

SRS DOCUMENT

2021

TIMETABLE MANAGEMENT SYSTEM

Under the Guidance of
PROF. Vijayshree Khedkar

SYMBIOSIS INSTITUTE OF TECHNOLOGY
(A CONSTITUTENT OF SYMBIOSIS INTERNATIONAL
UNIVERSITY)

Pune - 412115
2018-19



GROUP MEMBERS:-

1. Digha Jain - 19070122052
2. Divya Venkatesh - 19070122054
3. Harsh Chandekar - 19070122066

TABLE OF CONTENTS

| | | |
|-----------|---|----------|
| 01 | Introduction..... | 3 |
| | 1.1 Purpose 3 | |
| | 1.2 Document Conventions 3 | |
| | 1.3 Intended Audience 3 | |
| | 1.4 Reading Suggestions 3 | |
| | 1.5 Product Scope 4 | |
| | 1.6 References 4 | |
| | | |
| 02 | Overall Description..... | 5 |
| | 2.1 Product Perspective 5 | |
| | 2.2 Product Functions 5 | |
| | 2.3 User Classes and Characteristics 5 | |
| | 2.4 Operating Environment 6 | |
| | 2.5 Design and Implementation Constraints 6 | |
| | 2.6 Process Model 6 | |
| | 2.7 Assumptions and Dependencies 7 | |
| | | |
| 03 | Functional Requirements..... | 8 |
| | 3.1. Login 8 | |
| | 3.2. View 9 | |
| | 3.3. Logout 10 | |

| | | |
|-----------|--|-----------|
| 04 | Nonfunctional Requirements..... | 10 |
| | 4.1. Security Requirements | 10 |
| | 4.2. Performance Requirements | 11 |
| | 4.3. Availability | 11 |
| | 4.4. Maintainability | 11 |
| | 4.5. Hardware Requirements | 11 |
| | 4.6. Software Requirements | 11 |
| 05 | Analysis Models | 12 |
| | 5.1. Use-Case Diagrams | 12 |
| | 5.2. Class-Diagram | 13 |
| 06 | Implementation Screenshots | 14 |
| 07 | Test Cases | 19 |
| | 7.1. Test cases for login | |
| | 7.2. Test Cases for creating batch | |
| | 7.3. Test cases for adding professor | |
| | 7.4. Test cases for creating a course | |
| | 7.5. Test cases for assigning professor to batch | |
| 08 | Conclusion and Future Scope | 24 |
| 09 | Glossary | 26 |

INTRODUCTION

1.1. PURPOSE

The purpose of this Software Requirements Specification document is to provide a detailed overview of the Timetable Generation software, its parameters, and goals from both the user and admin end. It will explain the features of the system, the interface, the *constraints*, and how the Application will operate. The main aim is to design a timetable using the user data that will be ideal for time management.

1.2. DOCUMENT CONVENTIONS

This document is made to fit an A4 paper. **Blue** weighted text is used for the main headings and **Bold**-faced text has been used for the sub-headings. The *italicized* words are mentioned in the glossary at the end.

1.3. INTENDED AUDIENCE

The SRS is a formal document to be used by the programmer and the user to verify if all the requirements and the functionalities are present in the project. This Software is intended for both the faculty and the students to organize their timetables or schedules in a time-efficient manner. This document can also be used as a base-guide for future updates and additional features.

1.4. READING SUGGESTIONS

This document has been arranged approximately in order of increasing specificity. The overall description should be sufficient for the users and the *functionalities* and test- cases are added for the developers and learners who wish to pursue this project.

1.5. PRODUCT SCOPE

This system aims to enable a convenient and easy-to-use platform for timetable generation, access, and modification. It will have additional features to add the department, subject, semester, and faculty information.

It will be made to accommodate additional data for the timings and duration of lectures, and the classrooms allotted for the labs. Hence this project will be a very useful tool for academicians everywhere as a paperless timetable system that can be updated easily and is very user-friendly.

1.6. REFERENCES

- * Database Programming with JDBC and Java by O'Reilly
- * Head First Java 2nd Edition
- * <http://www.jdbc-tutorial.com/>
- * Java and Software Design Concepts by Apress
- * <https://www.tutorialspoint.com/java/>
- * <http://www.javatpoint.com/java-tutorial>
- * <https://docs.oracle.com/javase/1/tutorial/>
- * <http://www.wampserver.com/en/>
- * <http://www.JSP.net/>
- * <http://www.tutorialspoint.com/mysql/>
- * httpd.apache.org/docs/2.0/misc/tutorials.html
- * <https://docs.oracle.com/javase/7/docs/api/javax/swing>
- * <https://docs.oracle.com/javase/7/docs/api/javax/swing>
- * <https://zetcode.com/javaswing/firstprograms/>
- * <https://docs.oracle.com/javase/tutorial/uiswing/components>
- * <https://docs.oracle.com/javase/7/docs/api/javax/JDBC>

OVERALL DESCRIPTION

2.1. PRODUCT PERSPECTIVE

The timetable management system provides a common platform for everyone in the university to view the timetable allotted to them. It helps in reducing the load of manually collecting and distributing timetables. It also helps students and faculty to have their schedules on hand always.

2.2. PRODUCT FUNCTIONS

The basic product *features* include:

1. Logging in and using the site through different roles: Admin, Teacher, and Student.
2. Generation and viewing of the respective timetables.
3. Adding/modifying/deleting existing timetables, faculty details, subject details, timings, and room allocations.

2.3. USER CLASSES AND CHARACTERISTICS

- Admin - The person who has good knowledge of the database and technical expertise. He has the highest *privilege* the software. He is the only person allowed to make changes to the timetables. It is the least frequented role.
- Teacher- The second most important user. They are expected to have full information on the academic front to ensure smooth working. They are allowed to view their respective timetables.
- Student - The highest number of users for the site fall in this class. They can enter their credentials and view their timetables.

2.4. OPERATING ENVIRONMENT

The product can be used on various pc operating systems like Windows, Linux, etc. It is currently made using java and will be ready to be deployed in its future releases.

2.6. DESIGN AND IMPLEMENTATION CONSTRAINTS

Hard Constraints

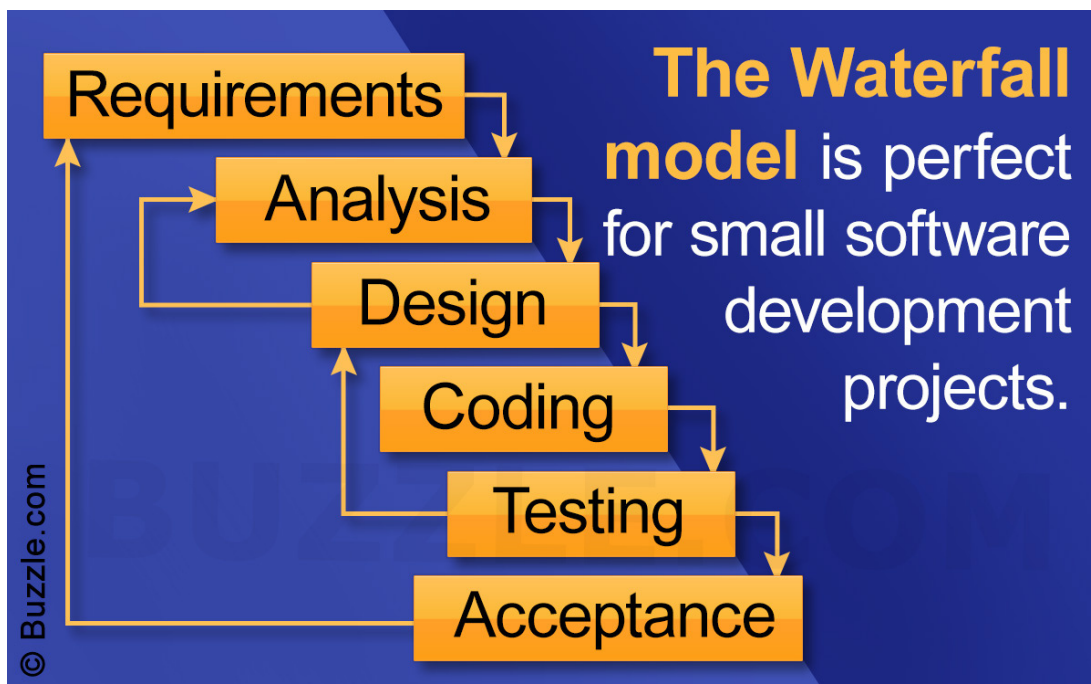
1. A slot is not assigned to more than one lecture at the same time.
2. An instructor cannot teach more than one class at the same time.

Soft Constraints

1. The lectures are not assigned to time slots, which are in the instructor's forbidden time zones.
2. Instructors' daily lecture hours should be restricted to be within the allowed maximum hours.

2.7. PROCESS MODELS

We have used the WATERFALL process model for this mini-project.



This model is chosen as the project is a relatively small one and the waterfall method works perfectly for small projects. The requirements are well-defined and reasonably stable which makes it a better fit. We expect a fully working model to be presented at the end of the project with future scope for improvement in it.

Since we are not really taking much user feedback and have a fixed deadline for the project, we went ahead with the Waterfall Model with frequent testing and *integration* to avoid major setbacks later.

2.8. ASSUMPTIONS AND DEPENDENCIES

This software assumes that the faculty communicate their needs correctly to the admin and he/she enters them into the system accordingly. It needs a stable internet connection at all times to function properly. This jar file for connecting our java project with the SQL backend will also be needed. The jar file (mysql-connector-java-8.0.23) can be downloaded from here: <https://dev.mysql.com/downloads/connector/j/>

FUNCTIONAL REQUIREMENTS

This section will cover all the *functional requirements* of the TIMETABLE MANAGEMENT SOFTWARE.

3.1. LOGIN

1. ADMIN

An admin is the main head of the authority and is responsible for any changes done into the timetable such as adding or removing the credentials of any faculty/student, change in the classroom number, allocating the number of eligible students for each class, in case a faculty is absent then doing the required changes.

An admin's view must be authenticated and be checked from the list of people present in the database.

INPUT: Username and Password

OUTPUT: If Any of the above-mentioned criteria differs from that present in the database then ask to retry else LOGGED IN SUCCESSFULLY!

2. TEACHER

A teacher is someone who can view the timetable. So, the teacher should be registered.

INPUT: Enter registered Username and Password

OUTPUT: LOGGED IN SUCCESSFULLY!

3. STUDENT

A student will be able to view their timetable.

INPUT: Enter registered Username and Password

OUTPUT: If Any of the above-mentioned criteria differs from that present in the database then ask to retry else LOGGED IN SUCCESSFULLY!

3.2. VIEW

1.ADMIN

Create a timetable

From the menu, select create tab and press enter.

Input: Number of subjects, subject name, corresponding teacher, number of slots per subject time, classroom number, and number of students. Select the proceed tab and press enter.

Output: The new timetable satisfying all the constraints.

Modify and view a timetable

From the menu, select modify tab and press enter.

Input: Select the corresponding slot and class and enter the modifications such as teacher name or classroom number. Select the proceed tab and press enter.

Output: The modified timetable

Delete a timetable

Select the Delete tab from the menu and press enter. A pop-up will be displayed asking for confirmation. Select yes to proceed, No to cancel.

Accessing and updating the database

Input: Selecting options such as modify the timetable.

Output: Updating the timetable with new faculties, subjects, Classrooms and slots.

2. TEACHER

View timetable

Input: Select the View tab from the menu and press enter

Output: The timetable of the individual along with their respective classes, classroom number and number of students.

3. STUDENT

View Timetable

Input: Select the View tab from the menu and press enter

Output: The timetable of the individual along with their respective subject teachers, classroom number and number of students.

3.3. LOGOUT

Click on the logout button to successfully log out of the website.

NON-FUNCTIONAL REQUIREMENTS

4.1. SECURITY REQUIREMENTS

Only the Admin will have access to make sensible changes such as the number of slots, corresponding teachers, Time table creation and modification, etc. All the clients: Admin, Teacher, Student must have legitimate mail ID and password to log in, hence the framework is quite secure as it will give access to approved users only.

The database can be modified only and only by the approved admin. Also, the database is encrypted by SSL/TLS key *encryption*.

4.2. PERFORMANCE REQUIREMENTS

The response to the action must be quick. The flow of data between the internet server and Database must be encrypted to make sure that there is no possible loss in data and data cannot be modified.

4.3. AVAILABILITY

The software will be made available in the form of a website 24 X 7 and the client will be accessing it on an average internet speed. An error message appears when something goes wrong to avoid availability problems.

4.4. MAINTAINABILITY

No special maintenance will be required. Only and Only the admin will have to update the database, in case changes need to be done in the timetable.

4.5. HARDWARE REQUIREMENTS

The website can be viewed on any Desktop computer, laptop, tablet, or phone having mobile data/connected to broadband with average internet speed.

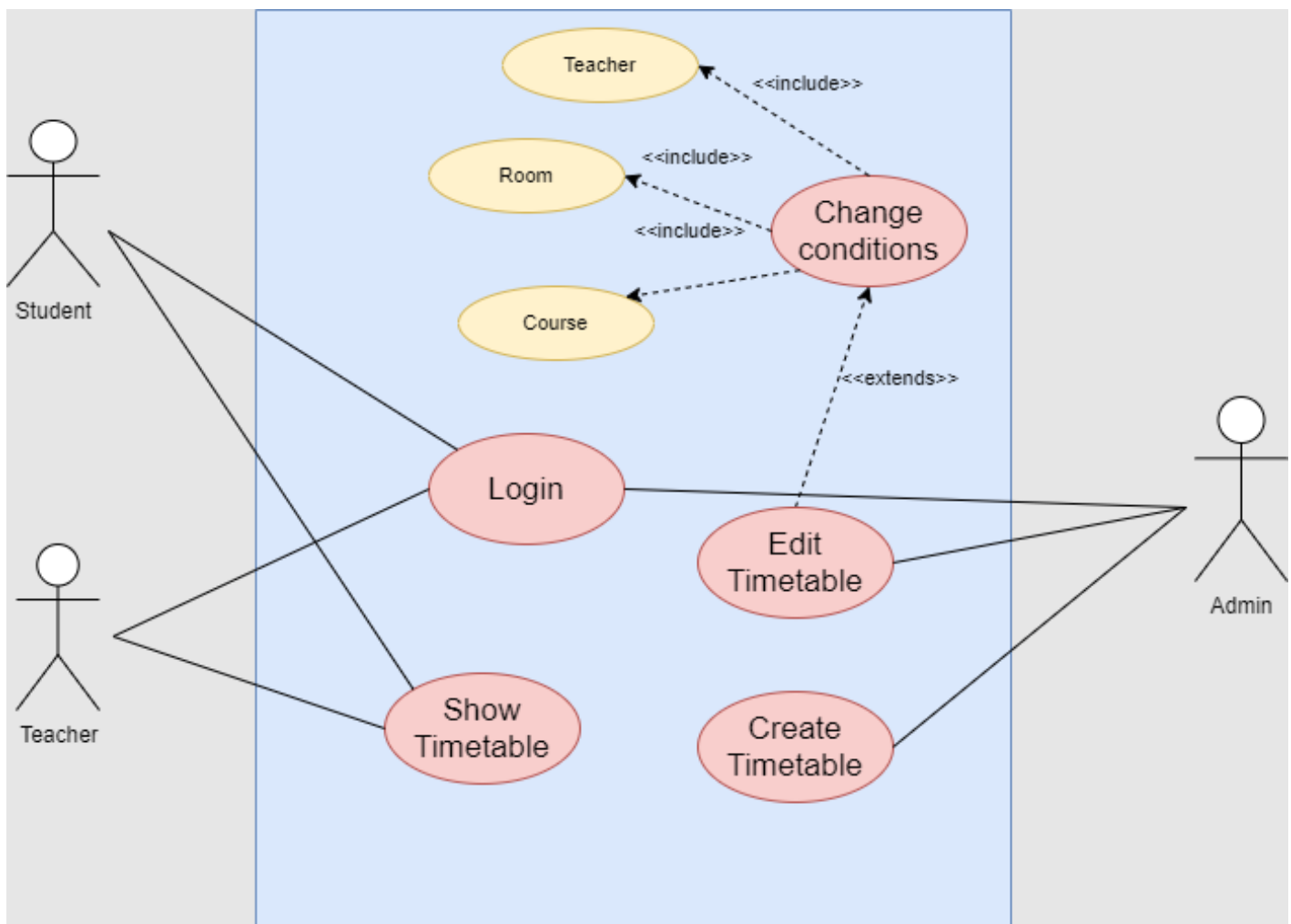
4.6. SOFTWARE REQUIREMENTS

A web browser should be available with the client. Any operating system can be used. The current version of the java program can be run using any java compiler software and a database connection.

ANALYSIS MODELS

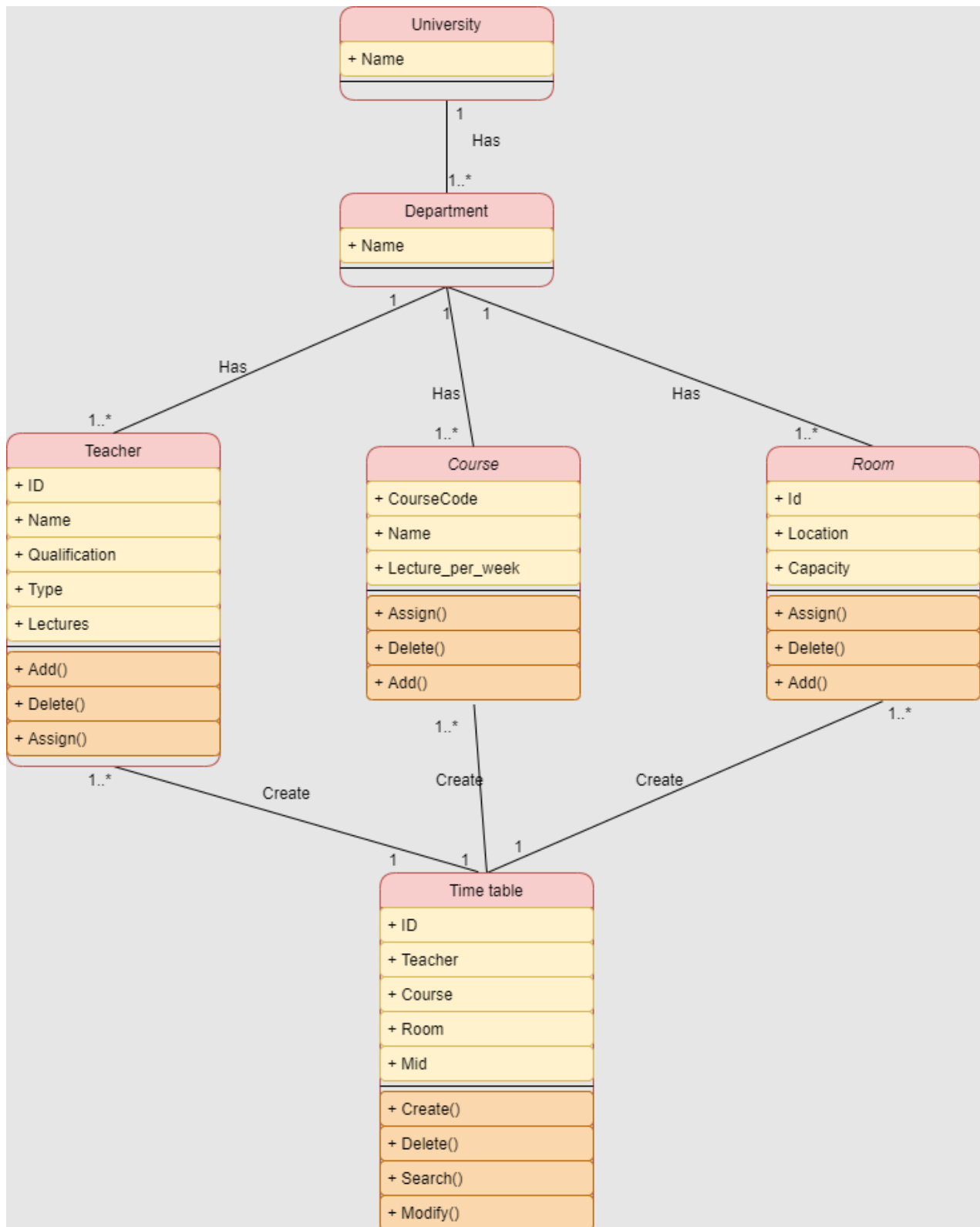
5.1. USE - CASE DIAGRAM

Use case diagrams are usually referred to as behavior diagrams used to describe a set of use cases that some system should or can perform in collaboration with one or more external users of the system.



5.2. CLASS DIAGRAM

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.



IMPLEMENTATION SCREENSHOTS

Login Frame

Timetable Management Software

☒ Admin ☐ Teacher ☐ Student

Username:

Password:

☐ Show Password

HOME VIEW Create Batch Create Course Create Professor Delete LOGOUT

View Timetable

| | | | | | | | | |
|-----------|--|--------------|--------------|--|--|--|--------------|--------------|
| Monday | | | music(reena) | | | | music(reena) | |
| Tuesday | | music(reena) | | | | | | |
| Wednesday | | | | | | | | |
| Thursday | | | | | | | | |
| Friday | | | | | | | | music(reena) |
| Saturday | | | | | | | | |

Batch: Professor:

IMPLEMENTATION SCREENSHOTS

[HOME](#) [VIEW](#) [Create Batch](#) [Create Course](#) [Create Professor](#) [Delete](#) [LOGOUT](#)

New Batch

Batch Name: [Add](#)

Courses and Professors

Select Course Select Professor [Add](#)


[HOME](#) [VIEW](#) [Create Batch](#) [Create Course](#) [Create Professor](#) [Delete](#) [LOGOUT](#)

Course Details:

Course Name:

Credits:

[Add](#)



IMPLEMENTATION SCREENSHOTS

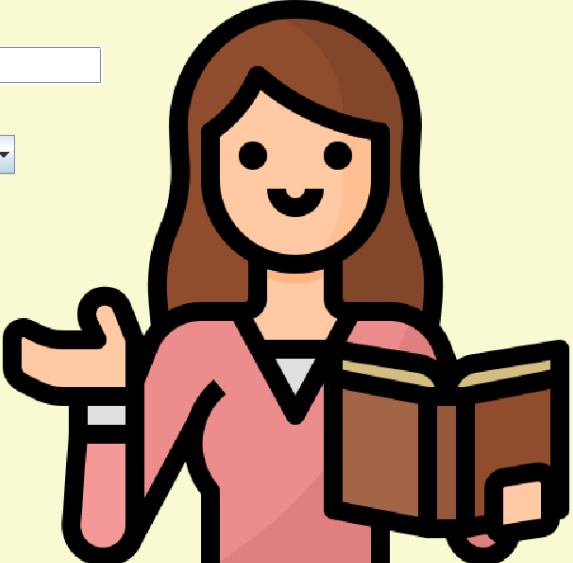
[HOME](#) [VIEW](#) [Create Batch](#) [Create Course](#) [Create Professor](#) [Delete](#) [LOGOUT](#)

New Professors

Name:

Courses:

[Add](#)



[HOME](#) [VIEW](#) [Create Batch](#) [Create Course](#) [Create Professor](#) [Delete](#) [LOGOUT](#)

Delete

Professor: [Delete](#)

Batch: [Delete](#)



The full code of the software can be found on the shared drive link. Here we are attaching some main code snippets that went into making this project.

```
1 package TIMETABLE_MANAGEMENT;
2 import java.sql.*;
3
4 public class SQLconnect
5 {
6     String d[][] = new String[6][8];
7     static final String DB_URL = "jdbc:mysql://localhost/Timetables";
8     static final String USER = "root";
9     static final String PASS = "root";
10    ArrayList<String> old_Professors = new ArrayList<String>();
11    ArrayList<String> old_Batches = new ArrayList<String>();
12    Map<String,Integer> old_courses = new HashMap<String,Integer>();
13
14    SQLconnect()
15    {
16        try(Connection conn = DriverManager.getConnection(DB_URL,USER,PASS))
17        {
18            Statement stmt = conn.createStatement();
19            if(!tableExists("PROFESSOR"))
20            {
21                String sql ="CREATE TABLE PROFESSOR ( ID INTEGER not NULL AUTO_INCREMENT, " + " NAME VARCHAR (255), "+ " PRIMARY KEY ( ID ))";
22                stmt.executeUpdate(sql);
23            }
24            else
25            {
26                String sql = "SELECT * FROM PROFESSOR ";
27                ResultSet res = stmt.executeQuery(sql);
28                //ResultSetMetaData metaData = res.getMetaData();
29                while(res.next())
30                {
31                    old_Professors.add(res.getString("NAME"));
32                }
33                res.close();
34            }
35        }
36        if(!tableExists("BATCH"))
37        {
38            String sql ="CREATE TABLE BATCH ( ID INTEGER not NULL AUTO_INCREMENT, "+" NAME VARCHAR (255), "+ " PRIMARY KEY ( ID ))";
39            stmt.executeUpdate(sql);
40        }
41    }
42 }
```

```
1 package TIMETABLE_MANAGEMENT;
2
3 public class timetableee
4 {
5     String a[][] = new String[6][8];
6     private String batch_name;
7
8     void setBatch_name(String s)
9     {
10         this.batch_name = s;
11     }
12     String getBatch_name()
13     {
14         return this.batch_name;
15     }
16
17
18
19 }
```

```

116     JPanel home = new JPanel();
117     home.setBorder(new LineBorder(new Color(0, 0, 0), 3));
118     home.setBackground(new Color(128, 0, 0));
119     home.setBounds(250, 50, 850, 200);
120     home.setLayout(null);
121
122     JLabel lblNewLabel_1h = new JLabel("The ");
123     lblNewLabel_1h.setForeground(new Color(255, 255, 255));
124     lblNewLabel_1h.setFont(new Font("Old English Text MT", Font.BOLD, 50));
125     lblNewLabel_1h.setHorizontalAlignment(SwingConstants.CENTER);
126     lblNewLabel_1h.setBounds(323, 69, 218, 83);
127     home.add(lblNewLabel_1h);
128
129     JLabel Timetable_Management = new JLabel("Timetable Management");
130     Timetable_Management.setForeground(new Color(255, 255, 255));
131     Timetable_Management.setHorizontalAlignment(SwingConstants.CENTER);
132     Timetable_Management.setFont(new Font("Old English Text MT", Font.BOLD, 50));
133     Timetable_Management.setBounds(124, 132, 616, 83);
134     home.add(Timetable_Management);
135
136     JLabel System = new JLabel("System");
137     System.setForeground(new Color(255, 255, 255));
138     System.setHorizontalAlignment(SwingConstants.CENTER);
139     System.setFont(new Font("Old English Text MT", Font.BOLD, 50));
140     System.setBounds(336, 194, 218, 83);
141     home.add(System);
142
143     JLabel lblNewLabel = new JLabel("");
144     lblNewLabel.setBorder(new MatteBorder(1, 1, 1, 1, (Color) new Color(0, 0, 0)));
145     lblNewLabel.setIcon(new ImageIcon("C:\\Users\\Divya\\eclipse-workspace\\Login\\Images\\study.jpg"));
146     lblNewLabel.setBounds(194, 310, 490, 272);
147     home.add(lblNewLabel);
148
149     JPanel view = new JPanel();
150     view.setBackground(new Color(255, 218, 185));
151     view.setForeground(new Color(255, 218, 185));
152     view.setBounds(345, 10, 744, 628);
153     view.setLayout(null);
154
155     JComboBox view_batch_v = new JComboBox(all_batches);
156     view_batch_v.setBounds(149, 440, 126, 33);
157     view.add(view_batch_v);

```

The entire project was divided into 6 main classes which managed the various functionalities mentioned in the document. The classes were: Frontend, Batch, Professor, Course, SQLConnect, and Timetable.

TEST CASES

7.1. Test cases for login

This module describes the login module. If there is a valid user account, it proceeds to the next activity otherwise an error occurs.

| Test Objective | Precondition | Steps | Test Data | Expected Result | Actual Result | Test Status |
|----------------|--------------|--|--|-------------------------|-------------------------|-------------|
| User Login | A valid user | 1. Choose correct option from admin, teacher or student. 2. Enter a username 3. Enter a Password | Username: "admin" Password: "admin" | Logged in successfully! | Logged in successfully! | Passed |
| User Login | A valid user | 1. Choose correct option from admin, teacher or student. 2. Enter a username 3. Enter a Password | Username: "random" Password: "random" | Logged in successfully! | Login Failed! | Failed |
| User Login | A valid user | 1. Choose correct option from admin, teacher or student. 2. Enter a valid username 3. Enter a valid Password | Username: "teacher" Password: "teacher" | Logged in successfully! | Logged in successfully! | Passed |

TEST CASES

| | | | | | | |
|------------|--------------|--|---|-------------------------|-------------------------|--------|
| User Login | A valid user | 1. Choose correct option from admin, teacher or student. 2. Enter a username 3. Enter a Password | Username: <u>"teena"</u> Password: <u>"tina"</u> | Logged in successfully! | Login Failed! | Failed |
| User Login | A valid user | 1. Choose correct option from admin, teacher or student. 2. Enter a username 3. Enter a Password | Username: "student" Password: "student" | Logged in successfully! | Logged in successfully! | Passed |
| User Login | A valid user | 1. Choose correct option from admin, teacher or student. 2. Enter a username 3. Enter a Password | Username: "xyz" Password: <u>"abc"</u> | Logged in successfully! | Login Failed! | Failed |

TEST CASES

7.2. Test Cases for creating batch

| Test Objective | Preconditions | Steps | Test Data | Expected Output | Actual Output | Result |
|----------------|---------------------|--------------------|---|------------------------------|---|--------|
| Batch Creation | A unique batch name | Enter a batch name | "CSA" (if a new unique batch name) | "Batch created successfully" | "Batch created successfully" | Passed |
| Batch Creation | A unique batch name | Enter a batch name | "CSA" (if a batch with same name already exists) | "Batch created successfully" | "A batch with same name already exists" | Failed |

7.3. Test cases for adding professor

Note: One teacher can teach only one subject.

| Test Objective | Preconditions | Steps | Test Data | Expected Output | Actual Output | Result |
|------------------|-------------------------|---|---|--------------------------------|---|--------|
| Adding Professor | A unique professor name | 1.Enter a professor name 2.Enter course to be assigned | "RM" (if a new unique professor name) "maths" | "Professor added successfully" | "Professor added successfully" | Passed |
| Adding Professor | A unique professor name | 1.Enter a professor name 2.Enter course to be assigned | "RM" (if a professor with same name already exists) "maths" | "Professor added successfully" | "A Professor with same name already exists" | Failed |

TEST CASES

7.4. Test cases for creating a course

| Test Objective | Preconditions | Steps | Test Data | Expected Output | Actual Output | Result |
|-----------------|----------------------|--|---|-------------------------------|--|--------|
| Course Creation | A unique course name | 1.Enter a course name 2.Enter number of credits | "Maths" (if a new unique course name) "3" | "Course created successfully" | "Course created successfully" | Passed |
| Course Creation | A unique course name | 1.Enter a course name 2.Enter number of credits | "Maths" (if a course with same name already exists) "3" | "Course created successfully" | "A Course with same name already exists" | Failed |

TEST CASES

7.5. Test cases for assigning professor to batch

| Test Objective | Preconditions | Steps | Test Data | Expected Output | Actual Output | Result |
|---|---|---|---|---|---|--------|
| Assigning course and professor to a batch | 1.Select an existing course name, professor and course. 2.The course assigned to professor should be the one assigned to them at the time of their creation. | 1. Enter a batch name, professor and teacher. | "CSA" "RM" "MATHS" | Professor assigned to batch successfully. | Professor assigned to batch successfully. | Passed |
| Assigning course and professor to a batch | 1.Select an existing course name, professor and course. 2.The course assigned to professor should be the one assigned to them at the time of their creation. | 1. Enter a batch name, professor and teacher. | "CSB" (batch doesn't exist) or "RM" "English" Course assigned to teacher differs. | Professor assigned to batch successfully. | "Batch does not exist" Or "Professor initially was not assigned to this course" | Passed |

CONCLUSION & FUTURE SCOPE

CONCLUSION

The timetable management system is a java based web application which is used to create timetables for university. The objective of this software is to practically manage real-time problems of creating timetables.

- Description of all functional and *non-functional requirements* with use cases and class diagrams
- Designed *interface* for user compatibility of software
- Described system specifications for software and actions

Future Scope of the Project

In a nutshell, it can be summarized that the future scope of the project circles

around maintaining information regarding:

- * We can add a printer in the future.
- * We can give more advanced software for Timetable Management System including more facilities
- * We will host the platform on online servers to make it accessible worldwide
- * Integrate multiple load balancers to distribute loads of the system
- * Create the master and slave database structure to reduce the overload of the database queries
- * Implement the *backup* mechanism for taking backup of codebase and database on regular basis on different servers

CONCLUSION & FUTURE SCOPE

PROJECT OUTCOMES

This mini-project being our first attempt at making a fully working and executable project taught us a lot of things in the making. The project was successfully completed within the allotted time limit and these are the outcomes we gained from it.

1. It makes the timetable viewing experience amazing and quick.
2. Abstraction is provided in the System (ie. student /teacher cannot edit the timetable)
3. Implemented backend with SQL server.
4. Implemented frontend using JFrame and TabbedPane.
5. A simple modulo algorithm is used to make code light and optimal for small Applications.
6. Learned about all the technologies such as SQL, JFrame and implemented them successfully.
7. Brainstormed and discussed the issues we faced and successfully overcome them.
8. Gained experience on how to create a project and make the proper user interface for it.
9. Successfully Integrated all the technology and made the project executable in time.

GLOSSARY

Below, is a list of names and words of those committed to this project, such as:

- **Constraint** - A constraint is a limiting factor that slows a system down or prevents it from achieving its goal. The two types of constraints discussed here are:
 - a. **Hard Constraint** - A hard constraint is a constraint that must be satisfied by any feasible solution to the model. Hard constraints are more rigid than soft constraints.
 - b. **Soft Constraint** - The constraints that are allowed to be violated to a certain extent are called soft constraints
- **Features** - Features are the “tools” you use within a system to complete a set of tasks or actions.
- **Functionality** - Functionality is how the features work to provide you with the desired outcome.
- **Privilege** - In computing, privilege is defined as the delegation of authority to perform security-relevant functions on a computer system. A privilege allows a user to perform an action with security consequences.
- **Compatibility** - Software compatibility is a characteristic of software components or systems which can operate satisfactorily together on the same computer, or on different computers linked by a computer network.
- **Integration** - Software integration is the process of bringing together various types of software sub-systems so that they create a unified single system.

GLOSSARY

- **Requirements** - The software requirements are the description of features and functionalities of the target system. The two main types of requirements discussed here are:
 - **Functional Requirements** - A functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs.
 - **Non-functional Requirements** - Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They ensure the usability and effectiveness of the entire system.
- **Encryption** - Encryption is the process of taking plain text, like a text message or email, and scrambling it into an unreadable format called “cipher text.” This helps protect the confidentiality of digital data either stored on computer systems or transmitted through a network.
- **Interface** - When referring to software, an interface is a program that allows a user to interact with computers in person or over a network. An interface may also refer to controls used in a program that allows the user to interact with the program.
- **Backup** - A backup or data backup is a copy of computer data taken and stored elsewhere so that it may be used to restore the original after a data loss event.

Project Contributions

Digha Jain: SQL and Backend

Divya Venkatesh: Frontend and Designing

Harsh Chandekar: Algorithm and Integration