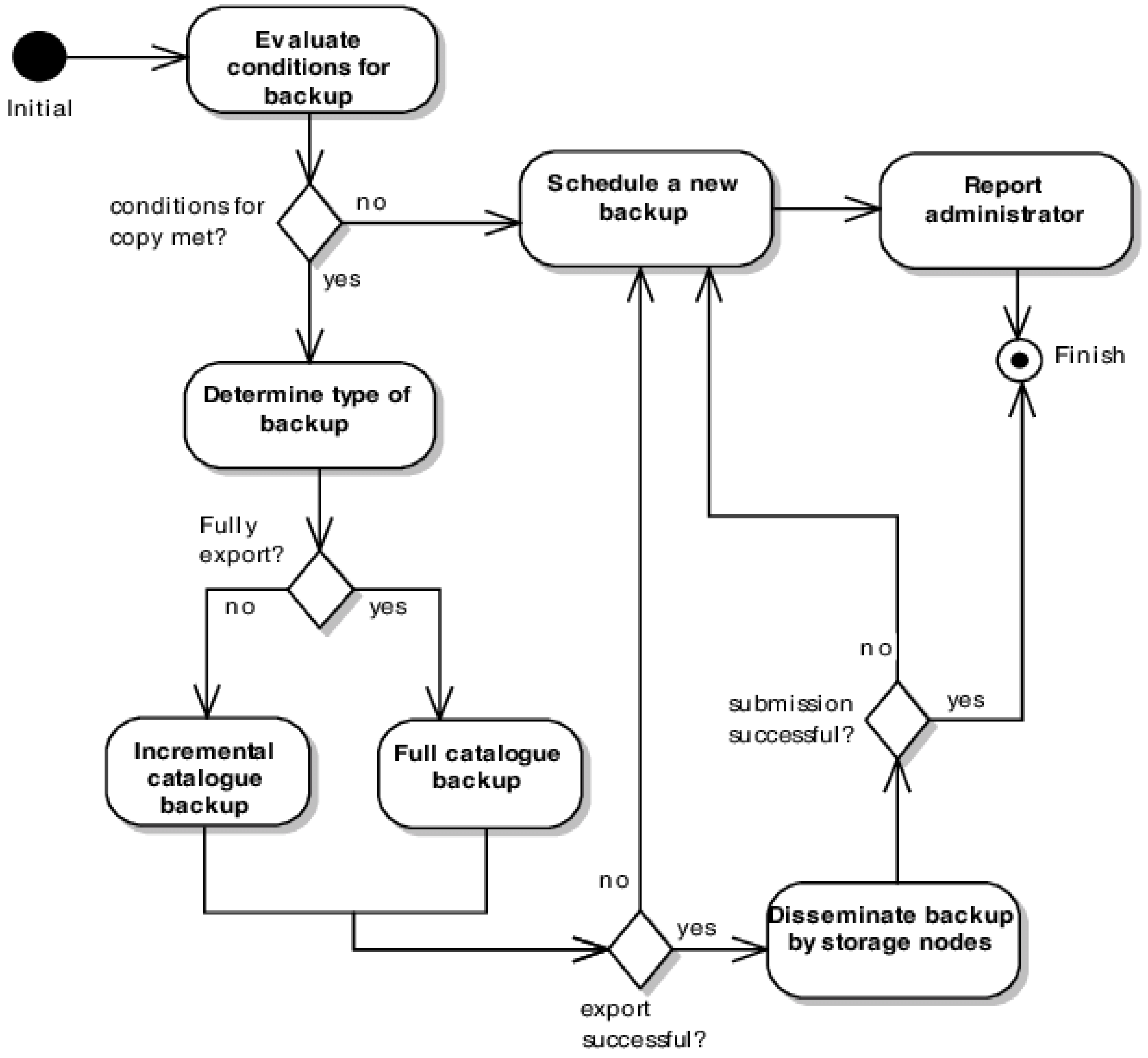


Fig 4.1.5.1 Backup and Disaster Recovery

CHAPTER 5

SOFTWARE COMPONENTS

5.1 SOFTWARE REQUIREMENT

- NODEJS
- JAVASCRIPT
- NODE HTTP SERVER
- MONGODB
- EXPRESS
- REACT
- BOOTSTRAP

5.1.1 NODEJS

Node.js is known for its asynchronous, event-driven architecture, which allows it to handle a large number of concurrent connections efficiently. This makes it particularly well-suited for applications that require high scalability, such as real-time web applications or streaming services.

5.1.2 NODE HTTP SERVER

Node.js HTTP servers are lightweight and efficient, making them suitable for handling a large number of concurrent connections with minimal resource consumption. Node.js uses an event-driven, non-blocking I/O model, allowing it to handle multiple incoming requests concurrently without blocking the execution of other tasks. This makes Node.js servers highly responsive and scalable. Creating an HTTP server in Node.js is straightforward, requiring only a few lines of code. This simplicity makes it easy for developers to get started with building web servers and APIs.

5.1.3 MONGODB

MongoDB is a NoSQL database that uses a flexible schema, allowing you to store heterogeneous data with varying structures. This flexibility is well-suited for dynamic web applications where data models evolve over time.

MongoDB stores data in JSON-like documents, which closely align with JavaScript objects. This makes it easy to work with MongoDB data in Node.js applications, as there's no need for complex mapping between database records and application objects.

5.1.4 EXPRESS

Express.js is a minimalist web framework for Node.js that provides a robust set of features for building web applications without imposing strict conventions. This allows developers to have greater flexibility and control over the architecture and design of their applications. Express.js features a powerful middleware system that allows you to extend and modify the request-response cycle easily. Middleware functions can handle tasks such as authentication, logging, error handling, and request processing, making it easy to add functionality to your application in a modular and reusable way.

Express.js provides a simple and intuitive routing system that allows you to define routes for handling HTTP requests based on URL paths and HTTP methods. This makes it easy to organize your application's code into logical units and handle different types of requests efficiently.

5.1.5 REACT

React utilizes a component-based architecture, allowing developers to create modular, reusable UI components. This approach promotes code reusability, maintainability, and scalability, making it easier to manage complex user interfaces.

5.1.6 BOOTSTRAP

The Bootstrap is a web framework that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

5.1.7 JAVASCRIPT

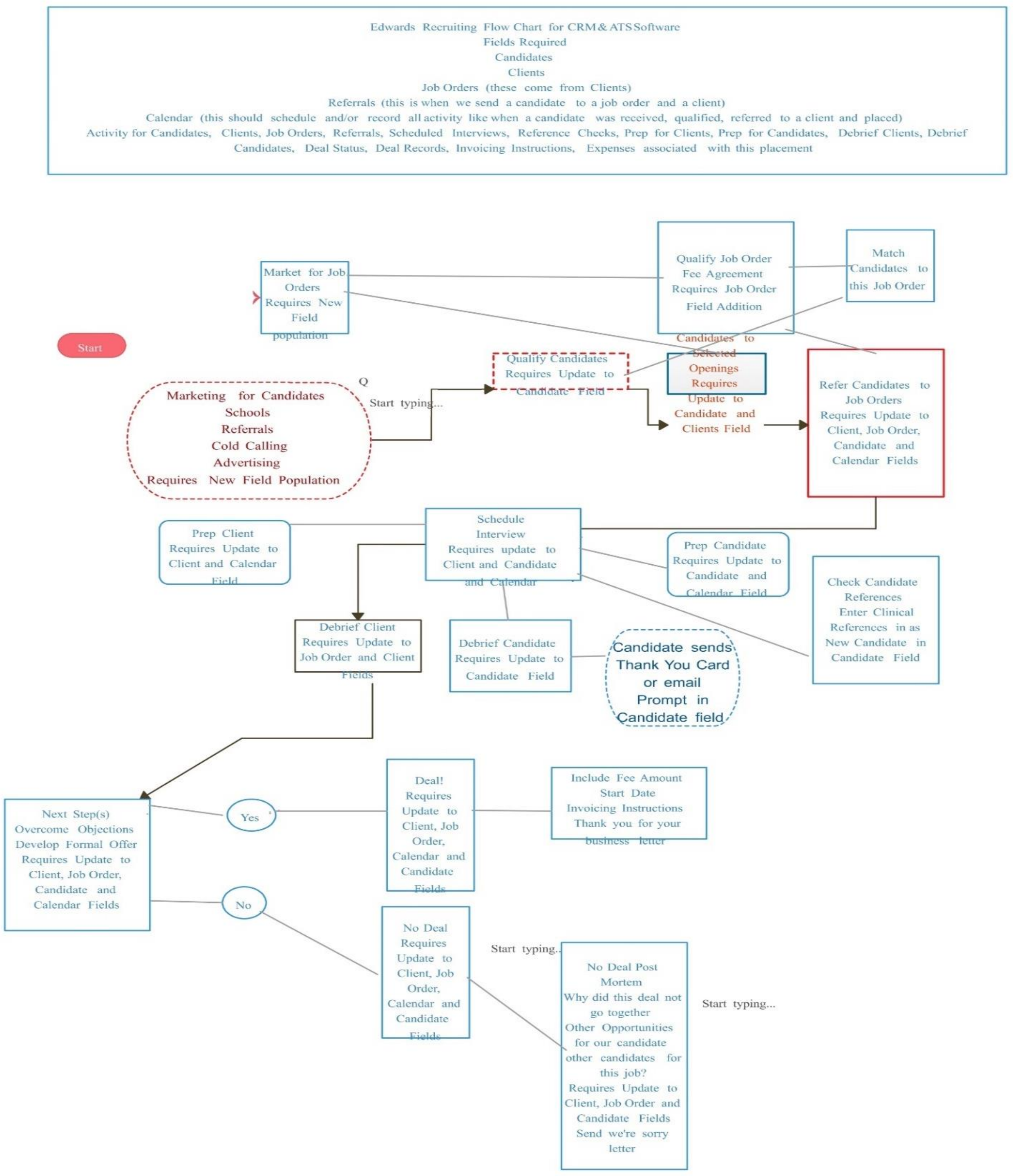
JavaScript is one of the most widely used programming languages, supported by all major web browsers and platforms. Its ubiquity makes it an essential skill for web developers and ensures broad compatibility for web applications.

JavaScript enables dynamic and interactive web experiences by allowing developers to manipulate the Document Object Model (DOM) of web pages in real-time. This interactivity enhances user engagement and enables features like form validation, animations, and interactive widgets.

CHAPTER 6

WORKFLOW DIAGRAMS

6.1 WORKFLOW DIAGRAM



Edwards Recruiting Flow Chart for CRM & ATS Software
Fields Required
Candidates
Clients
Job Orders (these come from Clients)
Referrals (this is when we send a candidate to a job order and a client)
Calendar (this should schedule and/or record all activity like when a candidate was received, qualified, referred to a client and placed)
Activity for Candidates, Clients, Job Orders, Referrals, Scheduled Interviews, Reference Checks, Prep for Clients, Prep for Candidates, Debrief Clients, Debrief Candidates, Deal Status, Deal Records, Invoicing Instructions, Expenses associated with this placement
Market for Job Orders Requires New Field population
Qualify Job Order Fee Agreement Requires Job Order Field Addition
Match Candidates to this Job Order
Start
Q
Start typing...
Qualify Candidates Requires Update to Candidate Field
Candidates to Selected Openings Requires Update to Candidate and Clients Field
Refer Candidates to Job Orders Requires Update to Client, Job Order, Candidate and Calendar Fields
Marketing for Candidates Schools Referrals Cold Calling Advertising Requires New Field Population
Prep Client Requires Update to Client and Calendar Field
Schedule Interview Requires update to Client and Candidate and Calendar
Prep Candidate Requires Update to Candidate and Calendar Field
Check Candidate References Enter Clinical References in as New Candidate in Candidate Field
Debrief Client Requires Update to Job Order and Client Fields
Debrief Candidate Requires Update to Candidate Field
Candidate sends Thank You Card or email Prompt in Candidate field
Next Step(s) Overcome Objections Develop Formal Offer Requires Update to Client, Job Order, Candidate and Calendar Fields
Yes
No
Deal! Requires Update to Client, Job Order, Calendar and Candidate Fields
Include Fee Amount Start Date Invoicing Instructions Thank you for your business letter
No Deal Requires Update to Client, Job Order, Calendar and Candidate Fields
Start typing...
No Deal Post Mortem Why did this deal not go together Other Opportunities for our candidate other candidates for this job? Requires Update to Client, Job Order and Candidate Fields Send we're sorry letter
Start typing...

Fig.6.1.1 Flow Diagram

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

To conclude the description about the project The project, developed using NODE and NoSQL is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement. The expanded functionality of today's software requires an appropriate approach towards software development. This hostel management software is designed for people who want to manage various activities in the hostel. For the past few years the number of educational institutions are increasing rapidly. There by the number of hostels are also increasing for the accommodation of the students studying in this institution. And software's are not usually used in this context. This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually

7.2 FUTURE ENHANCEMENT

Future enhancements for HOSTEL GATE PASS MANGEMENT systems may include the integration of artificial intelligence (AI) and machine learning (ML) algorithms to Face and matching processes, enhancing efficiency and accuracy. Additionally, Face Detection System (FDS) capabilities can improve parsing and communication. Furthermore, the integration of virtual reality (VR) and augmented reality (AR) technologies may revolutionize Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented.

APPENDIX 1

SOURCE CODE

REGISTRATION

```
import React from "react";
import "../styles/RegiserStyles.css";
import { Form, Input, message } from "antd";
import axios from "axios";
import { Link, useNavigate } from "react-router-dom";
const Register = () => {
  const navigate = useNavigate();

  //form handler
  const onfinishHandler = async (values) => {
    try {
      const res = await axios.post("/api/v1/user/register", values);
      if (res.data.success) {
        message.success("Register Successfully!");
        navigate("/login");
      } else {
        message.error(res.data.message);
      }
    } catch (error) {
      console.log(error);
      message.error("Something Went Wrong");
    }
  };
  return (
  <>
    <div className="form-container ">
      <Form
        layout="vertical"
        onFinish={onfinishHandler}
        className="register-form"
      >
        <h3 className="text-center">Register From</h3>
        <Form.Item label="Name" name="name">
          <Input type="text" required />
        </Form.Item>
        <Form.Item label="Email" name="email">
          <Input type="email" required />
        </Form.Item>
        <Form.Item label="Password" name="password">
          <Input type="password" required />
        </Form.Item>
        <Link to="/login" className="m-2">
          Already user login here
        </Link>
        <button className="btn btn-primary" type="submit">
          Register
        </button>
      </Form>
    </div>
  </>
);
```

```
};

export default Register;
```

LOGIN

```
import React from "react";
import "../styles/RegiserStyles.css";
import { Form, Input, message } from "antd";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";

const Login = () => {
  const navigate = useNavigate();
  //form handler
  const onfinishHandler = async (values) => {
    try {
      const res = await axios.post("/api/v1/user/login", values);
      if (res.data.success) {
        localStorage.setItem("token", res.data.token);
        message.success("Login Successfully");
        navigate("/");
      } else {
        message.error(res.data.message);
      }
    } catch (error) {
      console.log(error);
      message.error("something went wrong");
    }
  };
  return (
    <div className="form-container ">
      <Form
        layout="vertical"
        onFinish={onfinishHandler}
        className="register-form">
        <h3 className="text-center">Login From</h3>
        <Form.Item label="Email" name="email">
          <Input type="email" required />
        </Form.Item>
        <Form.Item label="Password" name="password">
          <Input type="password" required />
        </Form.Item>
        <Link to="/register" className="m-2">
          Not a user Register here
        </Link>
        <button className="btn btn-primary" type="submit">
          Login
        </button>
      </Form>
    </div>  );
};

export default Login;
```

```
# BACKEND CODE
const colors = require("colors");
const moragan = require("morgan");
const dotenv = require("dotenv");
const connectDB = require("./config/db");
const path = require("path");

//dotenv conig
dotenv.config();

//mongodb connection
connectDB();

//rest obejct
const app = express();

//middlewares
app.use(express.json());
app.use(moragan("dev"));

//routes
app.use("/api/v1/user", require("./routes/userRoutes"));
app.use("/api/v1/admin", require("./routes/adminRoutes"));
app.use("/api/v1/doctor", require("./routes/WardenRoutes"));

//static files
app.use(express.static(path.join(__dirname, "./client/build")));
app.get("*", function (req, res) {
  res.sendFile(path.join(__dirname, "./client/build/index.html"));
});

//port
const port = process.env.PORT || 8080;
//listen port
app.listen(port, () => {
  console.log(
    `Server Running in ${process.env.NODE_MODE} Mode on port
${process.env.PORT}`
      .bgCyan.white
  );
});
const express = require("express");
const {
  getAllUsersController,
  getAllWardenController,
  changeAccountStatusController,
} = require("../controllers/adminCtrl");
const authMiddleware = require("../middlewares/authMiddleware");

const router = express.Router();

//GET METHOD || USERS
router.get("/getAllUsers", authMiddleware, getAllUsersController);
```

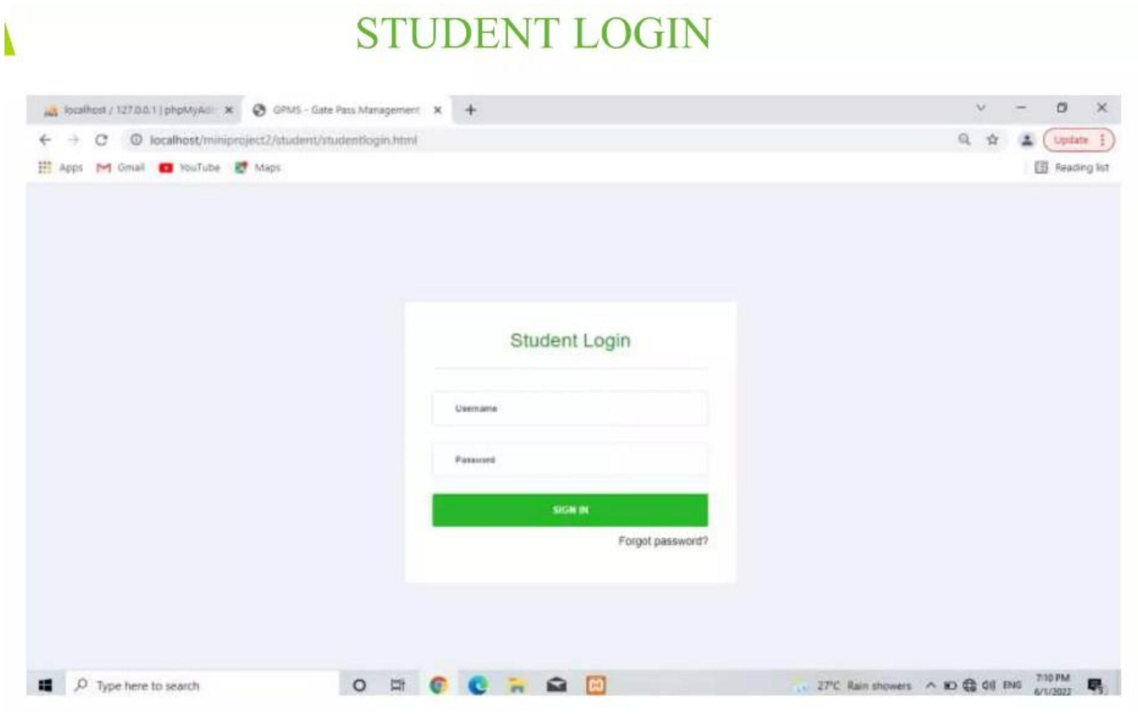
```
//GET METHOD || WARDEN || STUDENT
router.get("/getAllWarden", authMiddleware, getAllStudentController);

//POST ACCOUNT STATUS
router.post(
  "/changeAccountStatus",
  authMiddleware,
  changeAccountStatusController
);

module.exports = router;
const express = require("express");
const {
  loginController,
  registerController,
  authController,
  applyGATEPASSController,
  getAllNotificationController,
  deleteAllNotificationController,
  getAllHOSTELWARDENController,
  bookeAppointmnetController,
  bookingAvailabilityController,
  userAppointmentsController,
} = require("../controllers/userCtrl");
const authMiddleware = require("../middlewares/authMiddleware");
```

```
//router onject
const router = express.Router();

//routes
//LOGIN || POST
router.post("/login", loginController);

//REGISTER || POST
router.post("/register", registerController);

//Auth || POST
router.post("/getUserData", authMiddleware, authController);

//APply Warden || POST
router.post("/apply-doctor", authMiddleware, applyDoctorController);
//Notifiaction Warden || POST
router.post(
  "/get-all-notification",
  authMiddleware,
  getAllNotificationController
);
//Notifiaction  User || POST
router.post(
  "/delete-all-notification",
  authMiddleware,
  deleteAllNotificationController
);
```

```
//GET ALL WARDEN
router.get("/getAllWarden", authMiddleware, getAllWARDENController);
//BOOK APPOINTMENT
router.post("/book-appointment", authMiddleware, bookeAppointmnetController);
//Booking Avliability
router.post(
  "/booking-availbility",
  authMiddleware,
  bookingAvailabilityController
);
//Appointments List
router.get("/user-appointments", authMiddleware, userAppointmentsController);
module.exports = router;    node_text = []
    for node in G.nodes():
        adjacencies = list(G.adj[node])  # changes here
        node_adjacencies.append(len(adjacencies))
        node_text.append(f"{node}<br># of connections: {len(adjacencies)}")
    node_trace.marker.color = node_adjacencies
    node_trace.text = node_text
    # Create the figure
    fig = go.Figure(
        data=[edge_trace, node_trace],
        layout=go.Layout(
            title=title,
            titlefont_size=16,
            showlegend=False,
            hovermode="closest",
            margin=dict(b=20, l=5, r=5, t=40),
            xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
            yaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
        ),
    )
    # Show the figure
    st.plotly_chart(fig)
def create_annotated_text(
    input_string: str, word_list: List[str], annotation: str, color_code: str
):
    tokens = nltk.word_tokenize(input_string)
    # Convert the list to a set for quick lookups
    word_set = set(word_list)
    # Initialize an empty list to hold the annotated text
    annotated_text = []
    for token in tokens:
        # Check if the token is in the set
        if token in word_set:
            # If it is, append a tuple with the token, annotation, and color code
            annotated_text.append((token, annotation, color_code))
        else:
            # If it's not, just append the token as a string
            annotated_text.append(token)
    return annotated_text;
})
};
```

APPENDIX 2

SCREENSHOTS



FigA2.1 Student Login

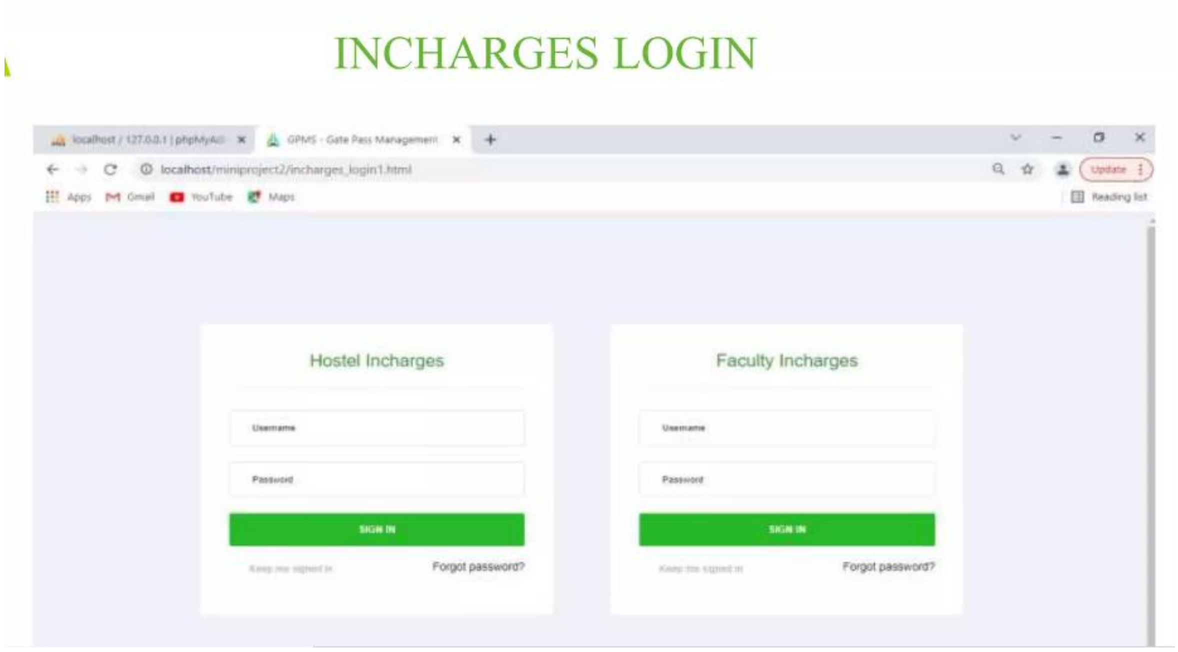
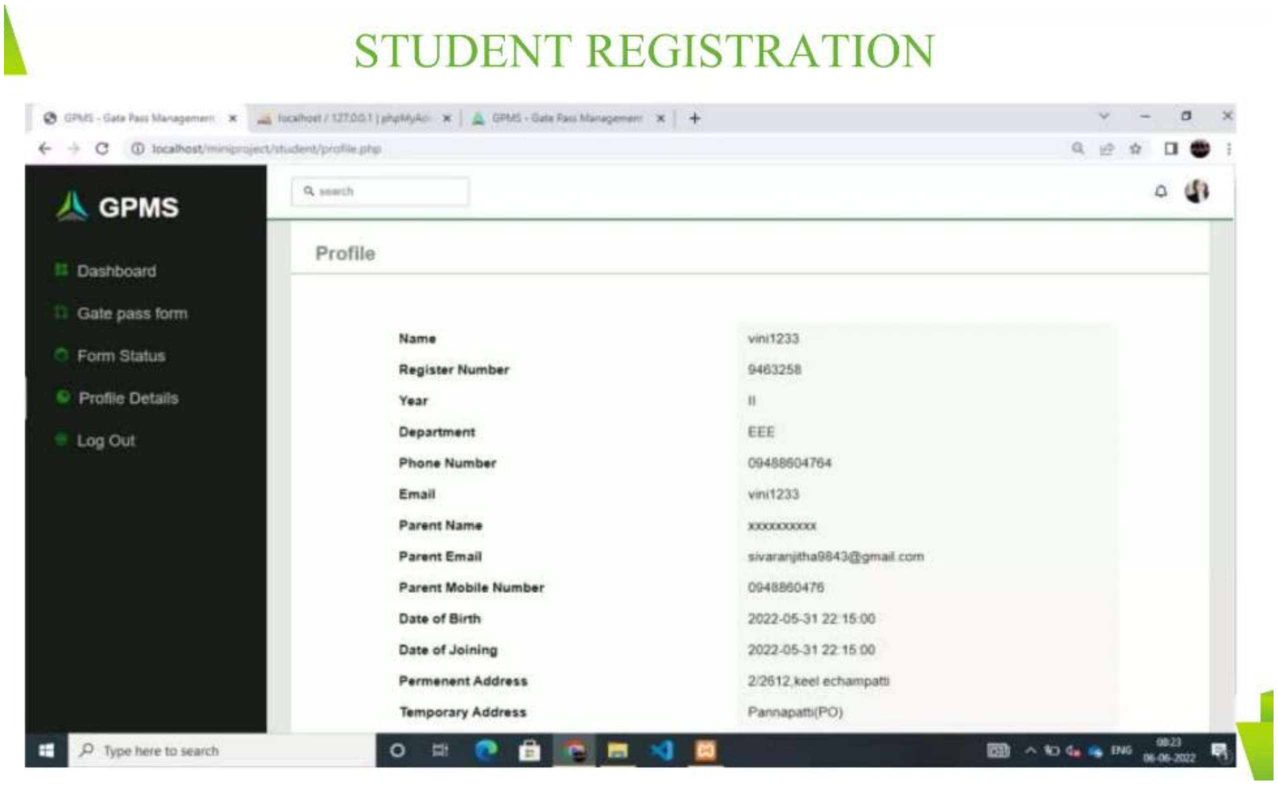


Fig A2.2 Warden Login

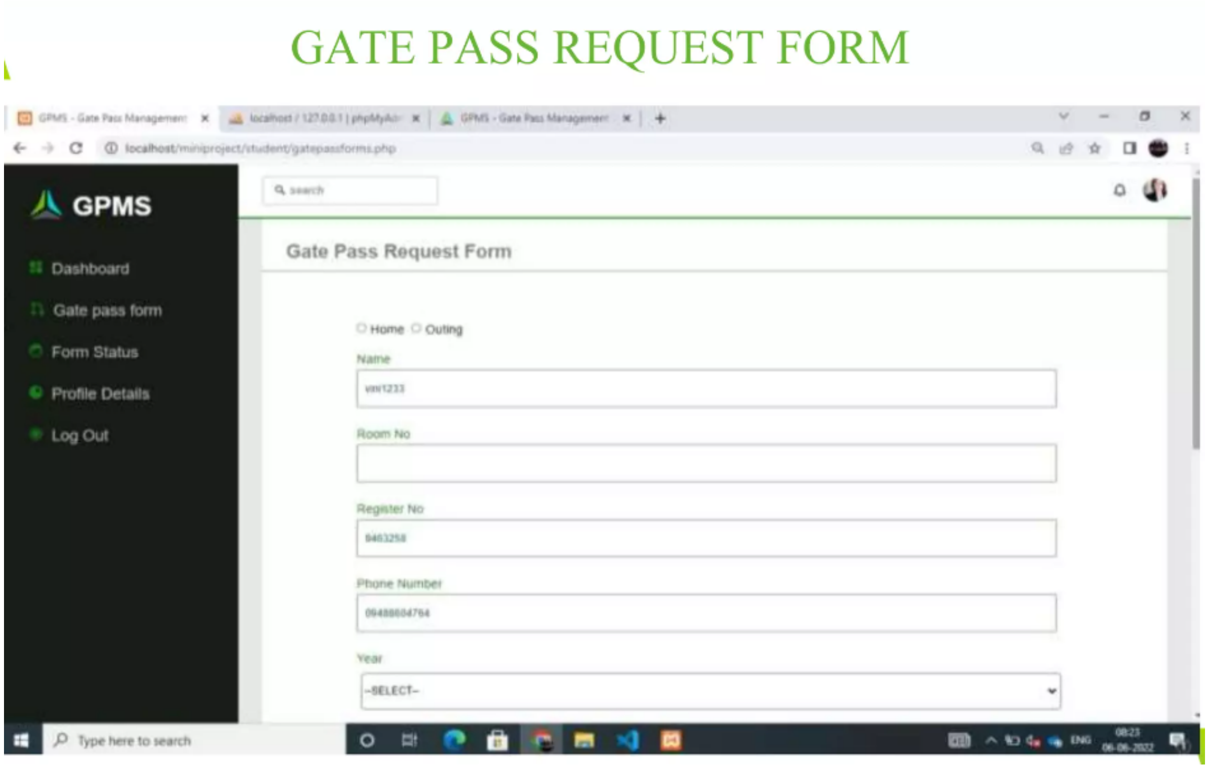

STUDENT REGISTRATION

localhost/miniproject/student/profile.php

Profile

Name	vini1233
Register Number	9463258
Year	II
Department	EEE
Phone Number	09488604764
Email	vini1233
Parent Name	xxxxxxxxxx
Parent Email	sivaranjitha9843@gmail.com
Parent Mobile Number	0948860476
Date of Birth	2022-05-31 22:15:00
Date of Joining	2022-05-31 22:15:00
Permenent Address	2/2612,keel echampatti
Temporary Address	Pannapatti(PO)

Fig A2.3 Student Registration


GATE PASS REQUEST FORM

localhost/miniproject/student/gatepassforms.php

Gate Pass Request Form

○ Home ○ Outing

Name: vini1233

Room No:

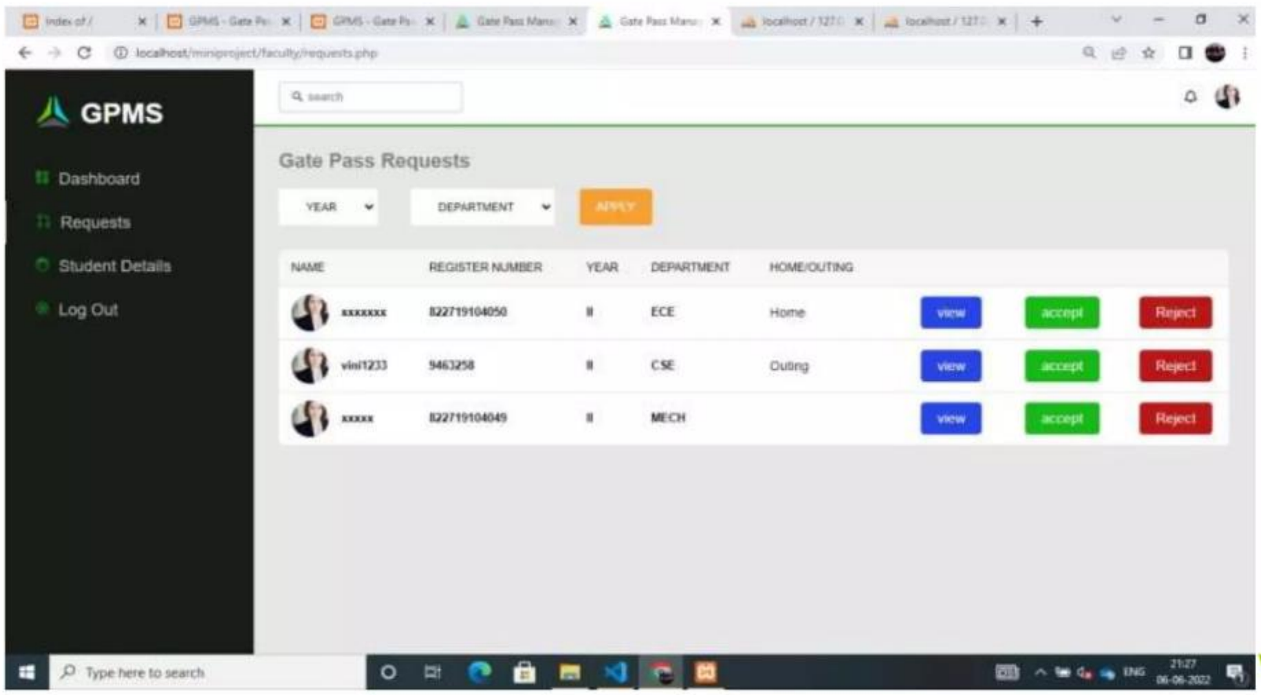
Register No: 9463258

Phone Number: 09488604764

Year: --SELECT--

Fig A2.4 Request form

GATE PASS APPROVED OR REJECTED STATUS



NAME	REGISTER NUMBER	YEAR	DEPARTMENT	HOME/OUTING			
xxxxxxx	822719104050	II	ECE	Home	view	accept	Reject
vini1233	9463258	II	CSE	Outing	view	accept	Reject
xxxxx	822719104049	II	MECH		view	accept	Reject

Fig A2.5 Pass Status

REFERENCE

1. J. R. Groff and P. N Weinberg. Complete Reference SQL Second Edition.K. Ayanlowo, O. Shoewu, S. O. Olatinwo, O. O. Ommitola and D. D. Babalola (2014). Journal of Science and Engineering Vol 5(1): Development of an Automated Hostel Facility Management System Retrieved from www.oricpub.com.

2. K.A. Muhammed Shaheer, A. Muhammed Shiras, R. Vinod Raj, G.V Prashobh (2009).Project Report on Hostel Management System.

3. S.R Ahmad, A.K Ghalib and S.A. Mahmood (2013). Indian Journal of Science (Vol 65 No.1): GIS Based Hostel Management System for Anna University.

4. W. N. H. B. W. JA'AFAR (2012). Hostel Management system. Html and Php from W3 schools. http//w3school.com en.wikipedia.org/wiki/PHP.

5. www.hotscripts.com/category/php/College Hostel Management Software by Initio (2010).

6. Loventis Booking System by Loventis Systems (2005). Indocon Hostel Management system. http://www.kassoftindia.com/Product/GeniusAcademic/hostelmgt.htm [accessed September 24, 2012].

PROGRAM OUTCOMES (POs)

PO 1	**Engineering knowledge**	Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
PO 2	**Problem analysis**	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design/development of solutions**	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigations of complex problems**	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer and society**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and sustainability**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual and team work**	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication**	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project management and finance**	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	**Life-long learning**	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Code & Name	**: C412 – (CS6811 – PROJECT)**
REGULATION	**: R2017**
YEAR / SEM	**: IV / VIII**

COURSE OUTCOMES

C412.1	**E**	**Estimate** the problem by applying acquired knowledge.
C412.2	**C**	**Develop** the executable project modules after considering risks
C412.3	**E**	**Choose** efficient tools for designing project modules.
C412.4	**C**	**Combine** all the modules through effective team work after efficient testing.
C412.5	**C**	**Elaborate** the completed task and compile the project report.

CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO 10**	**PO 11**	**PO12**	**PSO 1**	**PSO 2**
C412.1	3	2	2	2	2	2	-	2	3	2	2	3	3	3
C412.2	3	3	2	3	3	3	-	3	3	3	2	3	3	3
C412.3	3	3	3	3	3	2	2	3	3	3	2	3	2	3
C412.4	3	3	3	3	3	2	3	3	3	3	3	3	3	3
C412.5	3	3	3	3	3	2	3	3	3	3	3	3	3	3
C412	**3.00**	**2.80**	**2.60**	**2.80**	**2.80**	**2.20**	**2.67**	**2.80**	**3.00**	**2.80**	**2.40**	**3.00**	**2.80**	**3.00**