**TIMETABLE MANAGEMENT SYSTEM**

# **A MINI PROJECT REPORT**

**Submitted by**

| **KISHORE K** |  | **220701134** |
| --- | --- | --- |
| **MAGIZHAN SIVAKUMAR** |  | **220701154** |
|  |  |  |

In partial fulfillment for the award of the degree of **BACHELOR OF ENGINEERING**

**IN COMPUTER SCIENCE**

**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**



**THANDALAM**

**2024-2025**

# **BONAFIDE CERTIFICATE**

Certified that this project report “**ATTENDANCE MANAGEMENT SYSTEM**” is the bonafide work of “**KISHORE K (220701134), MAGIZHAN SIVAKUMAR(220701154)**” who carried out the project under my supervision.

# Submitted for the Practical Examination held on

**SIGNATURE SIGNATURE**

**Dr.R.SABITHA Ms.D.KALPANA**

**Professor and II Year Academic Head Assistant Professor (SG),**

**Computer Science and Engineering, Computer Science & Engineering,**

**Rajalakshmi Engineering College Rajalakshmi Engineering College**

**(Autonomous), (Autonomous),**

**Thandalam, Chennai - 602 105 Thandalam, Chennai**

INTERNAL EXAMINER EXTERNAL EXAMINER

**ABSTRACT**

The Time Table Management System (TTMS) for college administration offers a comprehensive solution to streamline and optimize the complex process of scheduling classes, resources, and faculty members within a college or university environment. This project aims to develop a user-friendly web-based platform that automates the generation, modification, and dissemination of timetables while considering various constraints and preferences.

Key functionalities of the system include:

1. Automated Timetable Generation: The system employs algorithms to automatically generate optimized timetables based on input parameters such as course schedules, faculty availability, classroom capacities, and student preferences.
2. Flexible Customization: Administrators can easily customize the generated timetables by making manual adjustments, accommodating special events, holidays, or unexpected changes in the academic calendar.
3. Resource Allocation: Efficient utilization of resources such as classrooms, laboratories, and equipment is ensured through intelligent allocation algorithms, minimizing conflicts and maximizing usage.
4. User Authentication and Access Control: Different user roles (administrators, faculty, and students) are implemented with appropriate access levels to ensure data security and privacy.
5. Communication and Notifications: The system facilitates seamless communication by sending automated notifications to stakeholders regarding timetable updates, class cancellations, or room changes.
6. Reporting and Analytics: Detailed reports and analytics provide insights into resource utilization, faculty workload distribution, and overall efficiency, aiding administrators in making informed decisions for future improvements.

# TABLE OF CONTENTS

1. **INTRODUCTION**
   1. INTRODUCTION
   2. OBJECTIVES
   3. MODULES

# SURVEY OF TECHNOLOGIES

* 1. SOFTWARE DESCRIPTION
  2. LANGUAGES
     1. MONGODB
     2. PYTHON

# REQUIREMENTS AND ANALYSIS

* 1. REQUIREMENT SPECIFICATION
  2. HARDWARE AND SOFTWARE REQUIREMENTS
  3. ARCHITECTURE DIAGRAM
  4. ER DIAGRAM

# PROGRAM CODE

1. **RESULTS AND DISCUSSION 6.CONCLUSION**

**7.REFERENCES**

# CHAPTER 1

* 1. **INTRODUCTION**

### In the dynamic landscape of higher education, effective time management stands as a cornerstone for ensuring the smooth functioning of academic institutions. Colleges and universities grapple with the intricate task of scheduling classes, allocating resources, and accommodating diverse faculty and student needs within constrained timeframes. Traditional manual methods of timetable management often lead to inefficiencies, scheduling conflicts, and administrative burdens.

### 

# **OBJECTIVES**

### Automate Timetable Generation: Develop algorithms and workflows to automate the generation of optimized timetables based on input parameters such as course offerings, faculty availability, classroom capacities, and student preferences.Enhance Flexibility and Customization: Implement features that allow administrators to easily customize timetables to accommodate special events, holidays, and unexpected changes in the academic calendar, while maintaining optimization principles.Optimize Resource Allocation: Design intelligent allocation algorithms to optimize the utilization of resources such as classrooms, laboratories

# MODULES

#### User Management Module

#### Timetable Generation Module

#### Record Management Module.

#### 

# CHAPTER-2

* 1. **SOFTWARE DESCRIPTION Visual studio Code:**

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

# LANGUAGES

1. **Python:**

- It is used for scripting the application's logic, managing database operations, and integrating different modules.

**2.Streamlit:**

* + A powerful Python library for creating interactive web applications with simple Python scripts, enabling rapid prototyping and deployment of data-driven applications with ease..

# **3.MongoDB:**

* + A scalable NoSQL database solution, offering high performance, flexibility, and seamless integration with modern applications..

# **CHAPTER-3 REQUIREMENT AND ANALYSIS**

* 1. **REQUIREMENT SPECIFICATION:**

Requirement Specification for Time Table Management System (TTMS) for College Administration:

**Functional Requirements:  
1.1 User Management:**

* + The system should allow registration and authentication of administrators, faculty, and students.
  + Administrators should be able to manage user accounts, including adding, updating, and deleting accounts.
  + Users should have different access levels based on their roles (administrator, faculty, student).

**1.2 Timetable Generation:**

* + The system should automatically generate initial timetables based on input parameters such as course offerings, faculty availability, and classroom capacities.
  + Administrators should be able to customize generated timetables by making manual adjustments.
  + Timetables should be optimized to minimize conflicts and maximize resource utilization.

**1.3 Course Management:**

* + Administrators should be able to add, update, and delete course details, including course codes, names, descriptions, and offerings.
  + Students should be able to view available courses, check course descriptions, and enroll in desired classes.

**1.4 Faculty Management:**

* + Administrators should be able to manage faculty profiles, including personal information, qualifications, and availability.
  + Faculty members should be assigned to teach specific courses and provided with teaching schedules.
  + Faculty should be able to input and update their availability preferences.

**1.5 Classroom and Resource Management**:

* + Administrators should be able to manage classroom details, including capacities, facilities, and locations.
  + Resources such as laboratories and equipment should be allocated based on availability and requirements.
  + Users should be able to request and book specific resources for academic or non-academic purposes.

**1.6 Communication:**

* + The system should send automated notifications to users regarding timetable updates, class cancellations, room changes, and other relevant information.
  + Users should be able to communicate with each other and with administrators for inquiries, feedback, and announcements.

**1.7 Reporting and Analytics:**

* + Administrators should have access to comprehensive reports on resource utilization, faculty workload distribution, scheduling conflicts, and other key metrics.
  + Analytics tools should be provided to identify trends, patterns, and areas for improvement in timetable management and academic scheduling.

**Non-functional Requirements:  
2.1 Usability:**

* + The system should have an intuitive and user-friendly interface for easy navigation and interaction.
  + It should provide clear instructions and guidance for users to perform tasks efficiently.

**2.2 Performance:**

* + The system should be able to handle concurrent user requests without significant performance degradation.
  + Timetables should be generated and updated in a timely manner, reflecting real-time changes and updates.

**2.3 Security:**

* + User authentication and data transmission should be secured using encryption protocols.
  + Access control mechanisms should be in place to ensure that users can only access data and features appropriate to their roles.

**2.4 Reliability:**

* + The system should have high availability and uptime, with minimal downtime for maintenance or updates.
  + Data integrity should be maintained, with regular backups and recovery procedures in place.

**2.5 Scalability:**

* + The system should be scalable to accommodate an increasing number of users, courses, and resources over time.

# **HARDWARE AND SOFTWARE REQUIREMENTS:**

**Hardware Requirements:**

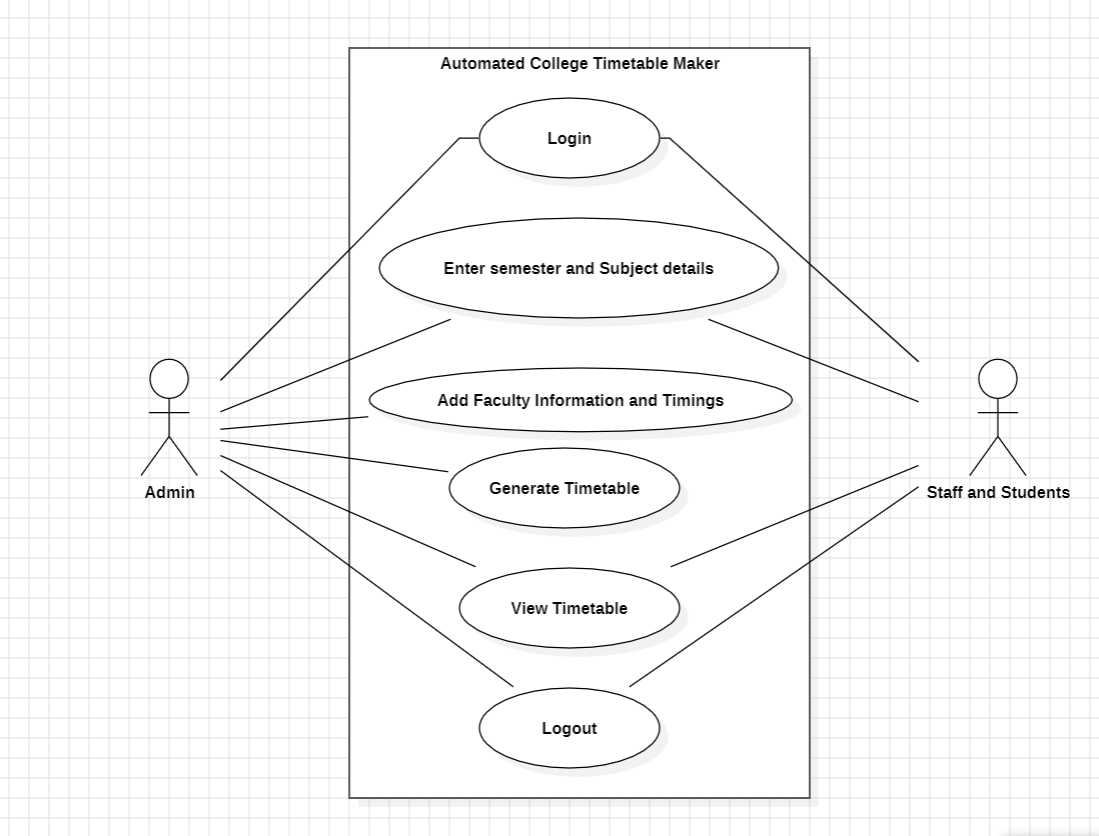
1. **Server:**
   * Processor: Multi-core processor (e.g., Intel Core i5 or equivalent)
   * RAM: Minimum 8 GB (16 GB or higher recommended for better performance)
   * Storage: Sufficient storage space for database and application files (SSD recommended for faster access)
   * Network Interface: Ethernet port for network connectivity
2. **Client Devices (Desktop/Laptop):**
   * Processor: Dual-core processor or higher
   * RAM: Minimum 4 GB
   * Storage: Sufficient space for web browser cache and temporary files
   * Network Interface: Ethernet port or Wi-Fi adapter for network connectivity
   * Display: Monitor with at least 1024x768 resolution
3. **Mobile Devices (Optional, for Mobile App):**
   * Compatible with Android or iOS platforms
   * Sufficient processing power and memory to run the mobile application smoothly
   * Network connectivity via Wi-Fi or cellular data

# **Software Requirements:**

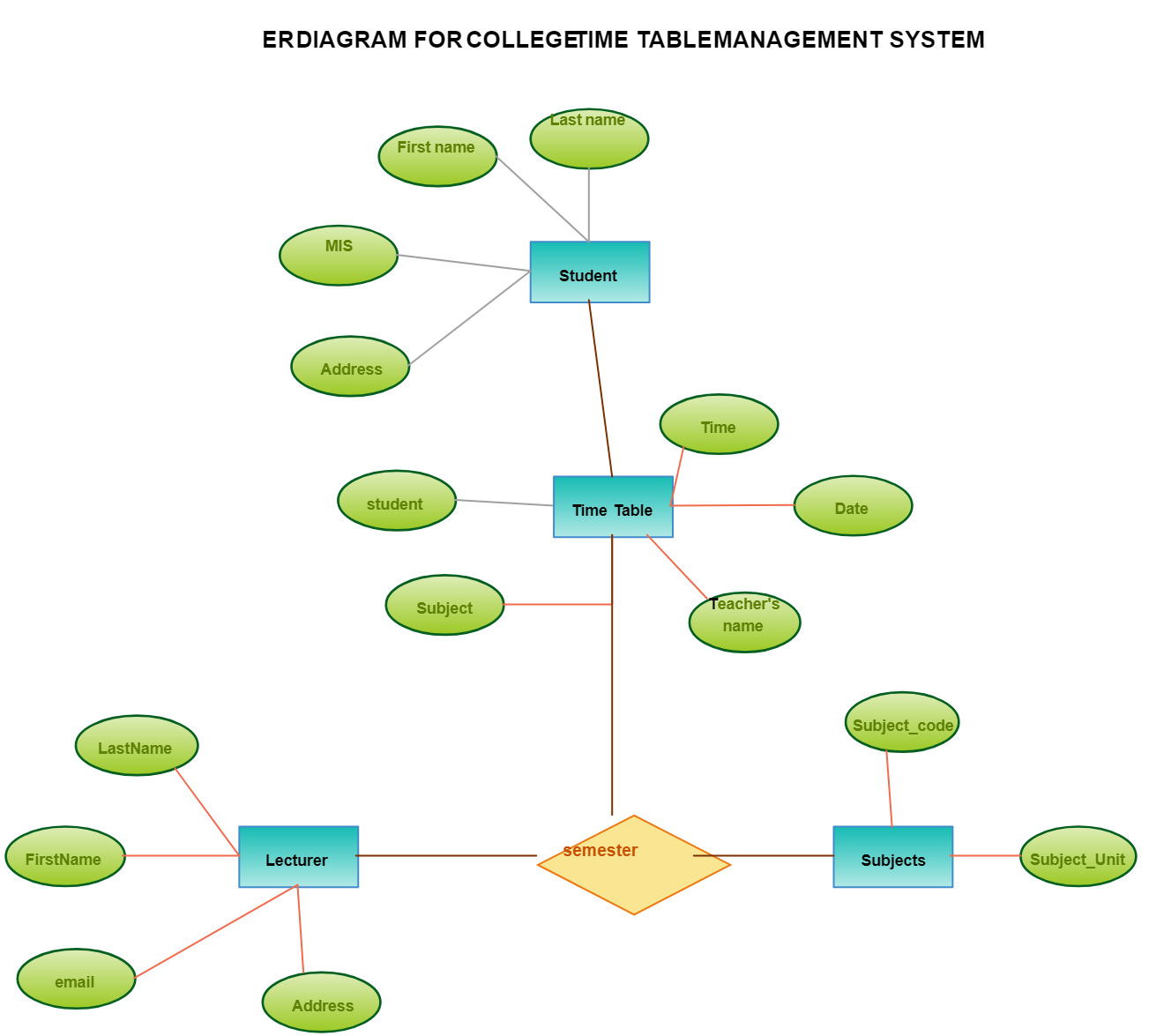
1. **Operating System:**
   * Server: Linux (e.g., Ubuntu Server, CentOS) or Windows Server
   * Clients: Windows, macOS, Linux, or any other operating system with a modern web browser
   * Mobile App: Android (Java/Kotlin) or iOS (Swift/Objective-C)
2. **Web Server:**
   * Apache HTTP Server, Nginx, or Microsoft IIS
3. **Database Management System:**
   * MySQL, PostgreSQL, or MariaDB
4. **Programming Languages and Frameworks:**
   * Backend: Node.js with Express.js, Python with Django or Flask, Java with Spring Framework
   * Frontend: HTML5, CSS3, JavaScript (with frameworks like React.js, Angular, or Vue.js)
   * Mobile App: Android (Java/Kotlin) with Android Studio, iOS (Swift/Objective-C) with Xcode
5. **Version Control:**
   * Git for version control management
6. **Additional Tools and Libraries:**
   * IDE (Integrated Development Environment): Visual Studio Code, IntelliJ IDEA, Eclipse, or any other preferred IDE for development
   * Package Managers: npm (for Node.js), pip (for Python), Gradle/Maven (for Java)
   * Authentication and Authorization: JWT (JSON Web Tokens), OAuth 2.0
   * Real-time Communication (Optional): WebSocket protocol with libraries like Socket.IO
   * Testing Frameworks: Jest, Jasmine, JUnit
   * Documentation: Swagger/OpenAPI for API documentation
7. **Security:**
   * SSL/TLS certificate for secure data transmission over HTTPS
   * Firewall configuration to restrict unauthorized access to the server
   * Regular software updates and security patches installation
8. **Deployment:**
   * Containerization: Docker for containerization and Kubernetes for orchestration (optional)
   * Continuous Integration/Continuous Deployment (CI/CD) Pipeline: Jenkins, Travis CI, GitLab CI/CD, or similar tools for automated deployment

These hardware and software requirements provide a foundation for setting up and deploying the Time Table Management System, ensuring compatibility, performance, and security across all components of the system.

# **ARCHITECTURE DIAGRAM:**



* 1. **ER DIAGRAM:**



# **CHAPTER-4 PROGRAM CODE**

import streamlit as st

from pymongo import MongoClient

from datetime import datetime

# MongoDB connection

client = MongoClient('mongodb://localhost:27017/')

db = client['timetable\_db']

collection = db['timetable']

# Streamlit UI

st.title('Timetable Management System')

# Form to add timetable entries

st.header('Add Timetable Entry')

with st.form('add\_entry\_form'):

date = st.date\_input('Date')

time = st.time\_input('Time')

subject = st.text\_input('Subject')

teacher = st.text\_input('Teacher')

room = st.text\_input('Room')

submit\_button = st.form\_submit\_button(label='Add Entry')

if submit\_button:

entry = {

'date': date.isoformat(),

'time': time.isoformat(),

'subject': subject,

'teacher': teacher,

'room': room

}

collection.insert\_one(entry)

st.success('Entry added successfully!')

# Display timetable entries

st.header('Timetable Entries')

entries = collection.find().sort([('date', 1), ('time', 1)])

for entry in entries:

st.subheader(f"{entry['date']} {entry['time']}")

st.write(f"\*\*Subject:\*\* {entry['subject']}")

st.write(f"\*\*Teacher:\*\* {entry['teacher']}")

st.write(f"\*\*Room:\*\* {entry['room']}")

if st.button(f"Delete Entry {entry['\_id']}", key=str(entry['\_id'])):

collection.delete\_one({'\_id': entry['\_id']})

st.success('Entry deleted successfully!')

st.experimental\_rerun()

# **CHAPTER-5 RESULT AND DOCUMENTATION**

**5.1 USER DOCUMENTATION:**

**REGISTERATION MODULE:**



**RECORD MANAGEMENT MODULE:**

****

# **CHAPTER-6**

**6.1 CONCLUSION:**

After completing the Attendance Management System project, we are confident that it effectively addresses the challenges present. This computerized system is designed to reduce human errors and significantly enhance operational efficiency. The primary goal of this project was to minimize human effort by automating routine tasks and streamlining the management processes.For instance, users can simply type a search string to quickly find specific records, and editing records is simplified through easy-to-use update functions.

Overall, the Attendance Management System achieves its primary objective of providing accurate and efficient management of attendance records. By automating data management and simplifying navigation and editing, the system ensures that records can focus on more strategic tasks.

This project demonstrates the practical application of database management systems (DBMS) in addressing real-world challenges, particularly in attendance management.

# **CHAPTER-7**

**7.1 REFERENCES:**

1. Python Documentation : Python Software Foundation. (n.d.). Available at: <https://docs.python.org/3/>

2. Streamlit, Inc. (2021) ‘Streamlit Documentation’, Streamlit, Available at: <https://docs.streamlit.io/>

3. Williams, K. (2018) ‘MongoDB in Action’, Manning Publications, pp.1-375.

4. Davidson, P. (2015) ‘MongoDB Basics’, Apress, pp.1-250.

5. Greene, M. (2020) ‘Practical Python Programming: 100+ Essential Coding Exercises and Projects’, No Starch Press, pp.1-450.