Design Document

Class to Classroom Scheduling System

Authors:
Elif Sude Özmen
Nesibe Nur Pekçakar
Aleyna Kök
Emir Aydın

Date: November 2024

Contents

1 Introduction **2**

2 Software Architecture **2**
2.1 Structural Design 2
2.2 Behavioral Design 4
2.2.1 Changing Class-to-Room Assignments 4
2.2.2 Adding New Classes or Project Meeting Hours 5
2.2.3 Viewing a Student's Class Schedule 6
2.2.4 Providing a Warning for Full Capacity 6
2.2.5 Viewing a Classroom's Schedule and Courses 6
2.2.6 File Parsing and Data Extraction 7
2.2.7 Automatic Course-to-Classroom Assignment 8
2.2.8 Generating Schedules by Course or Classroom 9
2.3 Help Menu and User Guide 10

3 Graphical User Interface **10**
3.1 Edit Matching 11
3.2 Download as a File 11
3.3 View Schedule 11
3.4 Additional Features 11
3.5 View Classroom Schedule 13
3.6 View Student Schedule 13

1 Introduction

Class to Classroom Scheduling System is a standalone desktop application designed to facilitate efficient class and classroom scheduling for educational institutions. The system automates the assignment of classrooms to courses, allows for flexible modifications like switching the automated matching when necessary, adding new project meeting classes, looking up a student's schedule, adding and removing students to classes. The application is targeted for Windows desktop environments and will operate without requiring an internet connection. The software app will be distributed as an executable file for systems that have Windows operating system. The software will be in English. Throughout the document, it is assumed that Java will be used as the primary programming language for implementation.

These meet the following non-functional requirements:

- **Non-Functional Requirement 2:** The app should be provided as an executable file (.exe) for installation and its execution.
- **Non-Functional Requirement 3:** The system shall be designed for the Windows operating system.
- **Non-Functional Requirement 4:** The system shall be operational without the need for an internet connection.

2 Software Architecture

The Class to Classroom Scheduling System will have a modular software architecture. It will consist of a back-end system that handles class-to-room assignment logic, schedule generation, and occupancy tracking, along with a graphical user interface (GUI) for user interactions. The core architecture will focus on parsing input data from given files, managing relationships between courses, classrooms, and students, and providing visual representations of schedules.

2.1 Structural Design

The system relies on the fundamental relationship between three core entities: Class, Classroom, and Student. Relationships between these objects define the scheduling and resource management operations.

- **Class:** Represents a course or session. Holds details such as the course name, assigned classroom, schedule, and a list of enrolled students.
- **Classroom:** Represents a physical room. Contains attributes like room capacity, current occupancy, and availability times.
- **Student:** Represents an individual student. Contains personal information and a list of their enrolled classes.

Suppose a new class, `"Algorithms 101"`, needs to be added to the system. The system creates a Class object:

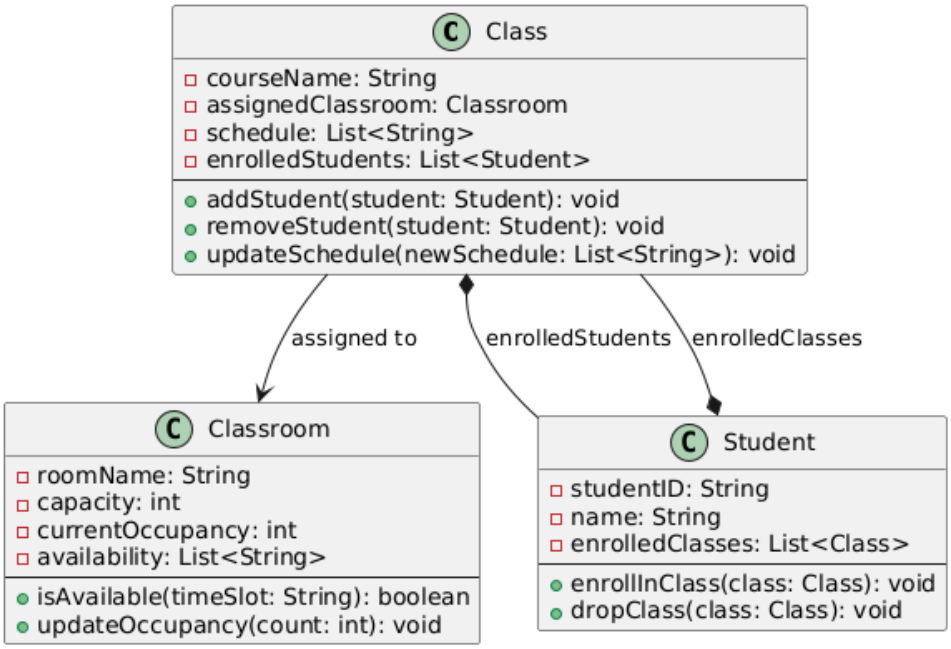

Figure 1: Relationship Between Three Fundamental Classes

```
Class algorithms101 = new Class("Algorithms 101", null, new Schedule(), new ArrayList
```

A Classroom object is assigned to the class based on availability and capacity:

```
Classroom roomA = new Classroom("Room A", 40, availableTimes);
algorithms101.setClassroom(roomA);
```

Students can be added to the class, ensuring capacity limits are respected:

```
algorithms101.addStudent(student1);
```

The `Class` object is linked to the `Student` objects' schedules to prevent overlapping sessions. Also, the system will read the file provided by the user that holds current classes and their enrolled student list and automatically generates the schedule using these three classes.

These classes and relationships between these classes meet the following functional requirements:

- **Functional Requirement 1:** The system shall enable the user to view automatic class-to-room assignments.
- **Functional Requirement 5:** The user shall be able to add/remove students to/from a class.
- **Functional Requirement 6:** The user shall be able to view available classrooms and their free time slots.
- **Functional Requirement 7:** The user shall view occupancy rates of courses based on room capacity and enrollment.

In Figure 2 2, the UML class diagram shows the three main classes: `Class`, `Classroom`, and `Student`, along with other needed classes like `Schedule` for every class and student, their relationships, a `FileParser` class for uploaded files, and `RoomBooking` for class-to-classroom matching.

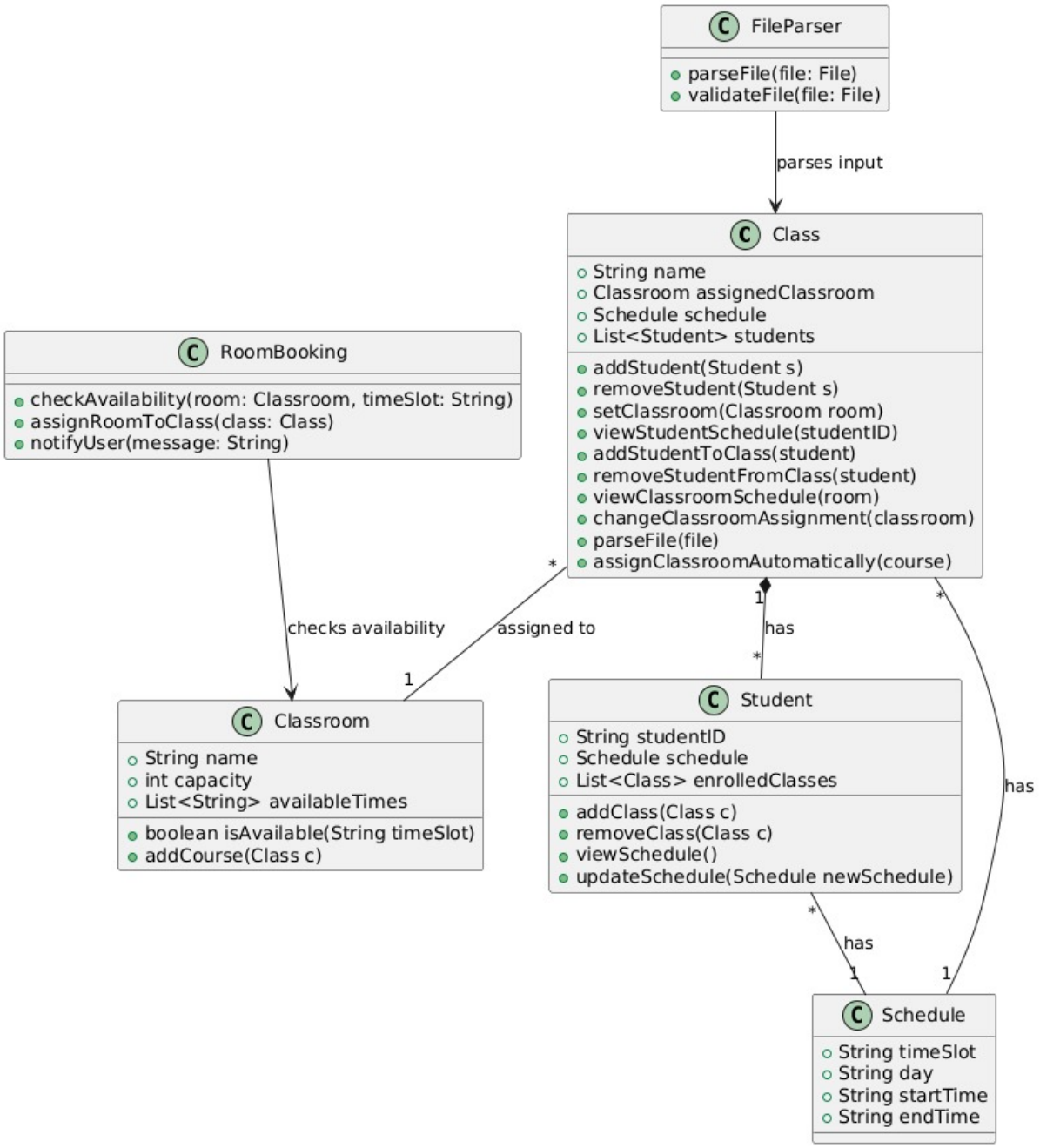

Figure 2: Class Diagram of the Project

2.2 Behavioral Design

The procedures, methods, and algorithms needed to meet the system's functional requirements are described in this section.

2.2.1 Changing Class-to-Room Assignments

Users must be able to manually change class-to-room assignments, according to Functional Requirement 2. The user can designate a new room and choose a class using the system. Following the selection, the system confirms that the selected room is available and ensures that there are no conflicts with other classes that are booked. It also makes

sure that the number of students in the class is equal to the room's capacity. The system gives the user an error message and lets them make changes if there are any issues, like time overlaps or insufficient capacity. Functional Requirement 2 will be satisfied by this feature.

Functional Requirement 2: The user shall be able to change class-to-room assignments.

As an example, let's say that a class called "Math 101" has 25 enrolled students and is currently scheduled to meet in Room A from 9:00 to 10:30. The system will first determine whether Room B is available between 9:00 and 10:30 if the user chooses to move this class there. The system will show an error message like:

> *"Room B is not available or has insufficient capacity for Math 101. Please pick a different room."*

2.2.2 Adding New Classes or Project Meeting Hours

Functional Requirement 3 states that users shall be able to insert new classes, project meetings, or hours. The system allows users to enter the class name, desired time slot, and room through an intuitive interface. Once submitted, the system checks the room availability, conflicts in schedule, and whether the room has enough capacity for the class size. Once everything is validated, the new session details are saved, and the room and time slot are marked as occupied.

Functional Requirement 3: The user shall be able to add a new class, project meeting, or hour.

Example: First Scenario: The user wants to schedule a new project meeting.

- **Class Name:** "Software Project Turing"
- **Desired Time Slot:** 14:00 - 15:00
- **Room:** Room C

System Process:

- The system checks Room C's schedule for 14:00 - 15:00 and finds it available.
- It verifies that Room C's capacity (e.g., 50) is sufficient for the meeting's expected participants (e.g., 30).
- The system ensures there are no conflicts with other classes or meetings for the instructor or students involved.

Result: After all checks, the system inserts the meeting into the schedule, marks Room C as occupied for 14:00 - 15:00, and saves the details in the database. A confirmation message is displayed to the user:

> *"The meeting 'Software Project Turing' has been successfully scheduled in Room C from 14:00 to 15:00."*

Error Scenario: If Room C is already reserved for the chosen time slot, the user would receive an error notice from the system:

> *"Room C is not available from 14:00 to 15:00. Please choose a different room or time slot."*

2.2.3 Viewing a Student's Class Schedule

Functional Requirement 4 states that users shall be able to view a student's class schedule. The system provides a method named `viewStudentSchedule(String studentID)` to handle this functionality. The method retrieves the student's schedule from the database using the provided `studentID`. The schedule includes all classes and project meetings assigned to the student, displayed with class names, times, and room assignments. This ensures users can plan additional classes or meetings without creating scheduling conflicts.

Example: If the `viewStudentSchedule("12345")` method is called for a student with the ID 12345, and the student is enrolled in:

- **Class Name:** "Math 101" **Time Slot:** 09:00 - 10:30 **Room:** Room A
- **Class Name:** "Physics 102" **Time Slot:** 11:00 - 12:30 **Room:** Room B

The method returns the following schedule:

```
Math 101 - Room A - 09:00 to 10:30 Physics 102 - Room B - 11:00 to 12:30
```

If there are no classes, the method returns a message such as:

"No classes found for student with ID: 12345."

This functionality ensures clear visibility of a student's schedule, meeting Functional Requirement 4.

Functional Requirement 4: The user shall be able to view a student's class schedule.

2.2.4 Providing a Warning for Full Capacity

If the user tries to register a student for a course that is already full, the system will alert them. The system compares the number of enrolled students with the available classroom capacity when the user starts the enrollment process. The course information that the system has retrieved is used for this comparison.

The enrollment is stopped and a warning message is displayed by the system if the number of enrolled students equals or surpasses the classroom's capacity. The warning ensures that no more students can be added, upholding the boundaries of the classroom.

This functionality is implemented through the `enrollStudent` method. While this method allows the enrollment process to proceed when there is enough capacity, it prevents over-enrollment by triggering a warning message when the classroom limit is exceeded. This ensures compliance with classroom capacity constraints.

Functional Requirement 8: The system shall provide a warning if the user attempts to add a student to a course that has reached full capacity.

2.2.5 Viewing a Classroom's Schedule and Courses

Viewing the schedule and list of courses allocated to a particular classroom will be possible through the system. The interface allows users to choose a classroom, and the system retrieves the relevant schedule from the database. All assigned courses, their time slots, and occupancy information are included in the schedule. To make it simple for users to examine how the classroom is currently being used, the obtained data is presented in an

organised, program-like style, such as a table or calendar view. This makes it possible for users to find open time slots and decide wisely when to book more sessions.

This functionality is implemented through the `viewClassroomSchedule` method. This method revokes all course assignments for the selected classroom and organizes them for clear visualization. The format ensures the user can quickly understand the room's availability and occupancy.

Functional Requirement 9: The user shall be able to view the schedule and list of courses for a specific classroom in a structured, program-like format.

2.2.6 File Parsing and Data Extraction

Our system's scheduling and resource management features allow for smooth user integration with input files that contain course, student, and classroom data.

Behavioral Flow:

1. **File Upload:**
   - The user selects an input file (e.g., CSV format) via the GUI.
   - The system validates the file format and schema (e.g., required columns like "Course Name," "Classroom ID," "Capacity").
2. **Data Parsing:**
   - The system reads the file line by line.
   - Extracted data is mapped to internal objects (e.g., `Course`, `Student`, `Classroom`).
   - Invalid entries trigger warnings, specifying row numbers and issues (e.g., *"Missing capacity for Classroom 102"*).
3. **Data Storage:**
   - Parsed data is stored in memory for runtime operations.
   - Errors are logged for troubleshooting and displayed in the GUI.

Key Features:

- **Scalability:** The system efficiently processes large datasets without significant delays.
- **Extensibility:** The parser is adaptable to future changes in file structure by modifying the schema configuration.

Example: The uploaded file contains:

```
Course Name, Classroom ID, Capacity, Start Time, End Time, Enrolled Students
Math 101, Room A, 30, 09:00, 10:30, 25
Physics 102, Room B, 40, 11:00, 12:30, 35
```

The system parses this data, creating objects:

- `Course("Math 101", "Room A", 30, "09:00-10:30", 25);`
- `Classroom("Room A", 30);`

Functional Requirement 10: The system will parse each given file and generate algorithms based on the necessary extracted information.

2.2.7 Automatic Course-to-Classroom Assignment

Our solution optimises resource performance and availability by automating the process of allocating courses to classrooms according to predetermined rules.

Behavioral Flow:

1. **During initialization:**
   - The system loads parsed data into memory, including courses and classrooms.
   - Classrooms are categorized according to their capacity and availability.
2. **Assignment Logic: Per course:**
   - Match course requirements (e.g., capacity, specific equipment) with available classrooms.
   - Check time slots to avoid scheduling conflicts.
   - Assign the course to the first available classroom that meets all criteria.
3. **Conflict Resolution:**
   - Notify the user and provide conflict facts if no classroom fits the standards.
   - Suggest alternative ways, such as rescheduling or splitting the class into small groups.
4. Dynamic updates to occupancy rates and classroom scheduling as assignments are completed.

Example Assignment Process:

- **Course:** "Chemistry 201," 35 students, requires a lab.
- **Available Classrooms:**
  - Room A (capacity: 30, no lab).
  - Room B (capacity: 40, lab available).
- **Assignment:** "Chemistry 201" is assigned to Room B.

Key Gains:

- Reduces manual workload by automating routine assignments.
- Ensures optimal utilization of classrooms and resources.

Functional Requirement 11: The system will automatically assign courses to suitable classrooms based on predefined criteria, such as course requirements, classroom capacity, and schedule relevance.

2.2.8 Generating Schedules by Course or Classroom

Our system has an extensible interface for building bespoke timetables in response to consumer questions.

Behavioral Flow:

1. **Schedule by Course:**
    - Users enter a course name (e.g., "Math 101").
    - The system obtains the course's designated classroom, time slot, and number of enrolled students.
    - The schedule is presented in a systematic manner, showing any potential conflicts or capacity difficulties.
2. **Schedule by Classroom:**
    - The user chooses a classroom (e.g., "Room B").
    - The system gets all courses allocated to that room, including time slots and occupancy rates.
    - A weekly timetable is prepared, displaying both open and occupied times.
    - The occupancy levels are colour-coded.

Example Scenarios: Generating a Course Schedule:

- **Input:** "Physics 102."
- **Output:**
    - Course: Physics 102
    - Classroom: Room B
    - Time: Monday 11:00-12:30
    - Enrolled Students: 35/40

Generating a Classroom Schedule:

- **Input:** "Room B."
- **Output:**
    - Room B Schedule:
        * Monday 09:00-10:30 - Math 101 (25/30)
        * Monday 11:00-12:30 - Physics 102 (35/40)

Functional Requirement 12: The system will allow users to generate schedules and programs by specifying either a course or a class name.

2.3 Help Menu and User Guide

This requirement ensures that users can use the product efficiently with minimal onboarding and an easily available help menu.

Behavioral Flow:

- The user clicks "Help" in the menu bar or toolbar.
- The system displays a structured help menu:
  - **Introduction:** Overview of system functionalities.
  - **Step-by-Step Tutorials:** Instructions for tasks like adding classes, generating schedules, or resolving conflicts.
  - **FAQ Section:** Common troubleshooting scenarios.
- The help menu includes search functionality for quick access to specific topics.

Additional Features:

- **Tooltips:** When you hover over buttons and fields, a quick explanation of their purpose appears.
- **Offline Accessibility:** The software comes with a help guide that can be accessed offline without an internet connection.
- **Illustrations & Visual Aids:** For clarity, there are screenshots and samples in the help menu.

Non-Functional Requirement 1: The user will be able to use the system after looking to the help menu.

3 Graphical User Interface

The interface design of the project provides easy classroom assignments for the associated lectures, access to all students' weekly schedules, and adjustments considering the capacity of the classes.

The user interface is planned to follow modern user interface guidelines. The main page will have a menu bar with the following actions:

- **Upload a File:** This button will allow users to upload a file for the app to store all the necessary information and handle suitable assignments.
  - **Functional Requirement 13:** The system will process and execute its operations for any given file format that meets the predefined data structure and content requirements.
- **Help Menu:** The interface will have a comprehensive help menu for the users.
- **About the App:** Provides detailed information about the goal of the app.

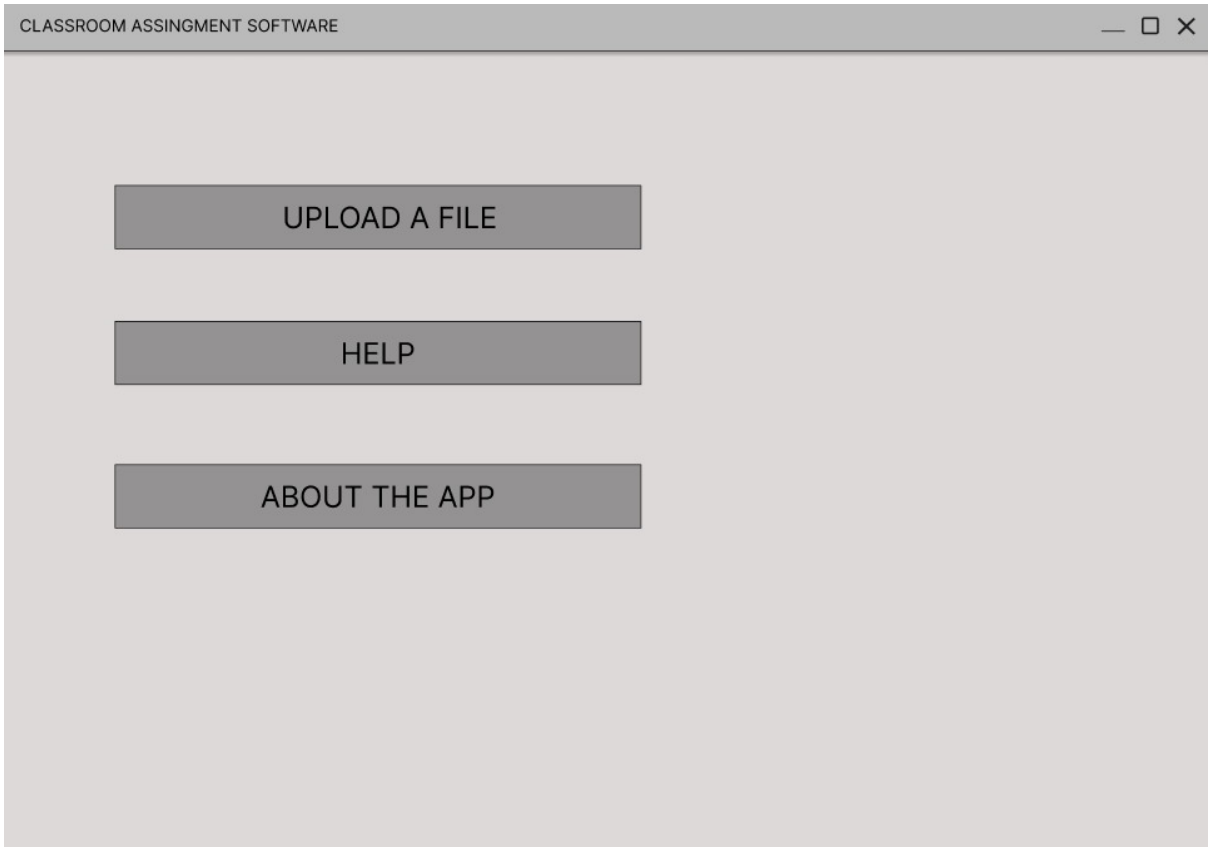

Figure 3: Main Page of the GUI

3.1 Edit Matching

The button "Edit Matching" will allow the user to make changes to the assignments.

3.2 Download as a File

To download the file to their computer, users should click "Download as a File."

3.3 View Schedule

The user will be able to see the schedule prepared according to the given file document.

The button "View Schedules" will navigate users to the following page:

3.4 Additional Features

The user will be able to handle necessary assignments, such as:

- Adding or removing a student from a course.
- Opening new classrooms if needed.
- Viewing available classrooms and their free time slots.

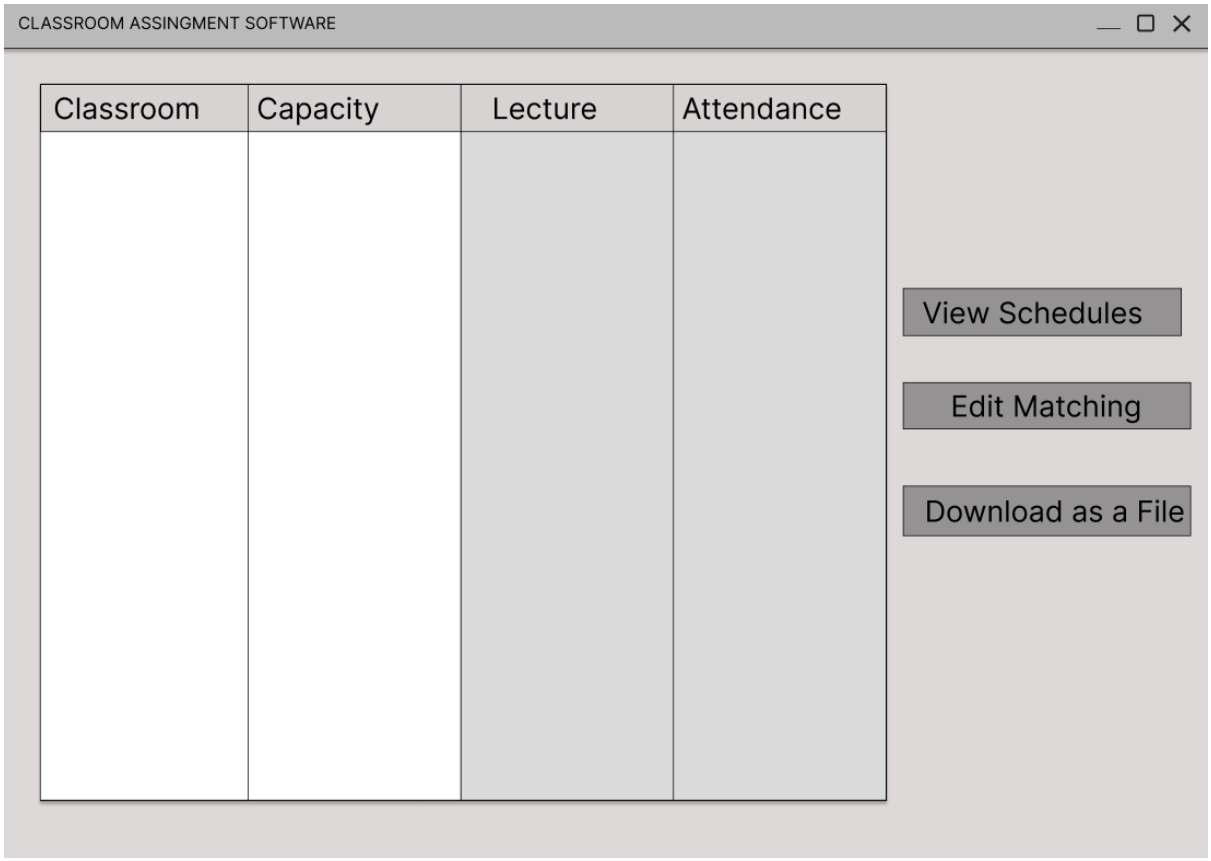

Figure 4: Table showing classroom names, capacities, lecture names, and attendance after uploading a file.

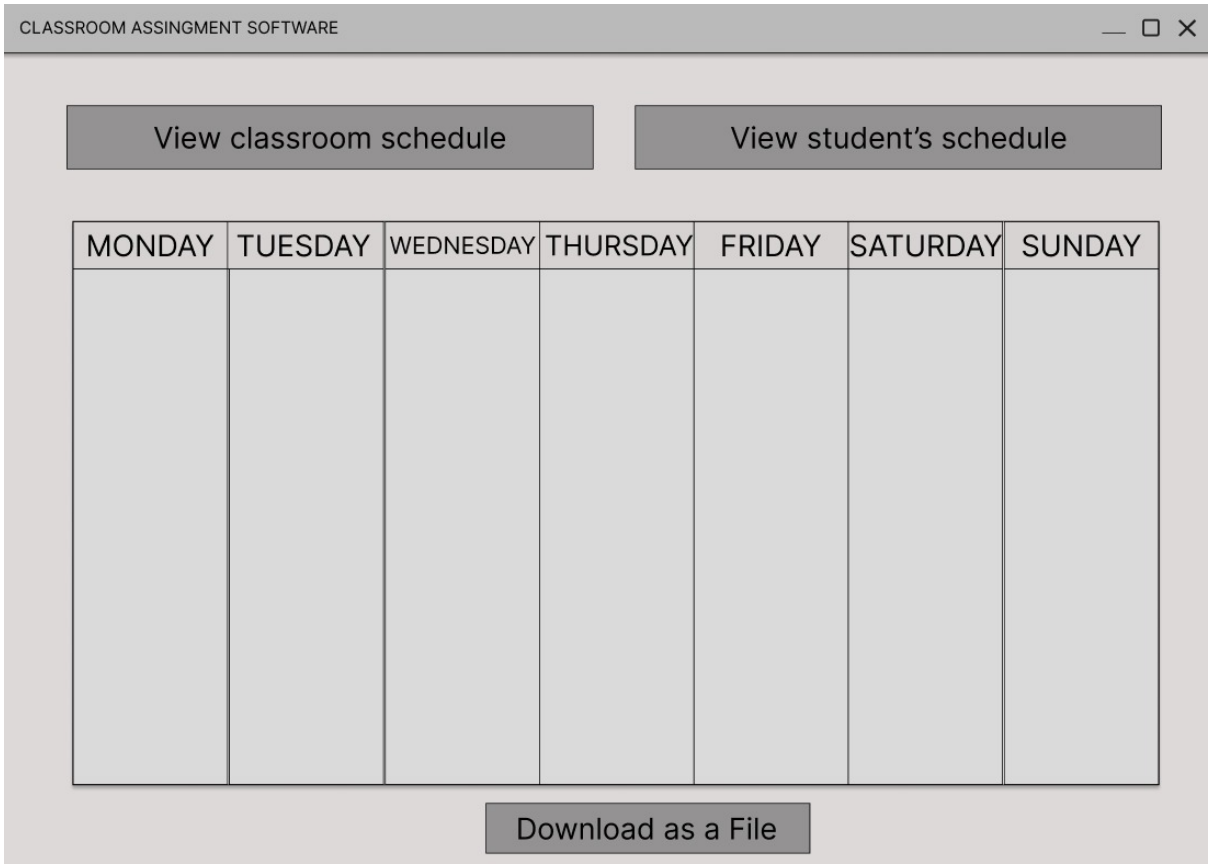

Figure 5: Schedule Portal of the GUI

3.5 View Classroom Schedule

The user will be able to see each class's information and the list of courses for a specific classroom.

3.6 View Student Schedule

The user will be able to check the schedule of each student.

Functional Requirement 14: The system will have a graphical user interface (GUI).