



SRI KRISHNA COLLEGE OF TECHNOLOGY
(An Autonomous Institution)
Approved by AICTE | Affiliated to Anna University Chennai
Accredited by NBA - AICTE | Accredited by NAAC with 'A' Grade
KOVAIPUDUR, COIMBATORE 641042



Automatic Time Table Generation

A PROJECT REPORT

Submitted by

JAYA ALTRIN G - 727822TUAM019

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

JULY 2024

BONAFIDE CERTIFICATE

Certified that this project report **AUTOMATIC TIME TABLE GENERATION** is the bonafide work of **JAYA ALTRIN G 727822TUAM019** who carried out the project work under my supervision.

SIGNATURE

Mrs. SOUNDARYA S

SUPERVISOR

Assistant Professor,

Department of Computer Science
and Engineering (Artificial
Intelligence and Machine
Learning),

Sri Krishna College of

Technology,Coimbatore-641042

SIGNATURE

Dr. SUMA SIRA JACOB

HEAD OF THE DEPARTMENT

Associate Professor,

Department of Computer Science and
Engineering (Artificial Intelligence and
Machine Learning),

Sri Krishna College of

Technology,Coimbatore-641042

Certified that the candidate was examined by me in the Project Work Viva Voce examination held on _____ at Sri Krishna College of Technology, Coimbatore-641042.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost we thank the **Almighty** for being our light and for showering his gracious blessings throughout the course of this project.

We express our gratitude to our beloved Principal, **Dr. M.G. Sumithra**, for providing all facilities.

We are grateful to our beloved Head, Computing Sciences **Dr.T. Senthilnathan**, for her tireless and relentless support.

With the grateful heart, our sincere thanks to our Head of the Department **Dr. Suma Sira Jacob**, Department of Computer Science and Engineering for the motivation and all support to complete the project work.

We thank **Mrs. Soundarya**, Assistant Professor, Department of Computer Science and Engineering, for his motivation and support.

We are thankful to all the **Teaching and Non-Teaching Staff** of Department of Computer Science and Engineering and to all those who have directly and indirectly extended their help to us in completing this project work successfully.

We extend our sincere thanks to our **family members** and our beloved **friends**, who had been strongly supporting us in all our endeavour

ABSTRACT

The efficient management of academic schedules is a critical challenge for educational institutions, impacting both faculty and students. Traditional timetable generation methods often involve manual effort and are prone to errors and inefficiencies. This project presents an automated timetable generation system designed to optimize the scheduling process, minimize conflicts, and ensure optimal resource utilization. By leveraging advanced algorithms and heuristics, the system dynamically creates timetables that accommodate various constraints, including classroom availability, faculty preferences, and course requirements. The system utilizes a constraint satisfaction approach to handle complex scheduling needs, incorporating features such as conflict resolution, priority-based scheduling, and real-time adjustments. A user-friendly interface allows administrators to input data, configure preferences, and generate timetables with minimal effort. The project aims to enhance the scheduling efficiency, reduce administrative workload, and improve overall educational experience by providing well-organized and conflict-free schedules. Through rigorous testing and validation, the system demonstrates its effectiveness in various academic settings, showcasing its adaptability and robustness in handling diverse scheduling scenarios. This automated solution represents a significant advancement in academic administration, offering a reliable and scalable approach to timetable generation.

TABLE OF CONTENT

CHAPTER.NO	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 PROBLEM STATEMENT	1
	1.2 OBJECTIVE	1
	1.3 OVERVIEW	2
2	SYSTEM SPECIFICATIONS	3
	2.1 VS CODE	3
	2.2 LOCAL STORAGE	3
3	PROPOSED SYSTEM	5
	3.1 PROPOSED SYSTEM	5
	3.2 ADVANTAGES	5
4	METHODOLOGIES	7
5	IMPLEMENTATION AND RESULT	10
	5.1 LOGIN	10
	5.2 USER REGISTER	11
	5.3 ADD TIMETABLE	12
	5.4 VIEW TIMETABLE	13
	5.5 CODING	14
6	BACKEND SYSTEM SPECIFICATION	20
	6.1 MYSQL	20
		21

	6.2 REST API	
	6.3 SPRING BOOT	22
7	SYSTEM ARCHITECTURE	23
	7.1 BACKEND	23
	7.2 FLOWCHART	25
	7.2 CODING	26
	7.3 SECURITY AND AUTHENTICATION	29
8	CONCLUSION	33
	8.1 CONCLUSION	33
	8.2 FUTURE SCOPE	33

LIST OF FIGURES

Figure No	TITLE	Page No
4.1	Process-flow Diagram	7
5.1	Login Page	10
5.2	Register Page	11
5.3	Add Time Table Page	12
5.4	View Time Table Page	13
6.1	My SQL	20
7.1	Backend System Architecture	23
7.2	Backend Flow Diagram	25

LIST OF ABBREVIATIONS

Abbreviation

Acronym

HTML

HYPERTEXT MARKUP LANGUAGE

CSS

CASCADING STYLESHEET

JS

JAVASCRIPT

SDLC

SOFTWARE DEVELOPMENT LIFE CYCLE

API

APPLICATION PROGRAMMING INTERFACE

DB

DATABASE

SQL

STRUCTURED QUERY LANGUAGE

CHAPTER 1

INTRODUCTION

Timetable management in educational institutions is often labor-intensive and prone to conflicts due to manual scheduling. Balancing constraints such as faculty availability and classroom assignments can lead to inefficiencies and errors. This project introduces an automated timetable generation system designed to optimize this process. By leveraging advanced algorithms, the system efficiently creates conflict-free schedules while adapting to various constraints. It offers a user-friendly interface for administrators and reduces the administrative burden, ensuring better resource utilization and improved scheduling efficiency.

1.1 PROBLEM STATEMENT

Timetable creation in educational institutions is often manual, leading to errors, conflicts, and inefficiencies due to constraints like faculty availability and room assignments. This process is time-consuming and prone to inaccuracies. The challenge is to develop an automated timetable generation system that efficiently handles these constraints, produces conflict-free schedules, and minimizes administrative effort. This system aims to enhance scheduling accuracy, optimize resource utilization, and streamline the timetable creation process.

1.2 OBJECTIVE

The objective of this project is to develop an automated timetable generation system that enhances the efficiency and accuracy of scheduling processes in educational institutions. The system aims to optimize timetable creation by leveraging advanced algorithms to produce conflict-free schedules that accommodate various constraints, such as faculty availability, classroom capacities, and course requirements. By minimizing manual intervention, the system seeks to reduce scheduling errors and administrative workload. Additionally, it strives to improve resource utilization and provide flexibility for dynamic adjustments, ultimately

streamlining the scheduling process and enhancing overall institutional efficiency.

1.3 OVERVIEW

Timetable management in educational institutions involves scheduling classes, allocating resources, and accommodating various constraints such as faculty availability, room capacities, and course requirements. Traditionally, this process is manual and prone to inefficiencies and errors. The automated timetable generation system addresses these challenges by applying advanced algorithms to create optimized, conflict-free schedules. The system is designed to handle complex constraints and adapt to dynamic changes, providing a more efficient and accurate solution for timetable management.

CHAPTER 2

SYSTEM SPECIFICATION

In this chapter, we are going to see the software that we have used to build the website. This chapter gives you a small description about the software used in the project.

2.1 VS CODE

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences.

VS Code is an excellent code editor for React projects. It is lightweight, customizable, and has a wide range of features that make it ideal for React development. It has built-in support for JavaScript, JSX, and TypeScript, and enables developers to quickly move between files and view detailed type definitions. It also has a built-in terminal for running tasks. Additionally, VS Code has an extensive library of extensions that allow developers to quickly add features like code snippets, debugging tools, and linting support to their projects.

2.2 LOCAL STORAGE

Local storage is a type of web storage for storing data on the client side of a web browser. It allows websites to store data on a user's computer, which can then be accessed by the website again when the user returns. Local storage is a more secure alternative to cookies because it allows websites to store data without having to send it back and forth with each request. Local storage is a key-value pair storage mechanism, meaning it stores data in the form of a key and corresponding value. It is similar to a database table in that it stores data in columns and rows, except that local storage stores the data in the browser rather than in a database.

preferences and settings, or to store data that is not meant to be shared with other websites. It is also used to cache data to improve the performance of a website. Local storage is supported by all modern web browsers, including Chrome, Firefox, Safari, and Edge. It is accessible through the browser's JavaScript API. Local storage is a powerful tool for websites to store data on the client side. It is secure, efficient, and can be used to store data that does not need to be shared with other websites.

Local Storage is a great way to improve the performance of a website by caching data. Local storage in web browsers allows website data to be stored locally on the user's computer. It is a way of persistently storing data on the client side, which is not sent to the server with each request. This allows users to store data such as preferences, login information, and form data without needing to send it to a server. It is typically stored in a browser's cookie file, but it can also be stored in other locations such as HTML5 Local Storage and Indexed DB. The data stored in local storage is persistent and can be accessed by the website even if the user closes the browser or navigates to another page. It is a great way for websites to store user-specific data, as it is secure, reliable, and fast. It is also a great way for developers to store data that does not need to be sent to the server with each request.

One of the key benefits of using local storage is its reliability. Unlike server-side storage, which can be affected by network outages or other server issues, local storage is stored locally on the user's machine, and so is not affected by these issues. Another advantage of local storage is its speed. Because the data is stored locally, it is accessed quickly, as there is no need to send requests to a server. This makes it ideal for storing data that needs to be accessed quickly, such as user preferences or session data. Local storage is also secure, as the data is stored on the user's machine and not on a server. This means that the data is not accessible by anyone other than the user, making it a good choice for storing sensitive information.

CHAPTER 3

PROPOSED SYSTEM

This chapter gives a small description about the proposed idea behind the development of our website

3.1 PROPOSED SYSTEM

The proposed system for automated timetable generation aims to streamline and enhance the scheduling process in educational institutions by leveraging advanced computational techniques. It features a user-friendly data input module that allows administrators to easily enter faculty availability, room capacities, course requirements, and student preferences. The system's core is an advanced constraint management engine that processes this data, addressing various constraints to avoid scheduling conflicts and optimize resource utilization. The timetable generation engine uses heuristic and optimization algorithms to create efficient and conflict-free schedules. Additionally, the system includes a real-time adjustment feature for dynamic updates and a validation module to ensure compliance with all requirements. The final timetable, along with detailed reports and resource utilization statistics, is presented through an output interface, reducing manual effort, minimizing errors, and improving overall scheduling efficiency.

3.2 ADVANTAGES

Enhanced Efficiency : Automates the scheduling process, significantly reducing the time and effort required for manual timetable creation.

Conflict-Free Scheduling: Advanced algorithms ensure that timetables are generated without conflicts, accommodating all constraints and preferences effectively.

Error Reduction: Minimizes human errors and inconsistencies associated with manual scheduling, leading to more accurate and reliable timetables.

Optimal Resource Utilization: Maximizes the use of available resources, including faculty and classroom spaces, by efficiently allocating them based on constraints and requirements.

Flexibility and Adaptability: Allows for real-time adjustments and dynamic updates to the schedule based on changing needs or unforeseen circumstances, ensuring that the timetable remains relevant and accurate.

Improved Administrative Productivity: Reduces the administrative workload and simplifies the scheduling process, freeing up time for other important tasks.

Enhanced Stakeholder Satisfaction: Provides well-organized schedules that meet the needs of students and faculty, leading to increased satisfaction and smoother operations within the institution.

Scalability: Handles large volumes of data and complex scheduling requirements, making it suitable for institutions of varying sizes and complexities.

CHAPTER 4

METHODOLOGIES

This chapter gives a small description about how our system works

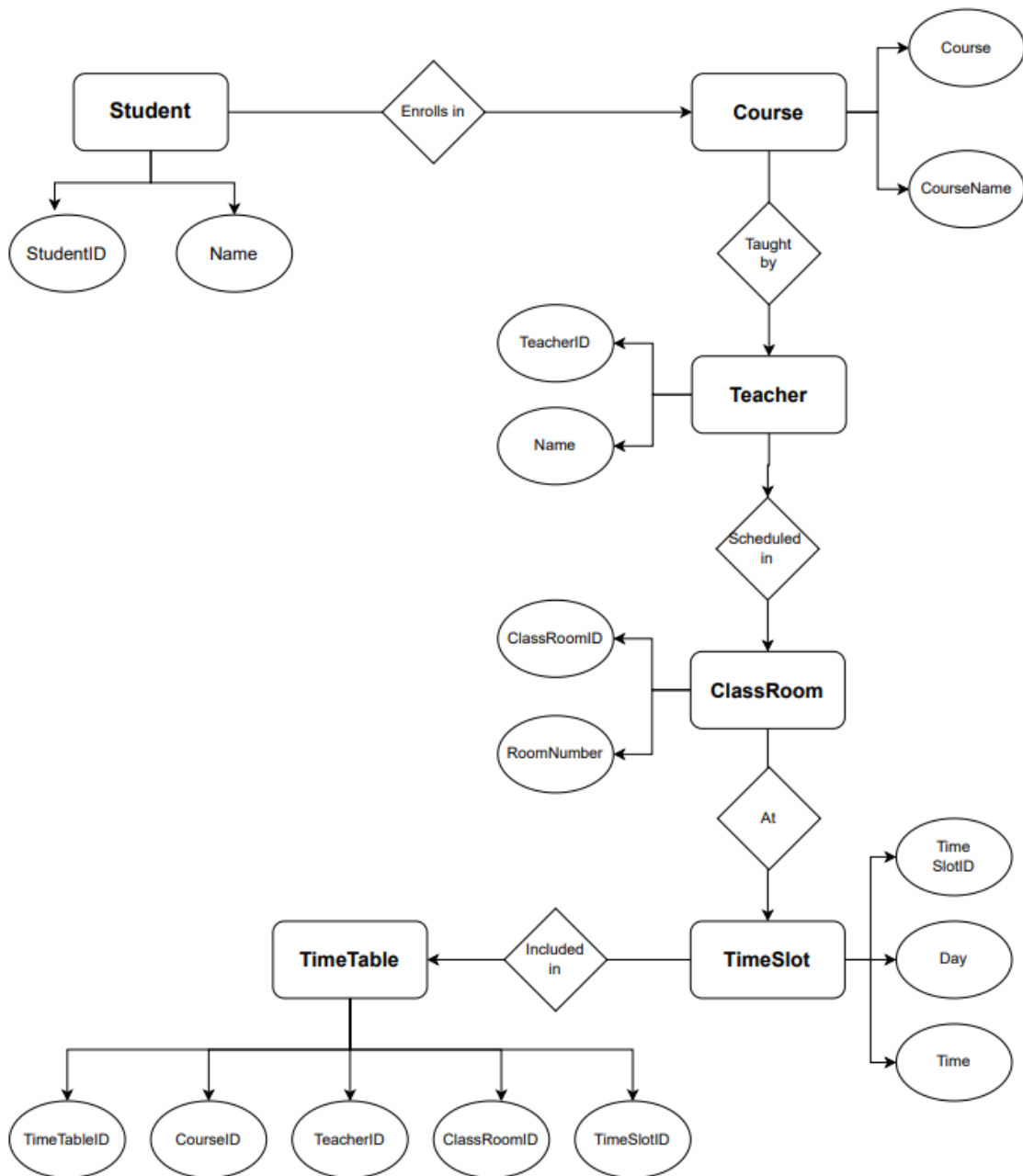


Fig 4.1.Process flow diagram

An automatic timetable generation system uses this schema to efficiently organize and schedule classes, ensuring there are no conflicts and all constraints are met. Here's how each component contributes to the project:

1. Student Entity:

- Purpose: Manages student information and their course enrollments.
- Usage: Helps in determining the courses a student needs to attend.

2. Course Entity:

- Purpose: Manages course details.
- Usage: Defines what courses are available for scheduling and who will teach them.

3. Teacher Entity:

- Purpose: Manages teacher details.
- Usage: Ensures each course is assigned a teacher, and their availability is considered during scheduling.

4. Classroom Entity:

- Purpose: Manages classroom details.
- Usage: Ensures there is a physical space assigned for each scheduled class.

5. TimeSlot Entity:

- Purpose: Manages timeslot details (day and time).
- Usage: Provides the temporal framework for scheduling classes, avoiding conflicts.

6. TimeTable Entity:

- Purpose: Integrates course, teacher, classroom, and timeslot data.
- Usage: Forms the core of the timetable generation, where each entry specifies which course is taught by which teacher, in which classroom, and at what time.

Automatic Timetable Generation Process

1. Input Data:

- Information about students, courses, teachers, classrooms, and timeslots is entered into the system.

2. Constraint Handling:

- The system takes into account constraints such as:
 - Teacher availability.
 - Classroom capacity and availability.
 - Student course enrollments.
 - Timeslot conflicts.

3. Timetable Generation:

- An algorithm (e.g., genetic algorithm, constraint satisfaction problem solver) processes the input data and constraints to generate an optimal timetable.
- The generated timetable ensures that each course is assigned a timeslot, a teacher, and a classroom without any overlaps or conflicts.

4. Output:

- The final timetable is produced, which can be queried and viewed by students and teachers.

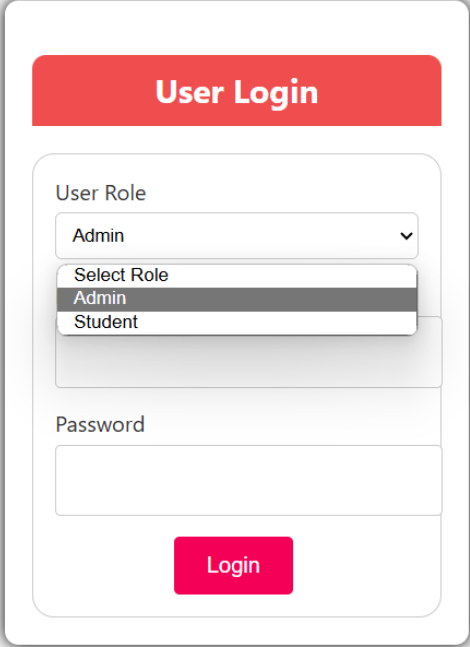
CHAPTER 5

IMPLEMENTATION AND RESULT

This chapter gives a description about the output that we produced by developing the website of our idea

5.1 LOGIN

Please log in to access your personalized timetable. Once logged in, you can effortlessly create and manage your weekly schedule. Our intelligent system automatically generates your timetable based on your preferences and academic requirements, ensuring that you never miss an important event or class. Whether you're a student managing your courses or a teacher organizing your lessons, our tool streamlines the scheduling process and keeps you organized. Enter your credentials to get started and make your scheduling experience smooth and efficient.

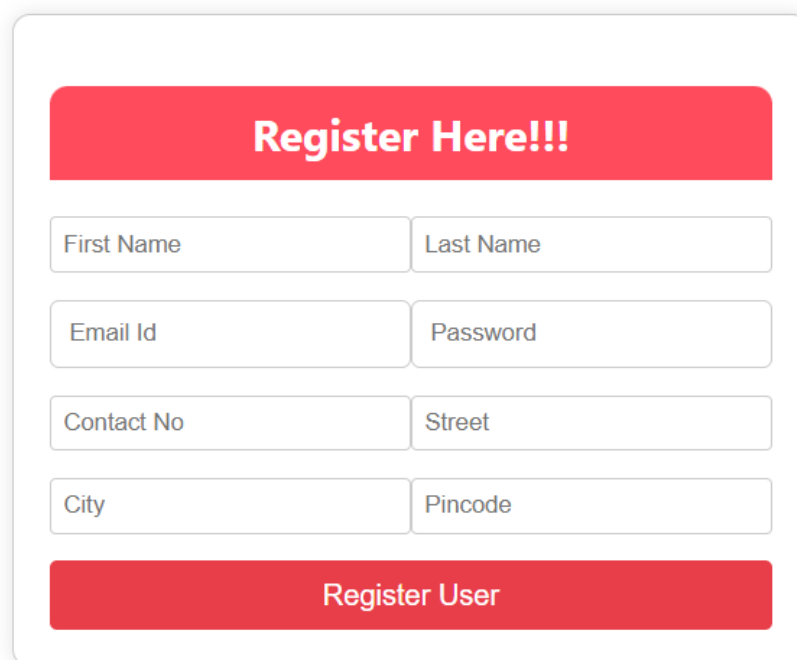


The image shows a 'User Login' form. At the top is a red header with the text 'User Login'. Below this is a section for 'User Role' with a dropdown menu currently showing 'Admin'. The dropdown menu is open, showing options: 'Select Role', 'Admin', and 'Student'. Below the role selection is a 'Password' field with a text input box. At the bottom of the form is a red 'Login' button.

Fig 5.1 LOGIN PAGE

5.2 USER REGISTER

Our Automatic Timetable Generation system simplifies scheduling by automatically creating optimal timetables based on user preferences and constraints. By entering your classes, activities, and available times, the system efficiently generates a conflict-free schedule that maximizes your time management. With features like real-time adjustments and calendar integration, staying organized has never been easier. Enjoy a seamless scheduling experience that adapts to your needs and helps you stay on top of your commitments effortlessly.



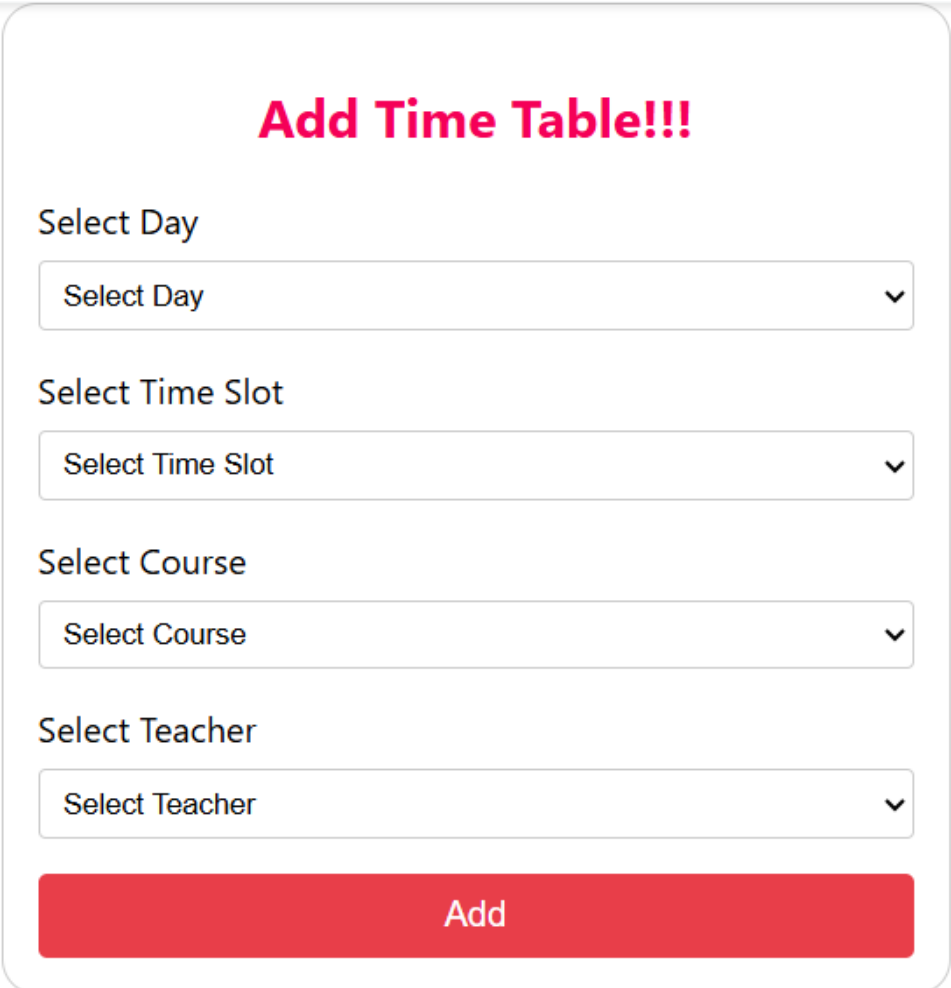
The registration form is presented within a white rounded rectangle with a subtle drop shadow. At the top, a red banner contains the text 'Register Here!!!' in white. Below this, there are eight input fields arranged in four rows of two. The first row contains 'First Name' and 'Last Name'. The second row contains 'Email Id' and 'Password'. The third row contains 'Contact No' and 'Street'. The fourth row contains 'City' and 'Pincode'. At the bottom of the form is a red button labeled 'Register User' in white text.

Register Here!!!	
First Name	Last Name
Email Id	Password
Contact No	Street
City	Pincode
Register User	

Fig 5.2 REGISTER PAGE

5.3 ADD TIME TABLE

Here, you can view and manage your automatically generated schedule with ease. The system has created a timetable based on your input and preferences, ensuring that all your classes and activities are seamlessly organized. You can view your weekly schedule at a glance, make adjustments if needed, and see any upcoming events or changes. Use the interactive features to drag and drop activities, adjust timings, or add new events. Stay on top of your commitments and make the most of your time with our intuitive timetable management tools.

A form titled "Add Time Table!!!" in bold pink text. It contains four dropdown menus: "Select Day", "Select Time Slot", "Select Course", and "Select Teacher". Each dropdown menu has a placeholder text "Select [Day/Time Slot/Course/Teacher]" and a downward arrow icon. At the bottom of the form is a red button with the text "Add" in white.

Add Time Table!!!

Select Day

Select Day ▼

Select Time Slot

Select Time Slot ▼

Select Course

Select Course ▼

Select Teacher

Select Teacher ▼

Add

Fig 5.3 ADD TIME TABLE PAGE

5.4 VIEW TIME TABLE PAGE

Here, you can effortlessly review your personalized schedule created by our automatic timetable generation system. This page provides a clear, interactive layout of your weekly or daily timetable, highlighting all your classes, activities, and appointments. Easily navigate through different days and times to get an overview of your commitments. You can also make adjustments, add new events, or rearrange existing ones with a simple drag-and-drop interface. Stay organized and manage your time effectively with a comprehensive view of your entire schedule.

Time Table

Time Slot \ Day	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 AM - 10:00 AM	NA	NA	NA	NA	NA
10:00 AM - 11:00 AM	NA	NA	NA	NA	NA
11:00 AM - 12:00 PM	NA	NA	NA	NA	NA
12:00 PM - 1:00 PM	NA	NA	NA	NA	NA
1:00 PM - 2:00 PM	NA	NA	NA	NA	NA
2:00 PM - 3:00 PM	NA	NA	NA	NA	NA
3:00 PM - 4:00 PM	NA	NA	NA	NA	NA
4:00 PM - 5:00 PM	NA	NA	NA	NA	NA

Download Timetable

Fig 5.4 VIEW TIME TABLE PAGE

5.5 CODING

LOGIN PAGE:

```
import React, { useState } from 'react';
import './Login.css';
import { useNavigate } from 'react-router-dom';

function Login() {
  const navigate = useNavigate();
  const [role, setRole] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");

  const handleLogin = () => {
    if (role === 'admin' && email === 'Admin@gmail.com' && password ===
'Admin') {
      navigate('/Admin');
    } else if (role === 'student' && email.endsWith('@skct.edu.in') && password ===
'Student') {
      navigate('/');
    } else {
      setError('Invalid credentials. Please try again.');
```

```
    }
  };

  return (
    <div className="login-container">
      <div className="login-box">
```

```

<h2>User Login</h2>
{error && <div className="error-message">{error}</div>}
<form>
  <div className="form-group">
    <label htmlFor="role">User Role</label>
    <select
      id="role"
      value={role}
      onChange={(e) => setRole(e.target.value)}
    >
      <option value="">Select Role</option>
      <option value="admin">Admin</option>
      <option value="student">Student</option>
    </select>
  </div>
  <div className="form-group">
    <label htmlFor="email">Email</label>
    <input
      type="email"
      id="email"
      value={email}
      onChange={(e) => setEmail(e.target.value)}
    />
  </div>
  <div className="form-group">
    <label htmlFor="password">Password</label>
    <input

```

```

        type="password"
        id="password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      />
    </div>
    <button type="button" onClick={handleLogin}>Login</button>
  </form>
</div>
</div>
);
}

```

```
export default Login;
```

TIMETABLE PAGE:

```

import React from 'react';
import html2canvas from 'html2canvas';
import jsPDF from 'jspdf';
import './viewTimeTable.css';

const ViewTimeTable = ({ schedule }) => {
  const days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'];
  const timeSlots = [
    '9:00 AM - 10:00 AM',
    '10:00 AM - 11:00 AM',
    '11:00 AM - 12:00 PM',

```



```

'12:00 PM - 1:00 PM',
'1:00 PM - 2:00 PM',
'2:00 PM - 3:00 PM',
'3:00 PM - 4:00 PM',
'4:00 PM - 5:00 PM',
];

```

```

const downloadPDF = () => {
  const input = document.getElementById('timetable');
  html2canvas(input).then((canvas) => {
    const imgData = canvas.toDataURL('image/png');
    const pdf = new jsPDF();
    const imgWidth = 190; // Width of the image in mm
    const pageHeight = 295; // Height of the page in mm
    const imgHeight = canvas.height * imgWidth / canvas.width;
    let heightLeft = imgHeight;

    let position = 0;
    pdf.addImage(imgData, 'PNG', 0, position, imgWidth, imgHeight);
    heightLeft -= pageHeight;
    while (heightLeft >= 0) {
      position = heightLeft - imgHeight;
      pdf.addPage();
      pdf.addImage(imgData, 'PNG', 0, position, imgWidth, imgHeight);
      heightLeft -= pageHeight;
    }
    pdf.save('timetable.pdf');
  });
}

```

```

    });
};

return (
  <div className="timetable-container">
    <h2>Time Table</h2>
    <div id="timetable">
      <table className="timetable">
        <thead>
          <tr>
            <th>Time Slot \ Day</th>
            {days.map((day) => (
              <th key={day}>{day}</th>
            ))}
          </tr>
        </thead>
        <tbody>
          {timeSlots.map((slot, i) => (
            <tr key={slot}>
              <td>{slot}</td>
              {days.map((day, j) => (
                <td key={j}>{schedule[i][j]}</td>
              ))}
            </tr>
          ))}
        </tbody>
      </table>

```

```
    </div>
    <button className="Download TimeTable" onClick={downloadPDF}>
      Download Timetable
    </button>
  </div>
);
};

export default ViewTimeTable;
```

CHAPTER 6

BACKEND SYSTEM SPECIFICATION

In this chapter, the content discusses the software employed for constructing the website. This chapter provides a brief description of the software utilized in the project.

6.1 MySQL

Tables_in_demo
batch
course
grade
student
teacher
time_table
time_table_friday
time_table_monday
time_table_thursday
time_table_tuesday
time_table_wednesday
user
user_roles

Fig 6.1 My SQL

Local storage is a type of web storage for storing data on the client side of a web browser. It allows websites to store data on a user's computer, which can then be accessed by the website again when the user returns. Local storage is a more secure alternative to cookies because it allows websites to store data without having to send it back and forth with each request. It is similar to a database table in that it stores data in columns and rows, except that local storage stores the data in the browser rather than in a database.

Local storage is often used to store user information such as preferences and settings, or to store data that is not meant to be shared with other websites.

It is also used to cache data to improve the performance of a website. Local storage is

supported by all modern web browsers ,including chrome, Firefox ,Safari, and Edge. It is accessible through the browser's JavaScript API. Local storage is a powerful tool for websites to store data on the client side. It is secure, efficient, and can be used to store data that does not need to be shared with other websites.

Local Storage is a great way to improve the performance of a website by caching data. Local storage in web browsers allows website data to be stored locally on the user's computer. It is a way of persistently storing data on the client side, which is not sent to the server with each request. This allows users to store data such as preferences, login information, and form data without needing to send it to a server.

It is typically stored in a browser's cookie file, but it can also be stored in other locations such as HTML5 Local Storage and Indexed. The data stored in local storage is persistent and can be accessed by the website even if the user closes the browser or navigates to another page. It is a great way for websites to store user-specific data, as it is secure, reliable, and fast. It is also a great way for developers to store data that does not need to be sent to the server with each request.

One of the key benefits of using local storage is its reliability. Unlike server-side storage, which can be affected by network outages or other server issues, local storage is stored locally on the user's machine, and so is not affected by these issues. Another advantage of local storage is its speed. Because the data is stored locally, it is accessed quickly, as there is no need to send requests to a server.

6.2 REST API

A REST API (Representational State Transfer Application Programming Interface) is a popular architectural style for designing networked applications. It is based on a set of principles and constraints that allow for scalability, simplicity, and interoperability between systems.

Client-Server: Separated entities communicate over HTTP or a similar protocol, with distinct responsibilities and the ability to evolve independently.

Stateless: Each request from the client to the server must contain all the necessary information to understand and process the request. The server does not maintain any

client state between requests.

Uniform Interface: The API exposes a uniform interface, typically using HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. Resources are identified by URLs (Uniform Resource Locators).

Cacheable: Responses can be cached by the client or intermediaries to improve performance and reduce the load on the server.

Layered System: Intermediary servers can be placed between the client and server to provide additional functionality, such as load balancing, caching, or security.

6.3 SPRINGBOOT

Spring Boot is an open-source Java framework that simplifies the development of standalone, production-ready applications. It offers several advantages for building robust and scalable applications.

Simplified Configuration: Spring Boot eliminates the need for complex XML configuration files by leveraging sensible default configurations and annotations.

Embedded Server: Spring Boot includes an embedded server (e.g., Apache Tomcat, Jetty) that allows developers to create self-contained applications. This eliminates the need for external server installation and configuration, making it easier to package and deploy the application.

Dependency Management: Spring Boot incorporates the concept of starter dependencies, which are curated sets of libraries that provide commonly used functionalities. It simplifies dependency management and ensures that all required dependencies are included automatically, reducing configuration issues and potential conflicts.

Auto-Configuration: Spring Boot's auto-configuration feature analyzes the class path and automatically configures the application based on the detected dependencies. It saves developers from writing boilerplate configuration code, resulting in faster development and reduced code clutter.

Actuator: Spring Boot Actuator provides out-of-the-box monitoring and management endpoints for the application.

CHAPTER 7

SYSTEM ARCHITECTURE

The application adopts a contemporary and scalable three-tier architecture. It comprises the frontend layer, backend layer, and the database layer. Each of these layers fulfils a pivotal role in the application's comprehensive functionality, facilitating seamless communication and efficient data management

7.1 BACKEND

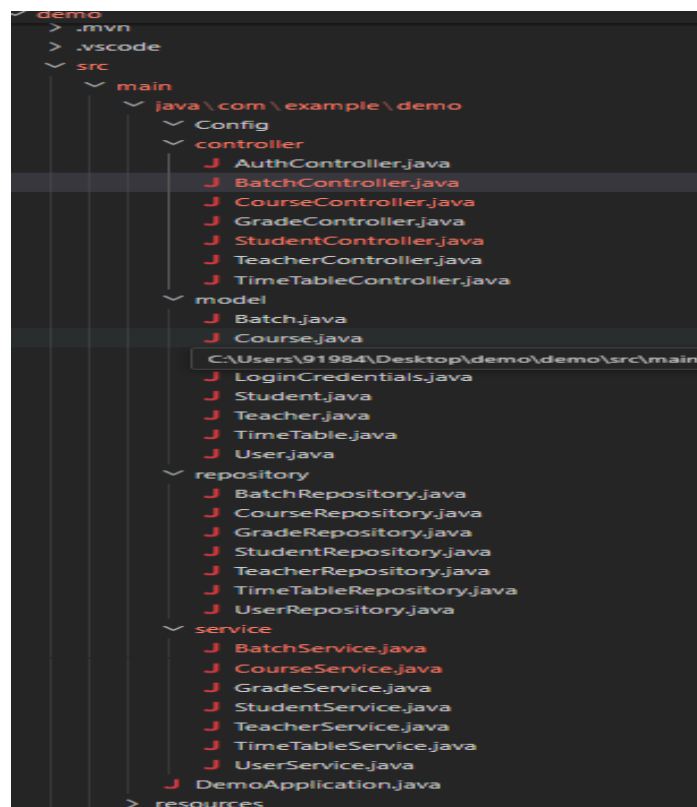


Fig 7.1 Backend System Architecture

The backend layer of the Automatic time table generation System is built upon the Spring Boot framework, a Java-based solution renowned for its capacity to simplify the development of resilient and scalable web applications. Spring Boot brings to the table a comprehensive array of features and libraries, offering streamlined solutions

for managing HTTP requests, data persistence, implementing robust security measures, and seamlessly integrating with external systems. Within the application, the backend takes on the primary responsibility of crafting RESTful APIs. These APIs are meticulously designed to empower CRUD (Create, Read, Update, Delete) operations, catering to the dynamic world of employee tax details management. Furthermore, the backend is equipped to handle user management and authentication, ensuring a secure and personalized user experience for both administrators and employees. In pursuit of enhanced security and modularity, the backend is strategically structured according to the principles of Spring Boot architecture.

Spring Boot: Spring Boot is a Java framework that simplifies the process of building enterprise-grade applications. It provides a robust set of features and conventions for developing backend systems, including dependency management, configuration, and automatic setup. Spring Boot follows the principle of convention over configuration, reducing the amount of boilerplate code required.

REST API: The backend of the Automatic time table generation System exposes a RESTful API that allows the frontend to communicate with the server. REST (Representational State Transfer) is an architectural style for designing networked applications. It uses standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD (Create, Read, Update, Delete) operations on resources. The API endpoints define the URLs and request/response formats for interacting with the system.

Controller: In the application, controllers have a crucial role in managing incoming HTTP requests. Controllers map these requests to the appropriate methods within the system. API endpoints are defined by controllers, and they orchestrate the processing logic of incoming requests. Controllers act as the gateway between the frontend and backend, receiving user inputs, validating and processing data, interacting with services, and returning the relevant responses.

Services: Services within the application encapsulate the essential business logic. Responsible for orchestrating complex operations and facilitating interactions between different system components, these operations encompass data retrieval, validation, transformation, and storage. In this context, services manage tax calculations, handle

user authentication, and other application-specific functionalities, ensuring a seamless user experience.

7.2 FLOWCHART

Here's a structured outline for the content of a chapter on the flowchart for an automatic timetable management project

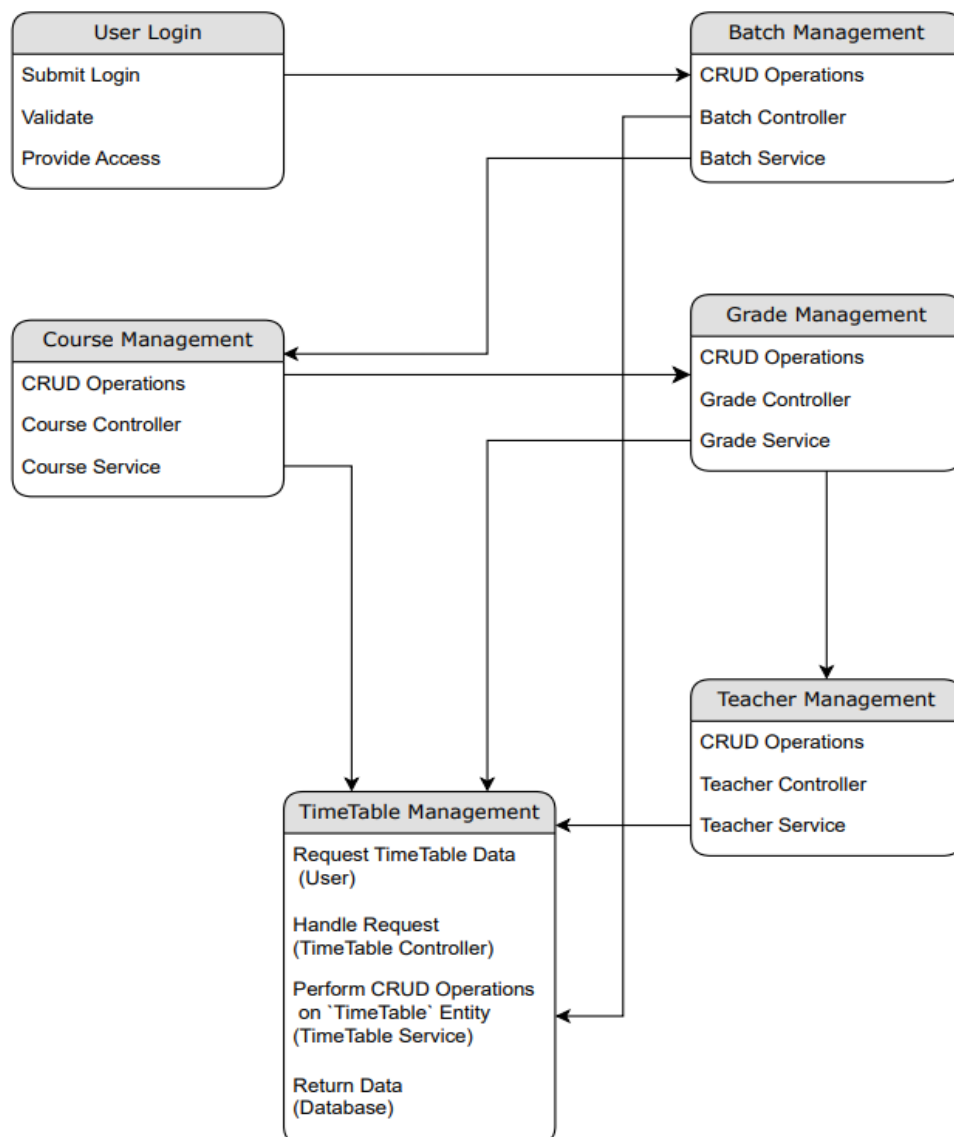


Fig 7.2 Backend Flow Diagram

7.3 CODING

TimeTableController.java

```
package com.example.demo.controller;
import com.example.demo.model.TimeTable;
import com.example.demo.service.TimeTableService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.Optional;

@RestController
@RequestMapping("/api/timetable")
public class TimeTableController {
    @Autowired
    private TimeTableService timeTableService;

    @GetMapping("/{id}")
    public ResponseEntity<TimeTable> getTimeTable(@PathVariable Long id) {
        Optional<TimeTable> timeTable = timeTableService.getTimeTable(id);
        return timeTable.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
    }
}
```

Timetable.java

```
package com.example.demo.model;
import java.util.List;
import jakarta.persistence.*;
```

```
@Entity
public class TimeTable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @ElementCollection
    private List<String> monday;
    @ElementCollection
    private List<String> tuesday;
    @ElementCollection
    private List<String> wednesday;
    @ElementCollection
    private List<String> thursday;
    @ElementCollection
    private List<String> friday;
    // Getters and Setters
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public List<String> getMonday() {
        return monday;
    }
    public void setMonday(List<String> monday) {
        this.monday = monday;
    }
    public List<String> getTuesday() {
        return tuesday;
    }
}
```

```

public void setTuesday(List<String> tuesday) {
    this.tuesday = tuesday;
}
public List<String> getWednesday() {
    return wednesday;
}
public void setWednesday(List<String> wednesday) {
    this.wednesday = wednesday;
}
public List<String> getThursday() {
    return thursday;
}
public void setThursday(List<String> thursday) {
    this.thursday = thursday;
}
public List<String> getFriday() {
    return friday;
}
public void setFriday(List<String> friday) {
    this.friday = friday;
}
}

```

TimeTableRepository

```

package com.example.demo.repository;
import com.example.demo.model.TimeTable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface TimeTableRepository extends JpaRepository<TimeTable, Long> {

```

```
}
```

TimeTableService.java

```
package com.example.demo.service;
import com.example.demo.model.TimeTable;
import com.example.demo.repository.TimeTableRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.Optional;
@Service
public class TimeTableService {
    @Autowired
    private TimeTableRepository timeTableRepository;
    public Optional<TimeTable> getTimeTable(Long id) {
        return timeTableRepository.findById(id);
    }
}
```

7.4 SECURITY AND AUTHENTICATION

1. User Authentication:

- Users, including students and administrators, would register and log in using email and strong, hashed passwords.
- The login system verifies credentials to control access to the timetable generation features and management tools, ensuring only authorized individuals can access sensitive scheduling information.

2. Data Encryption:

- Encrypt communication between the frontend (user interface) and backend (server) to protect data in transit.
- Encrypt sensitive information (e.g., user profiles, timetable details) stored in the database to safeguard it in case of a breach.

3. Session Management:

- Implement secure session tokens to manage user sessions, ensuring that sessions remain authenticated and secure during interactions with the timetable application.

4. Security Frameworks:

- Use security frameworks like Spring Security to handle user registration, login, password hashing, and role-based permissions, ensuring robust access control and data protection.

5. Protection Against Security Threats:

- Implement input validation and sanitization to defend against common threats like XSS and SQL injection, ensuring user inputs are safe and do not compromise the system.

6. User Awareness and Education:

- Provide users with guidelines for creating strong passwords and recognizing phishing attempts, helping them to manage their accounts securely.

7. Regular Security Audits:

- Conduct regular security audits and vulnerability assessments to identify and address potential weaknesses, keeping the application resilient against emerging security risks.

These measures would help ensure that the automatic timetable generation project remains secure, protecting both user data and the integrity of the scheduling system.

CODING

```
package com.max.quizspring.config;
import java.io.IOException;
import org.springframework.lang.NonNull;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
```

```

import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import com.max.quizspring.repo.JwtRepo;
@Component
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    private final JwtToken jwtTokenUtil;
    private final UserDetailsService userDetailsService;
    private final JwtRepo jwtRepo;
    @Override
    protected void doFilterInternal(@NonNull HttpServletRequest request, @NonNull
HttpServletResponse response,
        @NonNull FilterChain filterChain) throws ServletException, IOException {
        final String authHeader = request.getHeader("Authorization");
        final String token;
        final String username;
        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            filterChain.doFilter(request, response);
            return; }
        token = authHeader.substring(7);
        username = jwtTokenUtil.extractUsername(token);
        if (username != null && SecurityContextHolder.getContext().getAuthentication()
== null) {

```

```

        UserDetails userDetails =
this.userService.loadUserByUsername(username);
        var isValid = jwtRepo.findByToken(token).map(t -> !t.isExpired() &&
!t.isRevoked()).orElse(false);
        if (jwtTokenUtil.isValid(token, userDetails) && isValid) {
            UsernamePasswordAuthenticationToken authToken = new
UsernamePasswordAuthenticationToken(
                userDetails, null, userDetails.getAuthorities());
            authToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
            SecurityContextHolder.getContext().setAuthentication(authToken);
        }
    }
    filterChain.doFilter(request, response);
}
}

```


CHAPTER 8

CONCLUSION

This chapter tells about the conclusion that anyone can drive from the project and the learning we learnt by taking over this project.

8.1 CONCLUSION

The automatic timetable project successfully streamlines the scheduling process in educational institutions, reducing errors and conflicts associated with manual timetabling. The backend architecture of the automatic timetable generation project is meticulously designed to ensure a secure, efficient, and reliable system for managing and generating schedules. Central to this architecture is a robust security framework that addresses the protection of sensitive data and the integrity of system operations. User authentication is a cornerstone of the backend design, featuring secure registration and login processes that utilize email and hashed passwords.

8.2 FUTURE SCOPE

The future scope for the automatic timetable generation project includes integrating machine learning for optimized scheduling, enhancing user interfaces, and improving scalability. Further, it involves integrating with Learning Management Systems, offering personalized timetables, and enabling real-time updates and notifications. Advanced conflict resolution, comprehensive analytics, collaborative scheduling, and multilingual support will also be key. These enhancements will make the system more efficient, user-friendly, and adaptable to various educational contexts.

