Graduation Project

Class Scheduling System

Prepared by

Name: Ahmed Abdulbasset Mohammed	Department: CS
Name: Ahmed Ali Abdelhameed	Department: CS
Name: Ahmed Alsaeed Abdelsadek	Department: CS
Name: Ahmed Fawzy Hussien	Department: CS
Name: Ali Ismael Hassan	Department: IS
Name: Omar Taqi Radi	Department: IS

Supervised by

Dr/Mamdouh Fathallah

Dr/Ibrahim Mahmoud

Eng/Mohammed Ahmed

Eng/Aya Ibrahim Elshafei

June 2023

1. ACKNOWLEDGMENT

We'd like to thank everyone that made it possible for us to get this far, first Allah and second everyone that taught us over the years and made us the people we are. The project wouldn't have been possible without the cooperation of everyone within the team, their dedication and the amount of hours spent researching and gathering information for this project.

We'd like to also thank our project Supervisors Dr. Mamdouh Fathallah and Dr. Ibrahim Mahmoud for always being available for us whenever we need him and always providing input and suggestions to make the project overall better, without his guidance we'd have had to backtrack a few times. We'd also extend our special thanks to Dean Moussa for his supervision alongside Dr. Mamdouh Fathallah and Dr. Ibrahim Mahmoud.

And besides our main supervisors, we'd like to thank our assigned Assistant Professors; Eng. Mohammed Ahmed, and Eng. Aya Ibrahim Elshafei for the great help they offered and the huge assistance in guiding us toward a project that we are proud of.

2. TABLE OF CONTENTS

3. TABLE OF FIGURES

4. ABSTRACT

Schedule making is an NP-hard problem that needs tremendous amount of effort and a lot of time to manually make twice a year, probably more if there was a mistake first two times. So in order to make it feasible for universities to make schedules twice yearly, and to reduce the effort needed from schedule makers, and to also reduce the amount of wasted time in students' day while at the same time ensuring that they do not have conflicts that obstacle their learning path, Schedule making has to be at-least partially automated.

All the options that are accessible to us and are related to the field in the market have huge gaps that cannot be ignored. For example, a system might make a schedule that has gaps between the subject full time. Or a system might make a schedule that has the subject separated into two different days. Both of which are not usable in our situation.

The obstacle to tackle in our project is: NP-hard problems cannot be solved using normal algorithms without consuming enormous amount of processing power and time. So a heuristic algorithm has to be used.

After analyzing similar systems, it appears that a genetic algorithm can generate a feasible time-table using somewhere between 0.2 nanoseconds and 2 seconds. Which is a huge improvement compared to the current manual process that takes at least 2 days.

Chapter 1: INTRODUCTION

5. CHAPTER 1: INTRODUCTION

5.1 OVERVIEW

A schedule is a way to assign something so all involved parties are in mutual agreement and can coordinate and communicate effectively, schedules have been used in universities since education started as it was hard to reach mutual agreement between a lot of parties, as someone will always have a problem with the time, so schedule makers resort to picking a time which would be suitable for the most people but not all, that'd align with the teacher's time and availably as well as be at a reasonable time at which someone would be awake and fully coherent at. Specifically a University schedule would include both Lectures and Labs, Lectures are given by the professor and involve a lot of students that attend in a large room. Labs is much smaller and given by Assistant Professors and apply the knowledge given in the lecture practically via various programs and programming languages. Labs are also used for general revision as Assistant Professors have more time to spare and help individual students than Professors as they'd have over a hundred of students within a single Class, and Professors can be teaching more one class.

5.2 Problem

A bad schedule would mean students are stuck doing nothing for extended periods of time or not able to attend their lecture at all. A bad schedule will directly correlate to a student's performance within the registered class or the university as a whole, it'll also directly result in miscommunication between students, assistant professors and professors that teach a specific class. Lecturers use schedules to also track the progression of the students as some classes have a large amount of people which will result in it being divided into two or more groups which will require more coordination, communication and focus. A constantly changing schedule or an unstable schedule that conflicts with other classes for students within the same level will result in one group receiving more training and teaching than the other. Sometimes students also abandon scheduled lectures when it's divided into groups for other ones within the same week due to lack of compatibility with their other classes or due to a bad schedule set for a specific group, which results in some groups being overcrowded than others and will cause a negative feedback loop as it'll result in an unstable schedule and keeps changing more to adapt to the students' needs.

Some lectures have Labs to follow-up with what was taken in the lecture and provide practical experiments to make the student better understand the class. Labs are less populated than lectures which means there will be even more labs than lectures, a constantly changing or unstable schedule with Lectures will also result in the changing of Labs schedules as they're directly connected and teach mostly the same material, Labs are completely reliant on the lecture and the lecturer. An extra problem is how tedious it is to manually configure each schedule and there's always a margin for human error that might cause oversights which if severe enough would result in the schedule being incompatible with the students, a machine will not make a mistake.

5.3 OBJECTIVE

Our Objective is to create a system that's easily accessible and easily manageable by college staff and schedule makers, solve the aforementioned problems mentioned above to make a program that uses complex algorithms with a few inputs to output a schedule that is suitable and doesn't have any conflicts between the multiple classes, our program will also format the schedule by itself and would give the user the option to quickly print it so the user doesn't need to spend extra time utilizing third party software to format it.

The Smart Scheduling System will output a schedule for the entire week, for both Lectures and Labs if applicable and will assign periods based on credit hours and whether or not it's assigned Labs to make sure each class is getting it's needed amount of time. The Smart Scheduling System would also give the user the option to edit old schedules in a timely manner in case a professor becomes unavailable in the day he's scheduled to give a Lecture in.

Lectures and Students will be able to easily understand and not differentiate between a man-made schedule and the schedule made by the smart scheduling system so it'll feel like a seamless process with no new changes.

The Smart Scheduling System would be hosted online on a website so that Faculty staff can utilize it anywhere from any computer device without having to be restricted to a single laptop or being forced to download and install it every time they need to use it on a different machine, it'd also allow faculty staff to download an easily printable PDF that they can distribute to the faculty students and teaching staff.

The Smart Scheduling System will also have a unique GUI for both the Faculty Staff that are responsible for making the schedules and normal users that are checking the schedules. The GUI will be easy to navigate and will take the inputs from the user via text boxes and drop-down menus to eliminate any confusion and be easy and quick to use whenever it's needed.

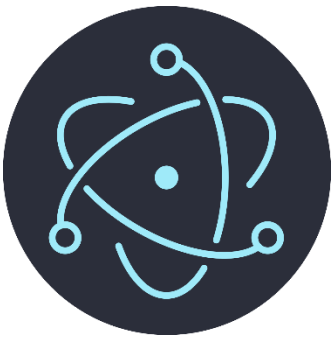
5.4 Tools

Front-end:

- Html
- CSS
- Javascript
- Angular

Back-end:

- Nodejs
- Express.js
- SQLite



5.4.1 Front-end tools

- **HTML**

  HTML, or HyperText Markup Language, has a fascinating history of development and evolution since its creation in the early 1990s. It was initially devised by Sir Tim Berners-Lee and his team at CERN as a means to structure and share scientific information on the emerging World Wide Web. The first standardized version, HTML 2.0, was released in 1995, introducing essential elements for web page creation. HTML 3.2 followed in 1997, bringing advancements like tables and forms. The advent of XHTML, an XML-based version of HTML, aimed for stricter adherence to rules and guidelines. HTML4, released in 1999, laid the groundwork for the separation of content and presentation using CSS. HTML5, introduced in 2014, marked a significant milestone, incorporating numerous new elements, attributes, and APIs for modern web development. HTML5 enabled the creation of multimedia-rich content, canvas-based graphics, and responsive, mobile-friendly designs. Today, HTML remains the backbone of the web, working in conjunction with CSS for styling and JavaScript for interactivity. It is widely used in website development, web applications, CMS platforms, e-commerce, and various online content formats. HTML's continuous evolution ensures its relevance and adaptability in shaping dynamic and engaging web experiences. we are using the latest version of it HTML 5.

- **Benefits of using HTML:**

  1. **Structured Presentation:** HTML provides the necessary structure to represent timetable data in a well-organized and readable format.

  2. **Responsive Design:** HTML allows us to design a responsive timetable website that can adapt to different screen sizes and devices.

  3. **Accessibility:** HTML's semantic elements can improve the accessibility of the timetable website. By using proper headings, labels, and markup.

  4. **Easy Maintenance:** HTML makes it straightforward to update and maintain the timetable website.

  5. **Integration with Other Technologies:** HTML seamlessly integrates with CSS and JavaScript, enabling us to enhance the visual presentation and interactivity of our timetable website.

  6. **Cross-Browser Compatibility:** HTML's wide support across different web browsers ensures that our timetable website will be accessible to users regardless of their browser preference.

- ## CSS

CSS, or Cascading Style Sheets, has a rich history of development and evolution since its introduction in the late 1990s. Initially known as CSS1, it provided a way to separate the presentation and styling of web documents from their underlying HTML structure. CSS2 followed with significant improvements, including positioning, floating elements, table handling, and typographic control. CSS3 introduced a modular approach, enabling the gradual introduction of new features like rounded corners, gradients, shadows, animations, and media queries. CSS frameworks and preprocessors further simplified and enhanced CSS development, while CSS Grid and Flexbox revolutionized web layout techniques. Today, CSS remains a crucial part of web development, allowing developers to create visually appealing designs, responsive layouts, and consistent branding across various platforms. Ongoing CSS standards development, such as CSS4 and CSS-in-JS approaches, continue to push the boundaries of CSS capabilities and further empower web designers and developers. CSS3 is most suitable for our project.

- ## Why use CSS?

1. **Separation of Concerns**: CSS allows for a clear separation of style and presentation from the underlying HTML structure. This separation makes the codebase more maintainable, as changes to the timetable's visual appearance can be made solely in the CSS file without modifying the HTML markup.

2. **Consistent Styling:** CSS enables consistent styling across the entire timetable website. By defining styles in a centralized CSS file, we can ensure that all timetable elements, such as headings, time slots, and event blocks, have a unified and professional look. This consistency enhances the user experience and creates a visually cohesive website.

3. **Customization Options:** CSS provides extensive customization options for the timetable's appearance. We can easily change colors, fonts, spacing, borders, and other visual aspects by modifying the CSS rules. This flexibility allows us to match the timetable's design to the overall branding and aesthetic of our website.

4. **Print-Friendly Formatting:** CSS can be used to create print-friendly timetables. By defining specific print styles, we can optimize the appearance of the timetable when users print it or save it as a PDF. This ensures that the printed version maintains a well-formatted structure for easy reference.

5. **Integration with Interactivity:** CSS seamlessly integrates with JavaScript, allowing us to add interactive features to the timetable.

- ## Javascript

  JavaScript, created by Brendan Eich in 1995, initially known as LiveScript, is a scripting language designed for web browsers to add interactivity and dynamic functionality to HTML pages. It gained traction as ECMAScript, standardized by Ecma International. Over time, JavaScript evolved to meet the demands of Web 2.0, with libraries like jQuery and frameworks such as Prototype and Dojo simplifying development. ECMAScript 5, released in 2009, introduced essential features and improved compatibility. However, ECMAScript 6 (ES6) in 2015 was a significant milestone, introducing powerful enhancements like arrow functions, classes, and modules. JavaScript expanded beyond the browser with Node.js in 2009, enabling server-side development. Today, JavaScript frameworks like React, Angular, and Vue.js dominate front-end development, while Node.js and Express.js facilitate full-stack JavaScript development. JavaScript's versatility extends to mobile app development, desktop applications, and IoT devices, making it one of the most widely used and versatile programming languages today.

- ## The advantages of using Javascript:

1. **Dynamic Interactivity:** JavaScript allows for dynamic and interactive features within the timetable. We can use JavaScript to create functions that enable users to add, edit, or remove events, change views, or perform other actions without reloading the entire page. This enhances the user experience and makes the timetable more engaging and efficient to use.

2. **Real-Time Updates:** With JavaScript, you can implement real-time updates in the timetable.

3. **Data Manipulation:** JavaScript provides powerful tools for manipulating and processing data in the timetable. We can use JavaScript to sort events based on various criteria, apply filters or search functionalities to help users find specific events, and perform calculations or aggregations on the timetable data. This flexibility allows for efficient data management and improves the usability of the timetable.

4. **Integration with APIs and External Data Sources:** JavaScript can be used to integrate the timetable with external APIs or data sources. This enables you to fetch data from databases, web services, or other sources and populate the timetable dynamically. You can also use JavaScript to make HTTP requests and interact with APIs to fetch or update timetable data in real-time.

5. **User Input Validation:** JavaScript can validate user input in the timetable to ensure data integrity and accuracy.

6. **Friendly and Responsive Design:** JavaScript allows for the creation of mobile-friendly and responsive timetable websites. By using JavaScript frameworks or libraries like React or Bootstrap, you can build responsive layouts that adapt to different screen sizes and devices. This ensures that the timetable is accessible and user-friendly across desktops, tablets, and mobile devices.

- ## Angular

  Angular is a powerful, feature-rich open-source framework for building dynamic web applications. Developed and maintained by Google, it offers a robust solution for creating single-page applications (SPAs) and is noted for its ability to create efficient, scalable code that's easily maintainable. Angular is built around a component-based architecture which promotes modularity, reusability, and testability of code. It's designed with a strong emphasis on performance, employing tools such as lazy loading and ahead-of-time (AOT) compiling. Angular leverages the strengths of TypeScript, a statically-typed superset of JavaScript, to offer developers improved security, code structuring, and tooling. Its broad range of features, including two-way data binding, dependency injection, directives, and a powerful router, make Angular a comprehensive choice for professional web development.

- The benefit of using Angular:

1. **Component-based Architecture**: Angular uses a component-based architecture, which helps in organizing code into modular, reusable and maintainable units. This also improves code readability and reusability.

2. **Single Page Application (SPA):** Angular is mainly used to develop single page applications. This allows for a smoother, faster user experience since only certain parts of the page are updated rather than the whole page.

3. **Routing:** Angular provides a powerful router enabling navigation from one view to another as users perform application tasks. It includes features like lazy-loading, dynamic routing, and nested routing.

4. **Performance:** With features like lazy loading, server-side rendering, and ahead-of-time (AOT) compiling, Angular is built with an emphasis on performance.

5. **TypeScript**: Angular applications are built using TypeScript language, a superscript for JavaScript, which ensures higher security as it supports types (primitives, interfaces, etc.). It helps catch and eliminate errors early when writing the code or performing maintenance tasks.

5.4.2 Back-End Tools

- **Node.js**

  Node.js, developed in 2009 by Ryan Dahl, is a JavaScript runtime built on Chrome's V8 engine. It introduced an event-driven, non-blocking I/O model, addressing the scalability limitations of traditional web servers. Node.js allows for efficient handling of concurrent requests, making it ideal for web servers, APIs, and real-time applications. It promotes full-stack JavaScript development, enabling developers to use the same language across the client and server sides. Node.js incorporates NPM, a package manager that provides access to a vast ecosystem of open-source libraries. It is commonly used for building microservices architectures, command-line tools, single-page applications, and real-time applications such as chat platforms and streaming services. With its high scalability, performance, and active community, Node.js continues to be a popular choice for server-side development, driving innovation in web application development.

- **why use Node.js:**

1. **JavaScript Full Stack:** With Node.js, we can utilize JavaScript on both the front-end and back-end of our timetable website. This enables a seamless transition as we can use the same language and share code between the client and server sides.

2. **Non-Blocking I/O:** Node.js follows a non-blocking, event-driven architecture, allowing it to handle I/O operations asynchronously. This means that while waiting for database queries or API calls to complete, Node.js can continue processing other requests, optimizing resource utilization, and improving overall performance.

3. **Ecosystem and Package Management:** Node.js comes with a robust ecosystem, including the Node Package Manager (NPM). NPM provides access to numerous open-source libraries and modules that can enhance the functionality of our timetable website, such as date/time manipulation libraries, database connectors, and scheduling modules. This extensive ecosystem saves development time and effort by leveraging existing solutions.

4. **Integration with Front-End Frameworks:** Node.js can seamlessly integrate with our front-end frameworks Angular. This allows us to build interactive and dynamic user interfaces for our time table website and also leveraging the back-end capabilities of Node.js for data management and server-side operations.

- ## Electron

Electron, formerly known as Atom Shell, is a cross-platform desktop application framework developed by GitHub in 2013. It combines the Chromium rendering engine and the Node.js runtime, enabling developers to create desktop applications using web technologies. Electron was open-sourced in 2014, allowing for widespread adoption and contributions from the developer community. With Electron, developers can leverage their existing web development skills, such as HTML, CSS, and JavaScript, to build desktop applications that can run on Windows, macOS, and Linux without major modifications. Electron is widely used in various domains, including productivity tools, text editors, chat applications, and developer IDEs. It provides access to native capabilities, allowing desktop applications to interact with the underlying operating system. Electron has become a popular choice for rapidly prototyping and building cross-platform desktop applications while benefiting from the familiarity and power of web technologies.

- ## the advantages of using Electron framework:

1. **Cross-Platform Compatibility:** this is the most important feature of Electron, we can create a timetable website that can be packaged and run as a desktop application on multiple platforms, including Windows, macOS, and Linux. This allows users to access and use the timetable application directly from their desktop, providing a convenient and consistent experience across different operating systems.

2. **Web Technology**: Electron allows developers to use web technologies (HTML, CSS, and JavaScript) to build desktop applications, which makes it an excellent option for web developers moving into desktop application development.

3. **Access to Device-level APIs:** Electron provides developers with access to native OS APIs, allowing your application to interact more deeply with the operating system, offering functionality that typical web applications can't provide.

4. **Offline Support:** With Electron, we can build a timetable application that can work offline. Electron allows us to cache resources and data, enabling users to access and use the timetable even without an internet connection.

5. **Enhanced Performance:** Electron leverages the power of Chromium and Node.js, providing a high-performance runtime environment for our timetable application. This ensures smooth rendering of the user interface, efficient handling of data, and responsiveness even when dealing with large data sets or complex operations.

- **SQL LITE**

SQLite, an open-source embedded relational database management system (RDBMS), has a rich history and development. Created by Dr. Richard Hipp in the early 2000s, SQLite was designed as a lightweight, serverless, and self-contained database engine. Originally developed for CVSTrac, it gained popularity due to its simplicity, small footprint, and easy integration into applications. Over the years, SQLite has evolved with regular updates, ensuring reliability, performance, and security. Its features include a serverless architecture, self-contained databases in a single file, transactional support, and ACID compliance. SQLite finds extensive use in embedded systems, IoT devices, mobile applications, desktop applications, testing environments, and small to medium-scale web applications. Its versatility, efficiency, and zero-configuration setup make it a preferred choice for developers seeking a lightweight and reliable embedded database solution.

- **Why did we use SQL LITE?**

1. **Local Data Storage:** SQLite is a serverless and embedded database, allowing the timetable application to store data directly on the user's desktop. This eliminates the need for a separate database server and enables offline access to the timetable data.

2. **Fast and Efficient Performance:** SQLite is known for its lightweight design and efficient performance. It is optimized for local storage and retrieval operations, ensuring fast and responsive data access within the timetable application.

3. **Data Integrity and Reliability:** SQLite provides ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring data integrity and reliability. Transactions can be used to maintain the consistency of the timetable data, allowing safe updates and rollbacks in case of errors or failures.

4. **Easy Deployment and Maintenance:** SQLite databases are self-contained in a single file, simplifying the deployment and maintenance of the timetable application. It eliminates the need for complex database setup or installation, making it easier for users to install and update the application.

5. **Enhanced Security:** SQLite provides built-in support for data encryption, ensuring the security and privacy of the timetable data. This helps protect sensitive information and ensures that only authorized users can access and modify the data.

6. **Scalability:** SQLite can handle a significant amount of data and concurrent users, making it suitable for a timetable application that may grow in usage over time. It efficiently manages database operations, providing optimal performance even with increasing data volume.

7. **Simplified Development:** SQLite's SQL-based query language is widely understood and supported, making it easier for us to work with and manipulate the timetable data. It provides a familiar and standardized approach to database operations.

Chapter 2: Literature Review

6. CHAPTER 2: LITERATURE REVIEW

6.1 INTRODUCTION

A literature review is an essential component of academic research and writing. It involves a comprehensive examination and analysis of existing published materials such as books, scholarly articles, conference papers, dissertations, and other relevant sources on a particular topic. The purpose of a literature review is to provide a synthesis of existing knowledge, identify gaps in research, and establish the context for a new study or project.

6.2 Solving timetable scheduling problem using genetic algorithms.

Title: Solving timetable scheduling problem using genetic algorithms.

Authors: B. Sigl, M. Golub, V. Mornar

Publication Year: 2003

Description: Genetic algorithms belong to guided random search techniques, which try to find the global optimum. Genetic algorithms are working with the set of potential solutions, which is called population. Each solution item (individual) is measured by fitness function. The fitness value represents the quality measure of an individual, so the algorithm can select individuals with better genetic material for producing new individuals and further generations.

Result: The algorithm was tested on small and large instances of the problem. Algorithm performance was significantly enhanced with modification of basic genetic operators. Intelligent operators restrain the creation of new conflicts in the individual and improve the overall algorithm 's behavior.

6.3 Solving Timetabling Problem Using Genetic and Heuristic Algorithms

Title: Solving Timetabling Problem Using Genetic and Heuristic Algorithms

Authors: Thanh, Nguyen Duc

Publication Year: 2007

Description: The combination of genetic and heuristic algorithms in solving timetabling problems involves initializing an initial population using a genetic algorithm, applying standard genetic algorithm steps to evolve the population, incorporating heuristic algorithms to further refine the best solutions, integrating the best solutions from both algorithms, and iterating the process for refinement. By leveraging the global search capabilities of the genetic algorithm and the local search capabilities of heuristic algorithms, this hybrid approach aims to find optimal or near-optimal timetabling schedules, taking into account constraints and objectives. Experimentation and parameter tuning are crucial to determine the most effective hybridization strategy for a specific timetabling problem.

Result: This algorithm was implemented and tested with the synthetic and real data of Nong lam University of HCM City, Vietnam. The experimental results reveal the usability and potential of the proposed algorithm in solving timetabling problems.

6.4 Genetically Evolved Solution to Timetable Scheduling Problem

Title: Genetically Evolved Solution to Timetable Scheduling Problem
Authors: Sandesh Timilsina, R. Negi, Yashika Khurana, Jyotsna Seth

Publication Year: 2015

Description: Genetic algorithm is non-deterministic and is used to solve mainly NP-hard problem like timetable scheduling problem. There are situations where even after spending many hours trying to find a perfect schedule, the optimal solution is not found. Scheduling timetable being a hard task for manual computation, the use of automated genetic algorithm can profoundly increase the efficiency of the process.

Result: The results show that the basic standard genetic algorithm is unable to satisfy most real-world timetabling problems as the problem gets complicated. Modification to the standard genetic algorithm is needed to achieve better results. The modification can be done in any way that suits the situation.

6.5 Literature Review Models

Current models:

most of the currently existing models focus on how to solve time tabling problems using genetic algorithms and other heuristic algorithms in a theoretical way and the use of genetic algorithms must be constrained to achieve maximum accuracy and most implementation of this algorithm is complex and hard to be understood by a normal schedule maker as it requires prior knowledge of algorithms especially heuristic algorithms.

Our model:

we focus on simplify the task of the schedule maker by a simple yet efficient system that utilizes an easy and simple user interface that enhances the user experience and does not require prior algorithmic experience to use.

Chapter 3: System Analysis

7. CHAPTER 3: SYSTEM ANALYSIS

7.1 USER STORIES

1. As a System Administrator, I should be able to add the subjects and their details.

2. As a System Administrator, I should be able to add the number of students registered in each subject.

3. As a System Administrator, I should be able to input number of days.

4. As a System Administrator, I expect the system to divide the students into Groups.

5. As a System Administrator, I expect the system to divide the students into sections.

6. As a System Administrator, I expect the system to detect conflicts.

7. As a System Administrator, I expect the system to solve conflicts.

8. As a System Administrator, I should be able to view unsolved conflicts.

9. As a System Administrator, I should be able to modify the table.

10. As a System Administrator, I expect the system to generate tables.

11. As a System Administrator, I should be able to view tables.

12. As a System Administrator, I should be able to choose the optimal table.

13. As a System Administrator, I should be able to have a copy of the chosen table.

7.2 Activity Diagram

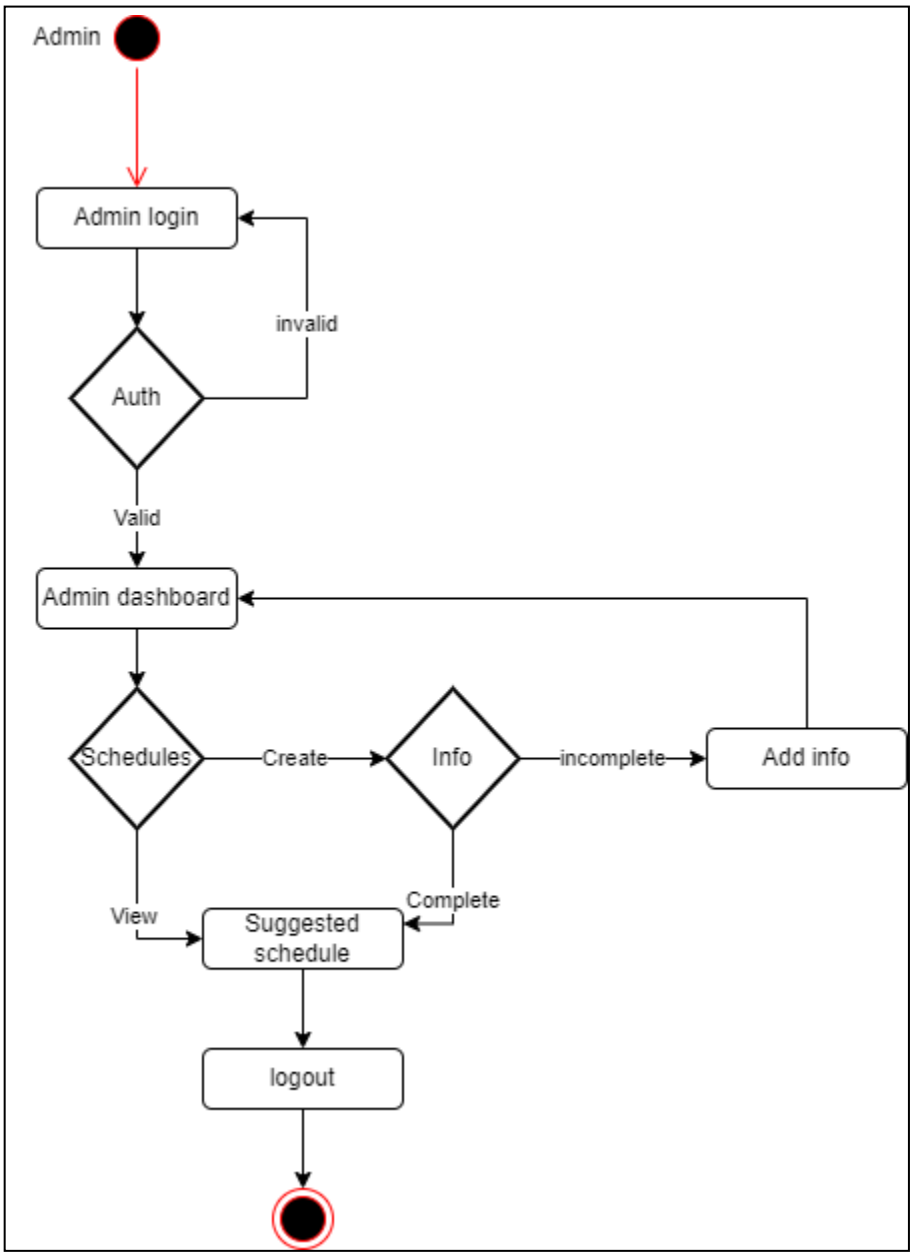


Figure 1 Activity Diagram

7.3 ERD Diagram

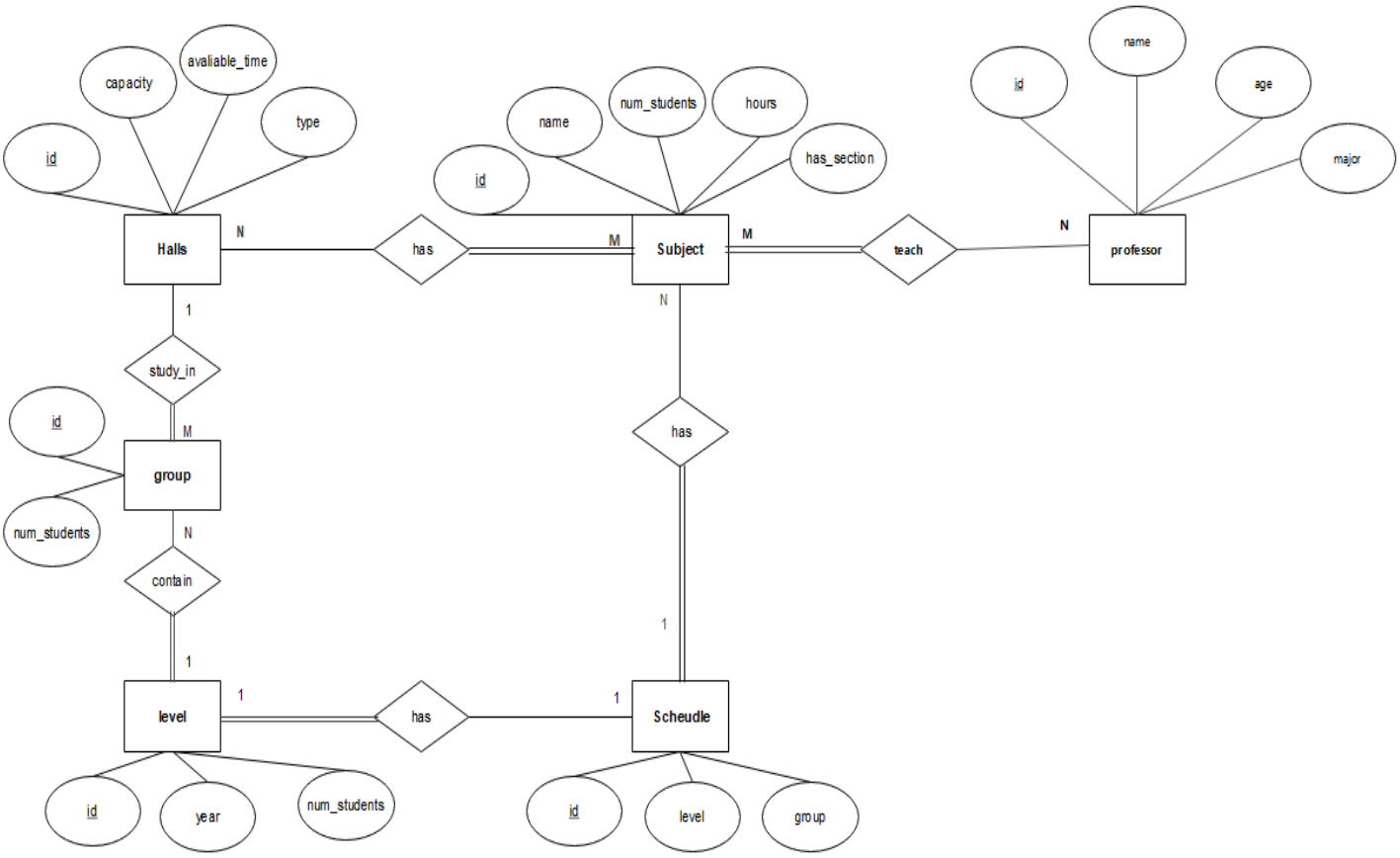


Figure 2 ERD Diagram

7.4 Relational model

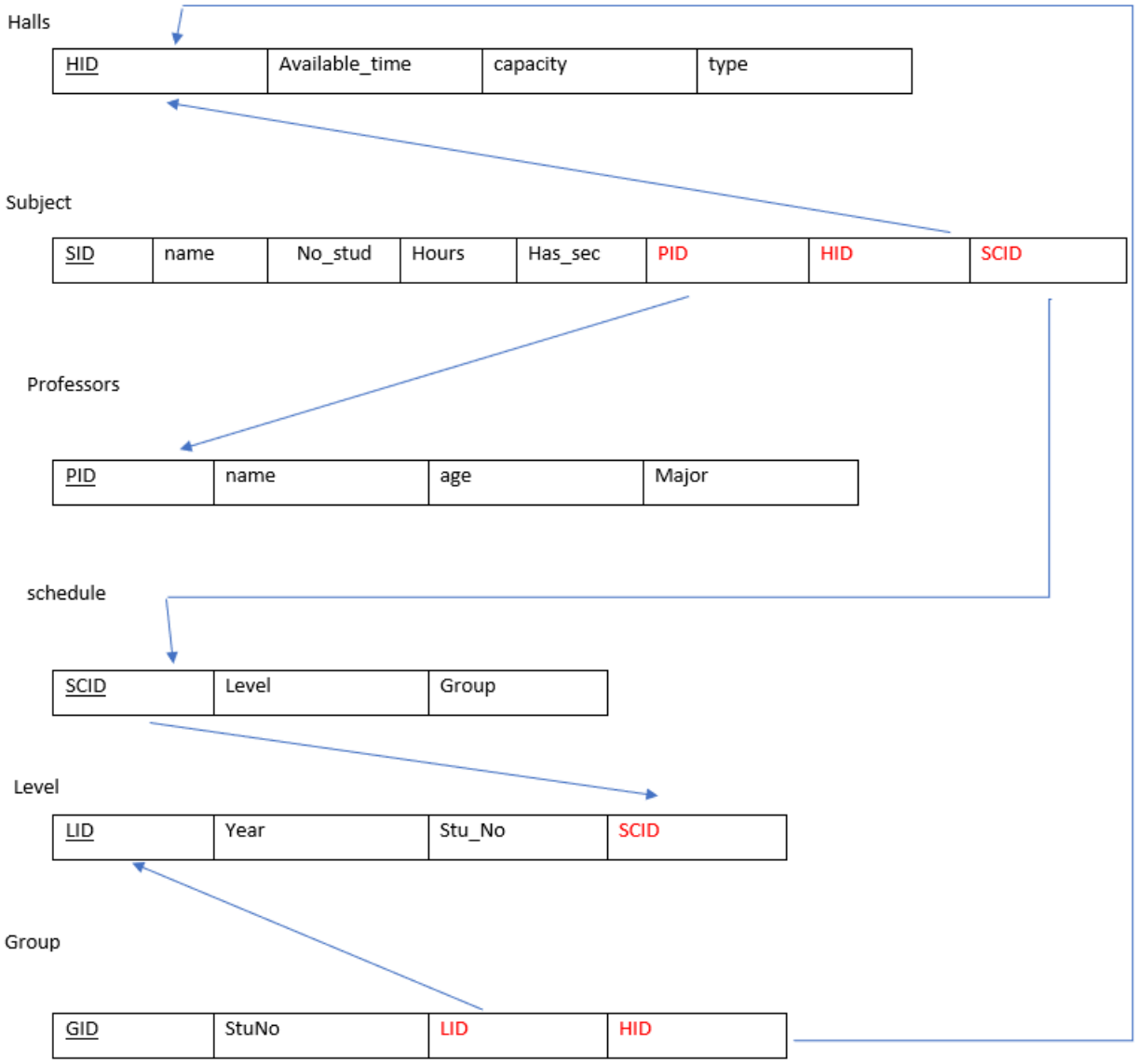


Figure 3 Relational Model

7.5 Database Schema

Figure 4 Database Schema

7.6 Context Diagram

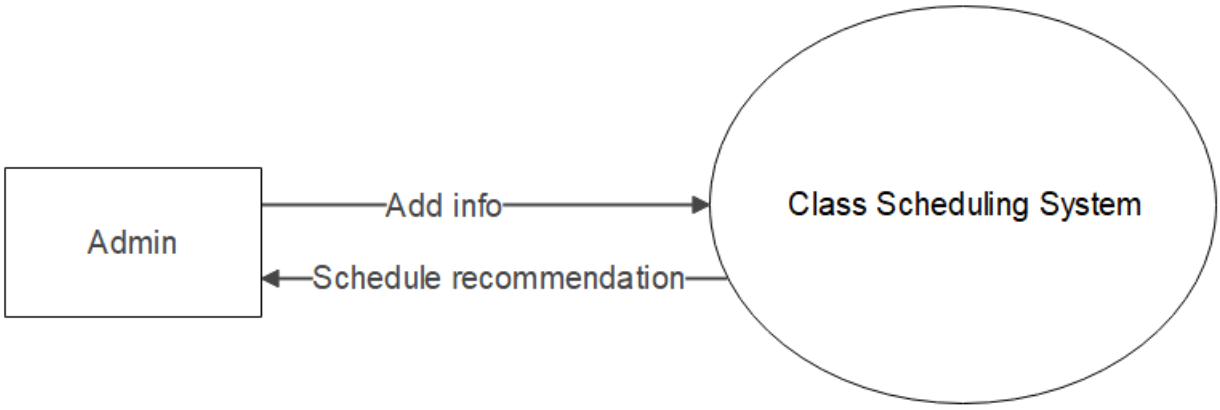


Figure 5 Context Diagram

7.7 Dataflow Diagram

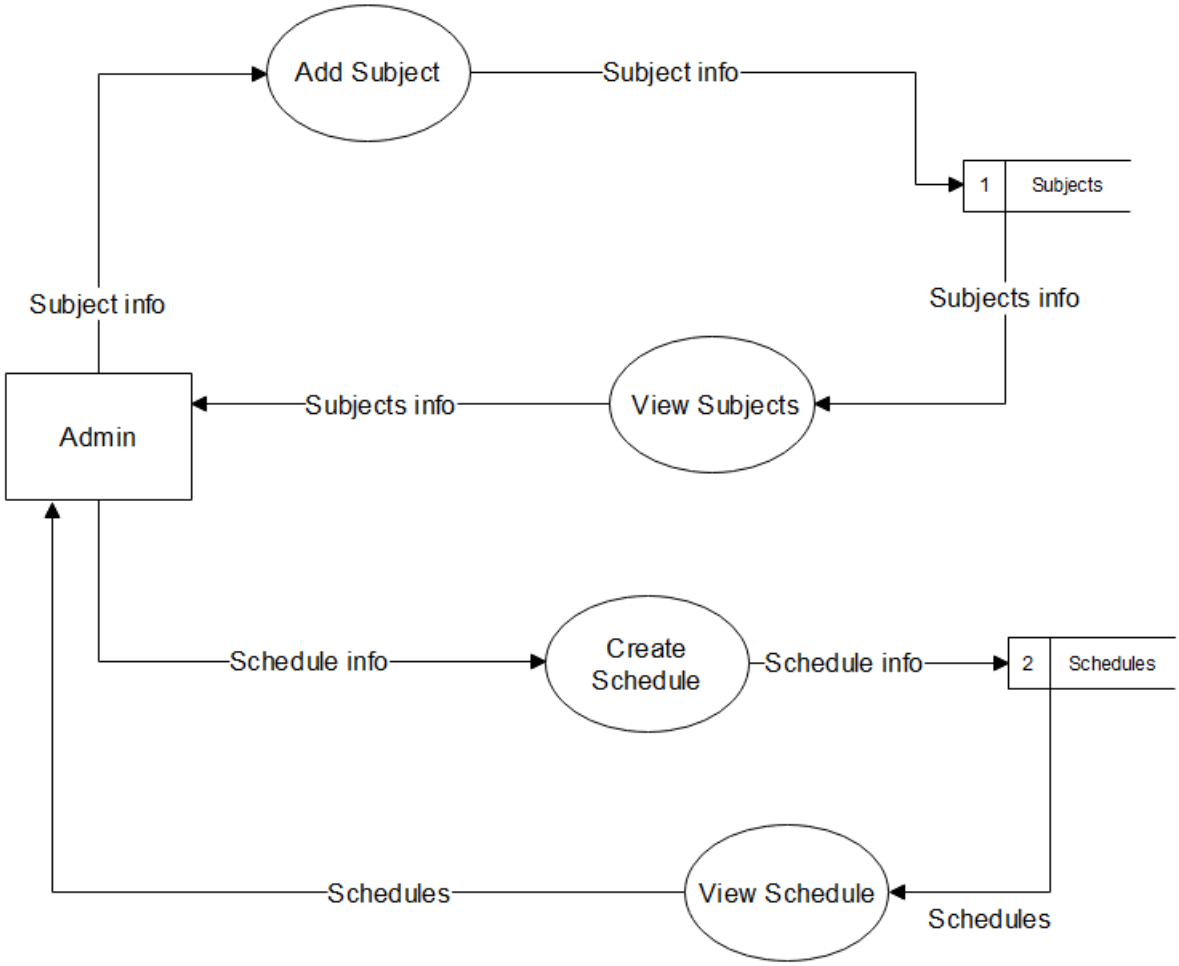


Figure 6 Dataflow Diagram

7.8 Use Case Diagram

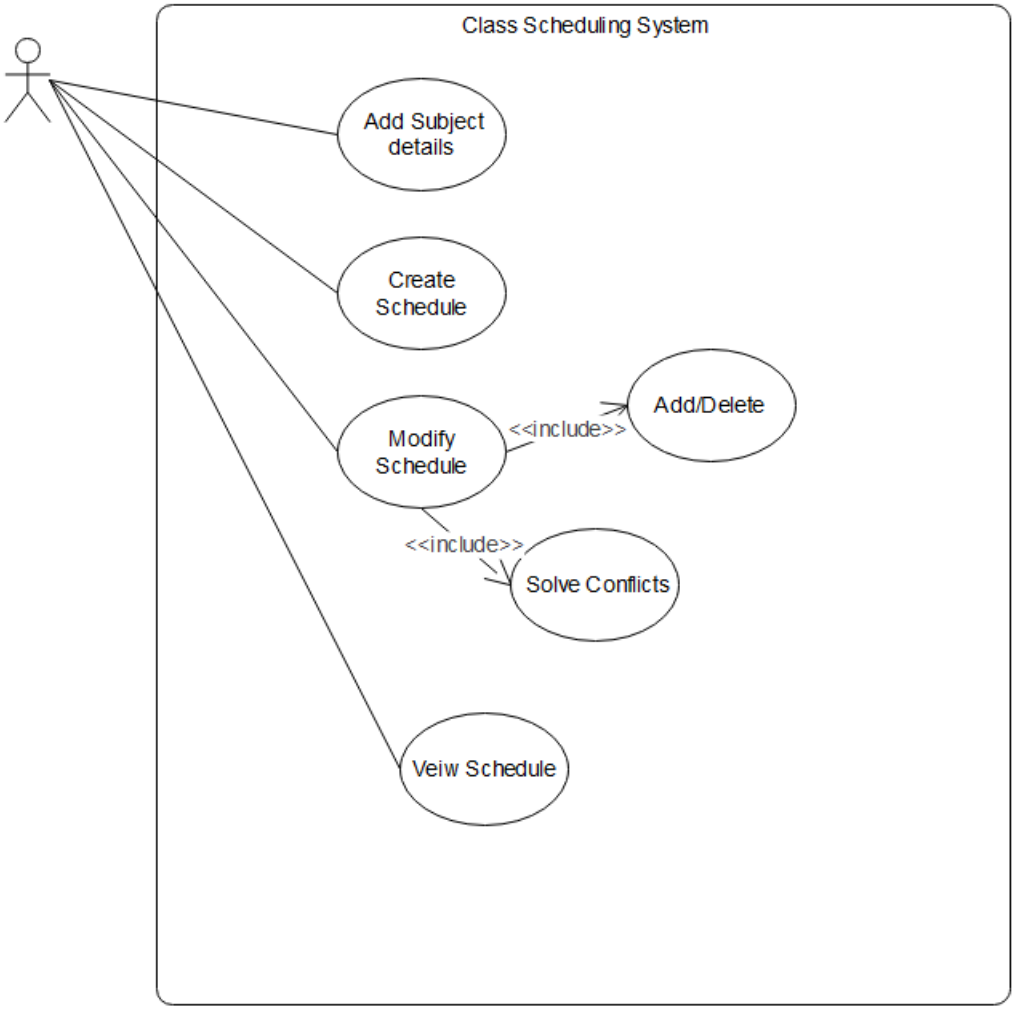


Figure 7 Use Case Diagram

Chapter 4: IMPLEMENTATION

8 Chapter 4: Implementation

8.1 UI implementation

Home screen

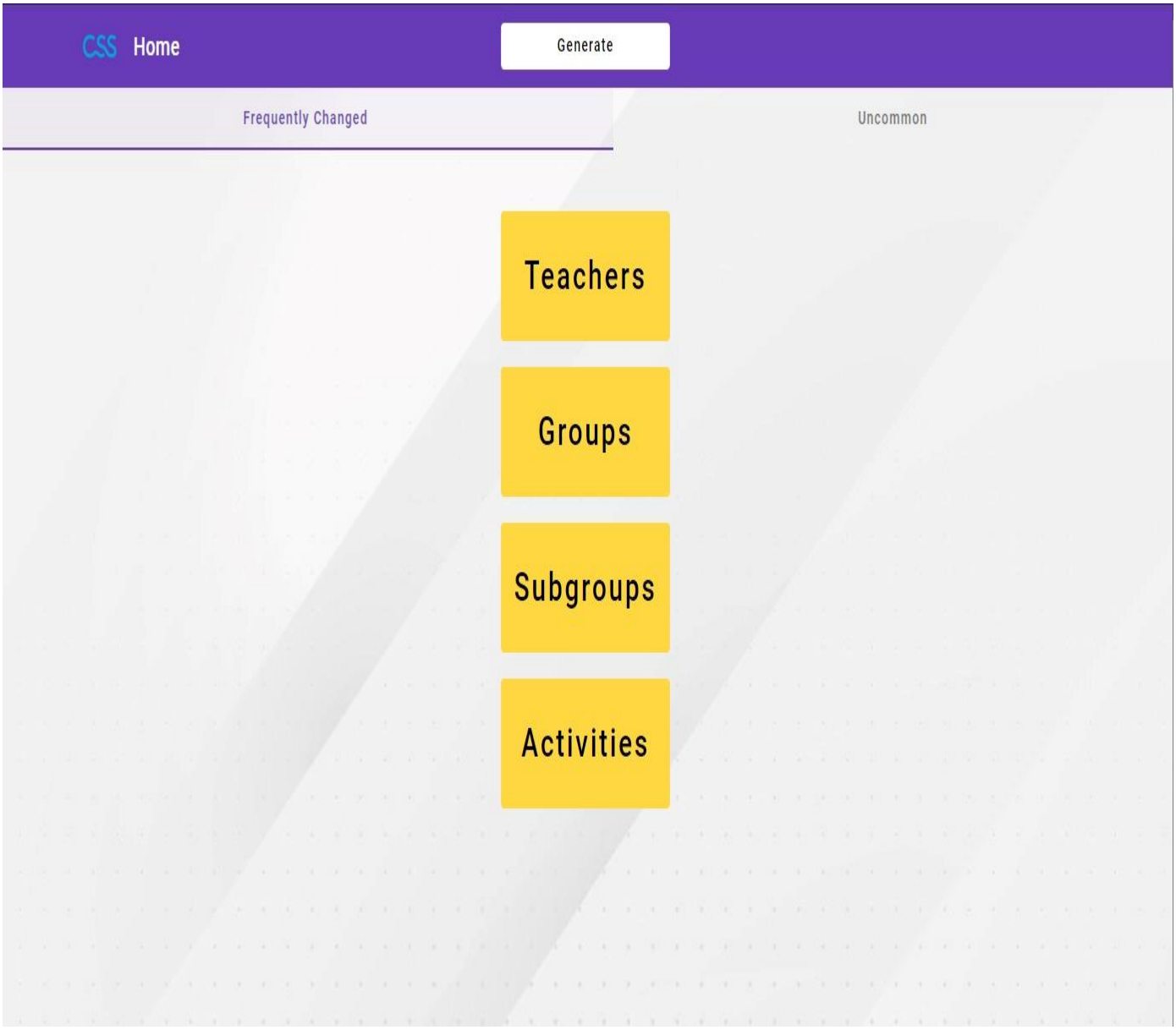


Figure 8 Home Page 1

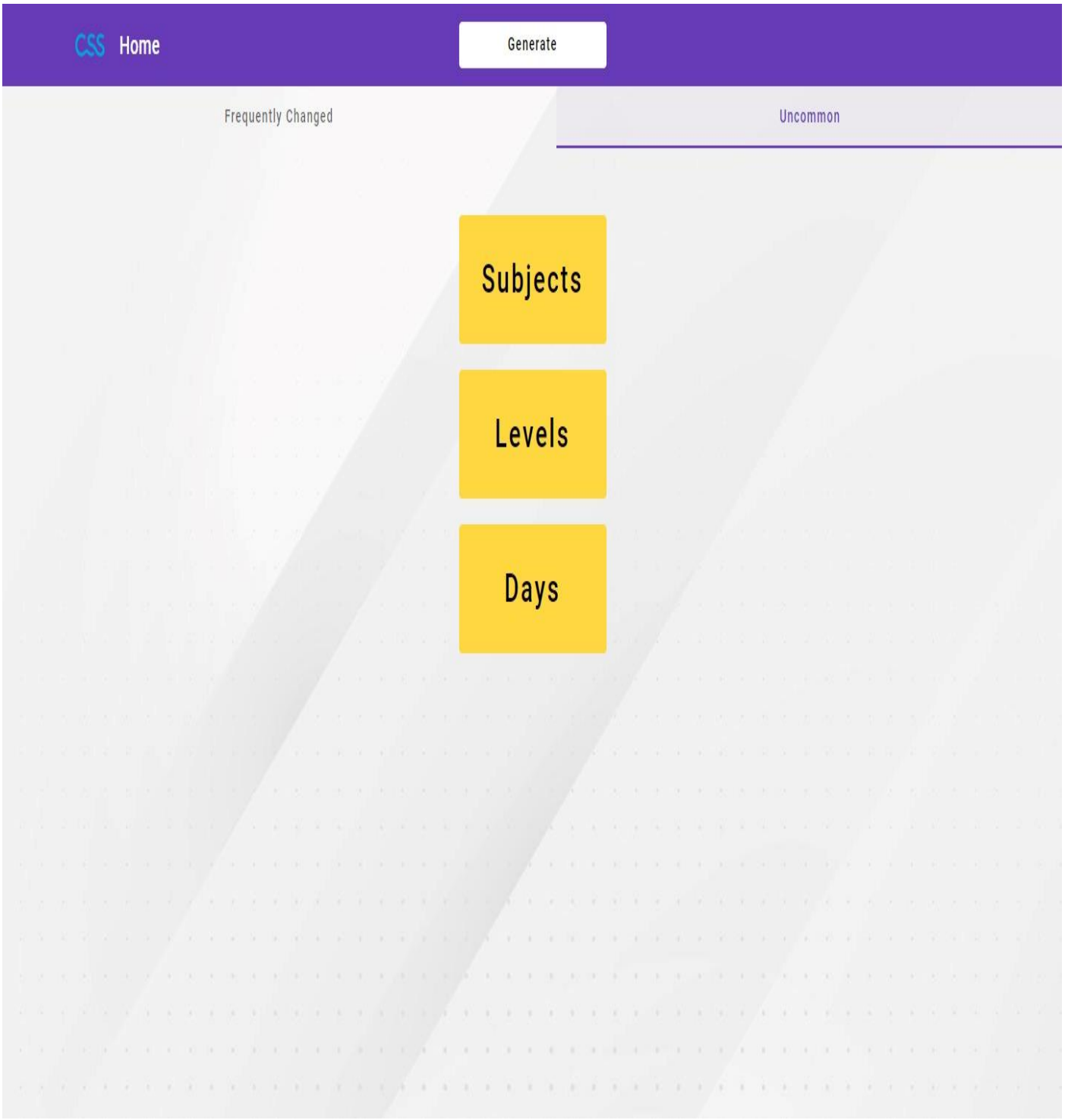


Figure 9 Home Page 2

Teacher page

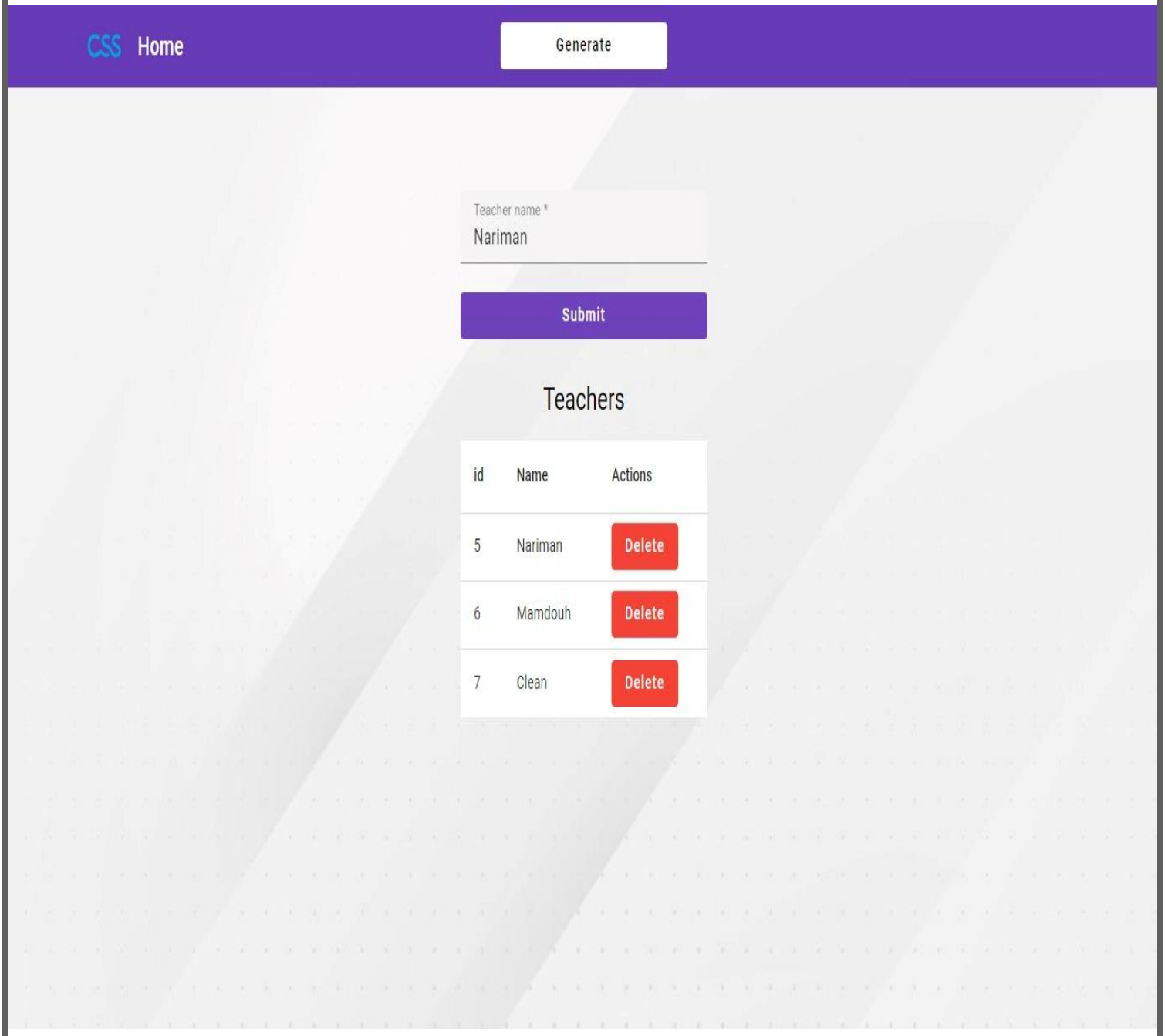


Figure 10 Teacher Page

Subject page

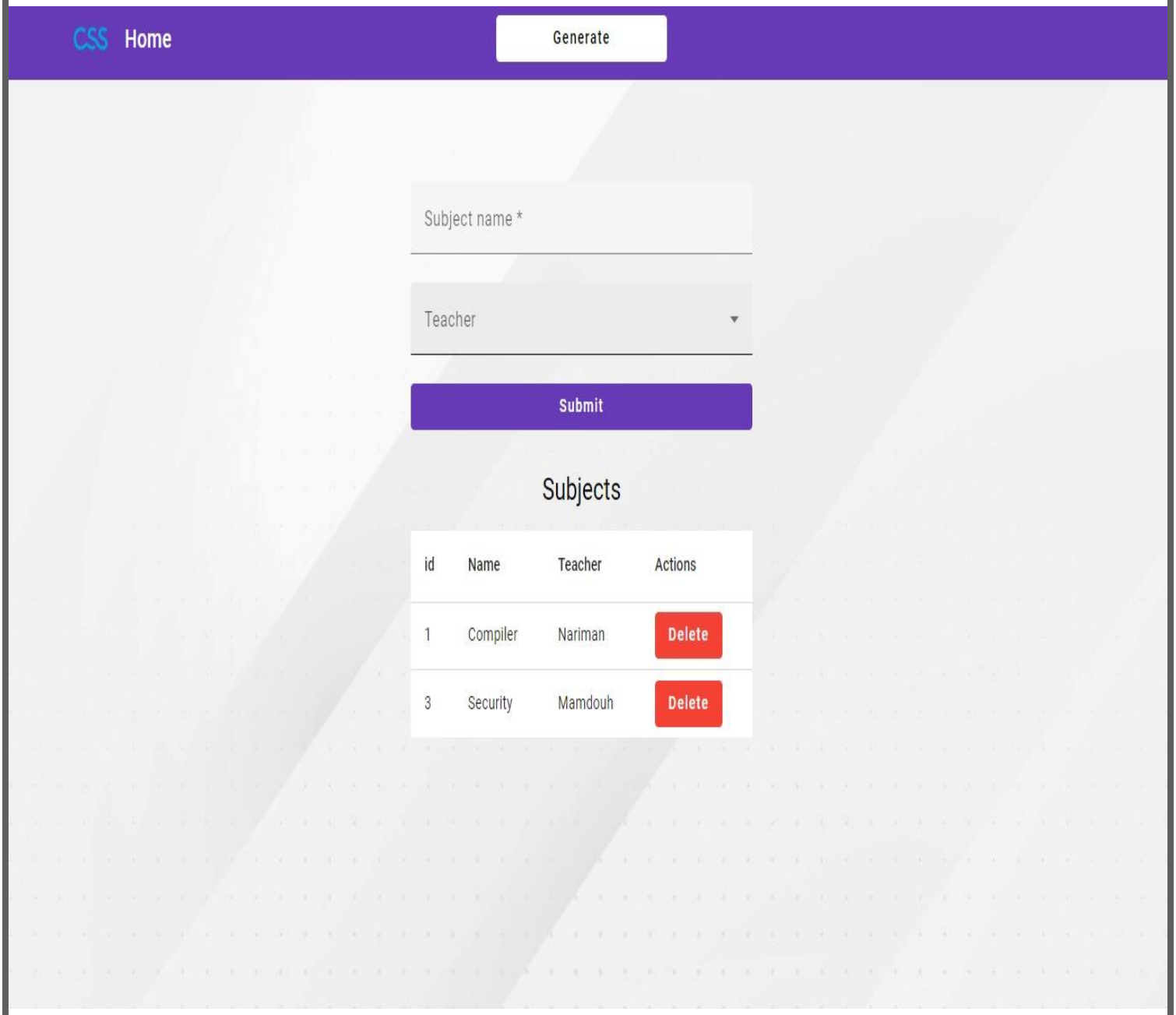


Figure 11 subject page

Days page

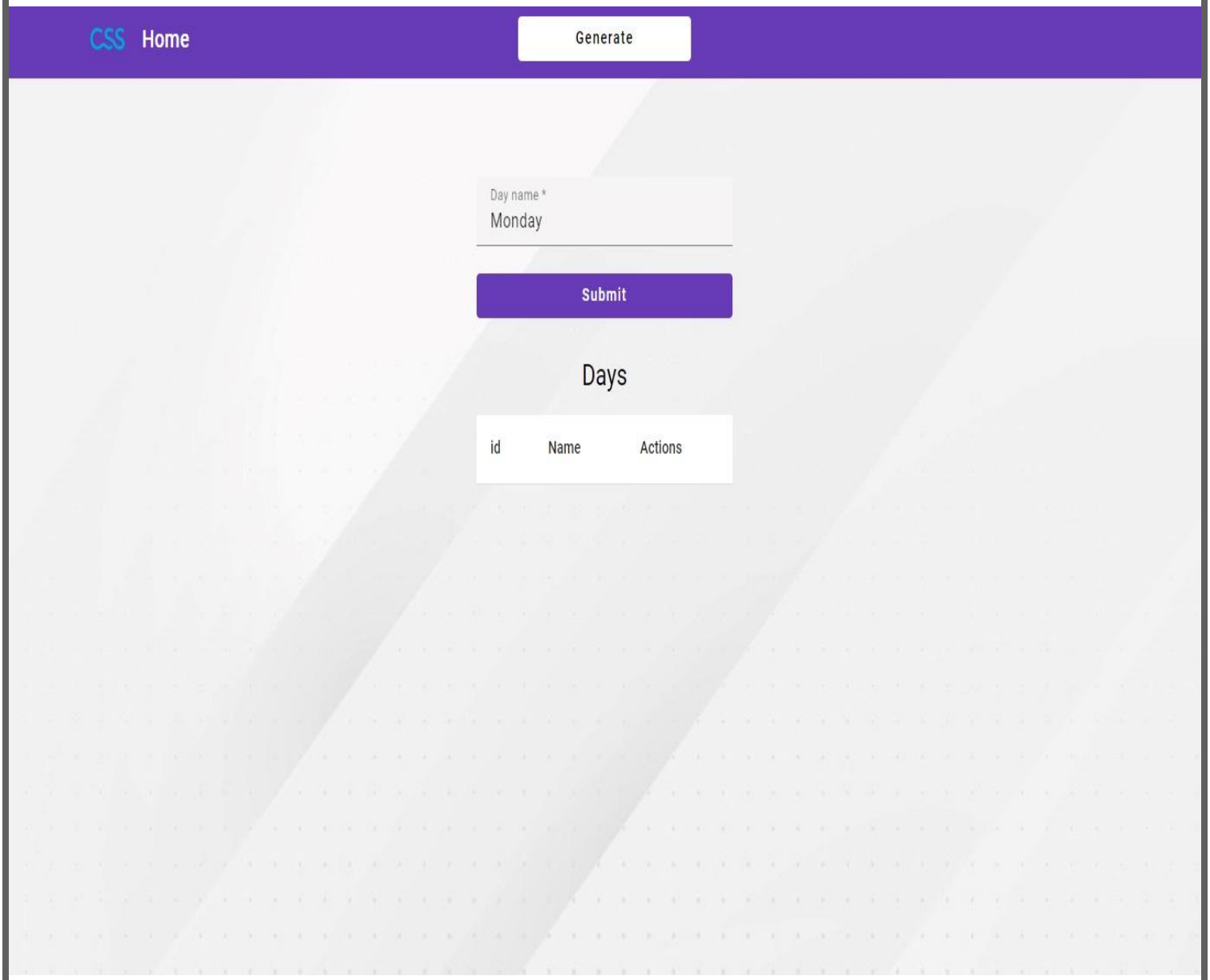


Figure 12 Days page

Levels insertion page

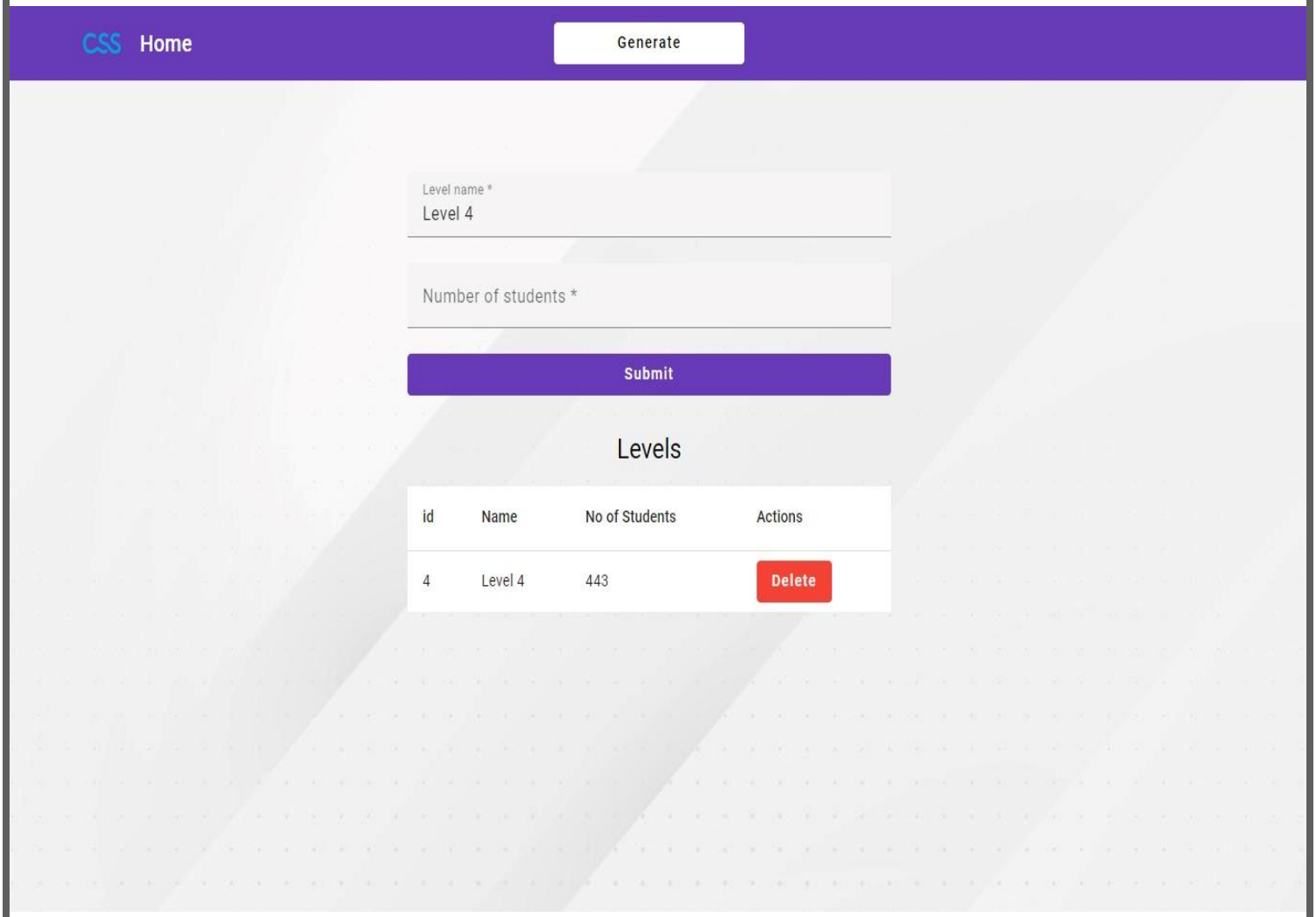


Figure 13 Levels page

Group insertion page

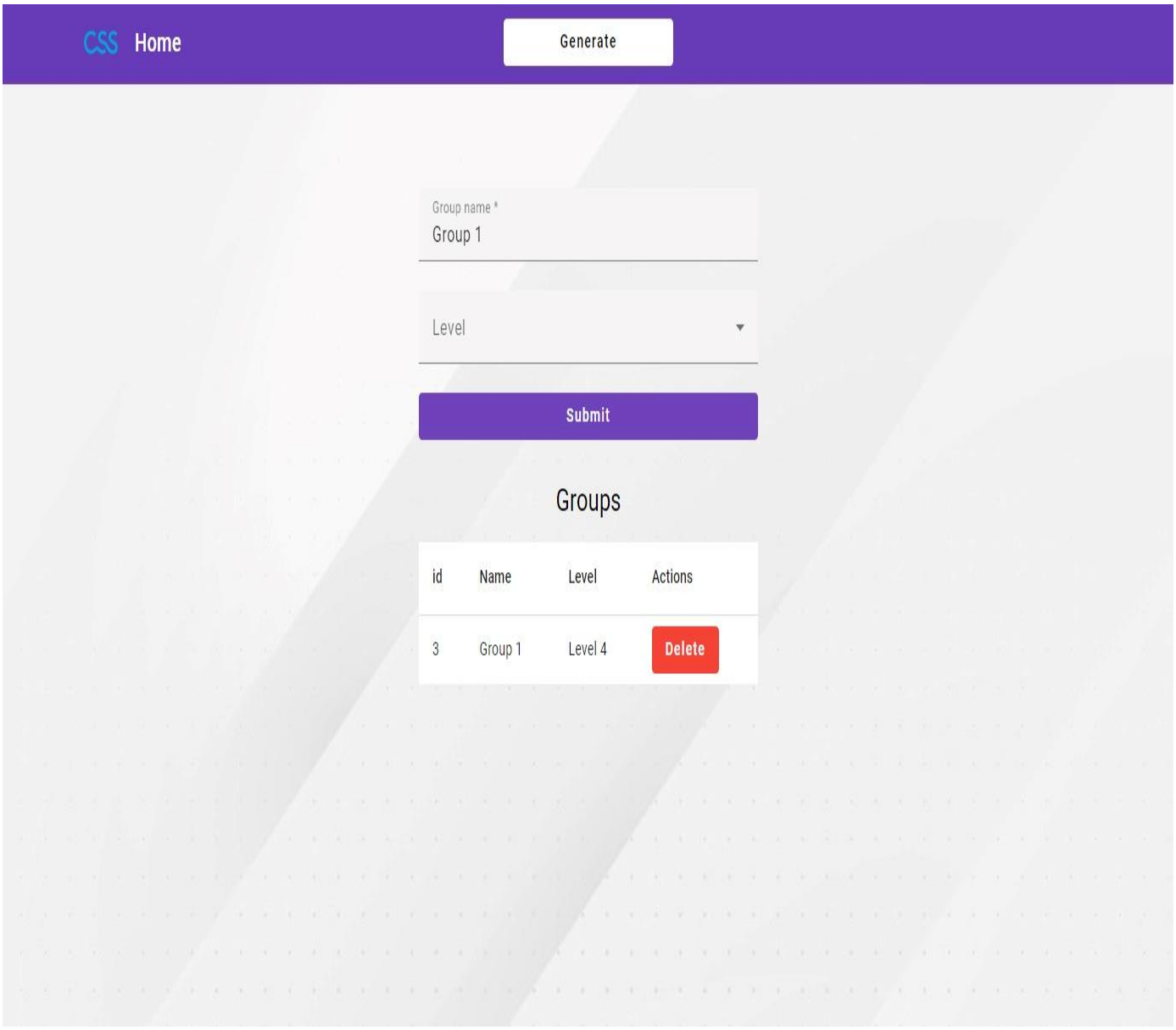


Figure 14 group page

Sub Group page

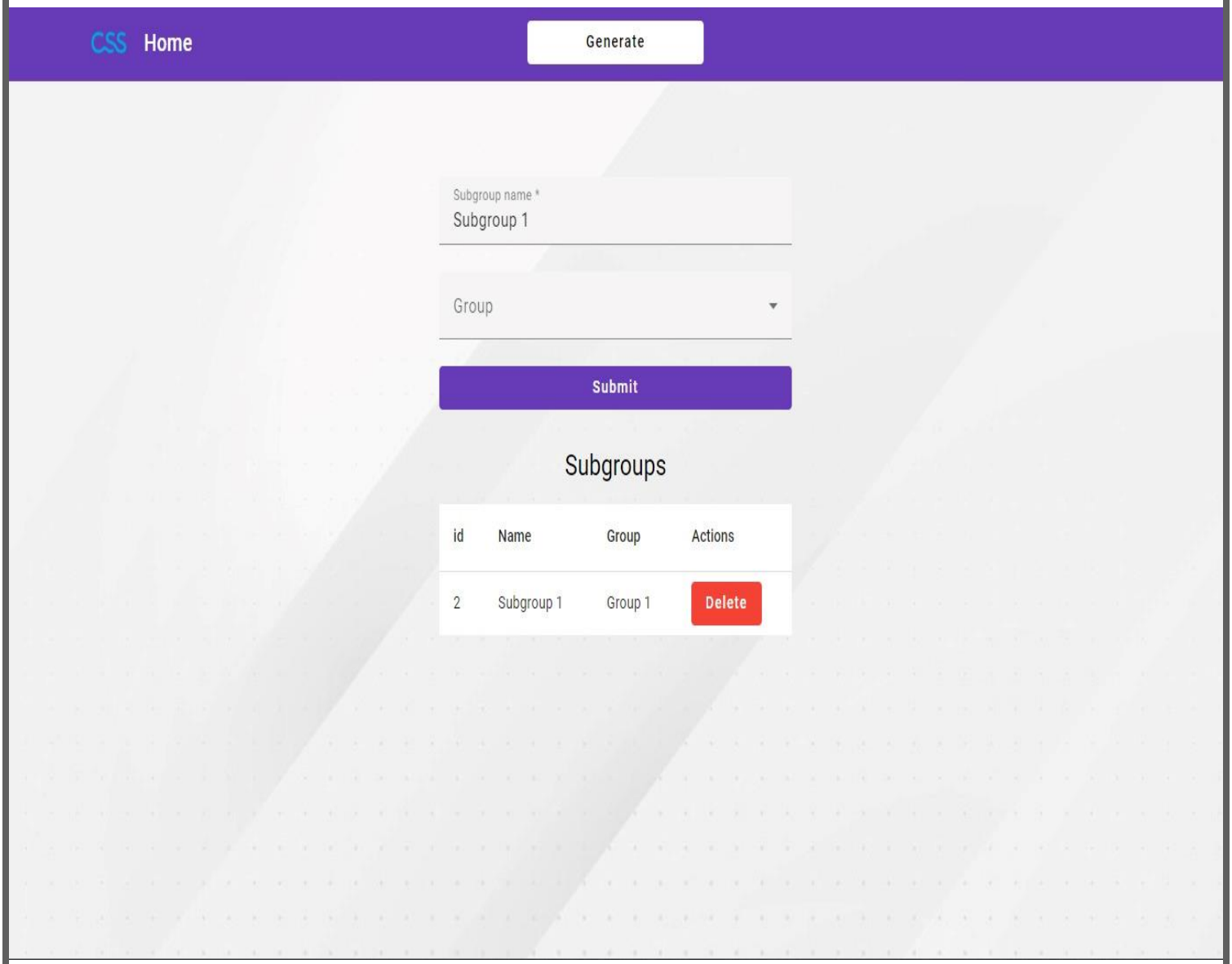


Figure 15 subgroup page

Activities page

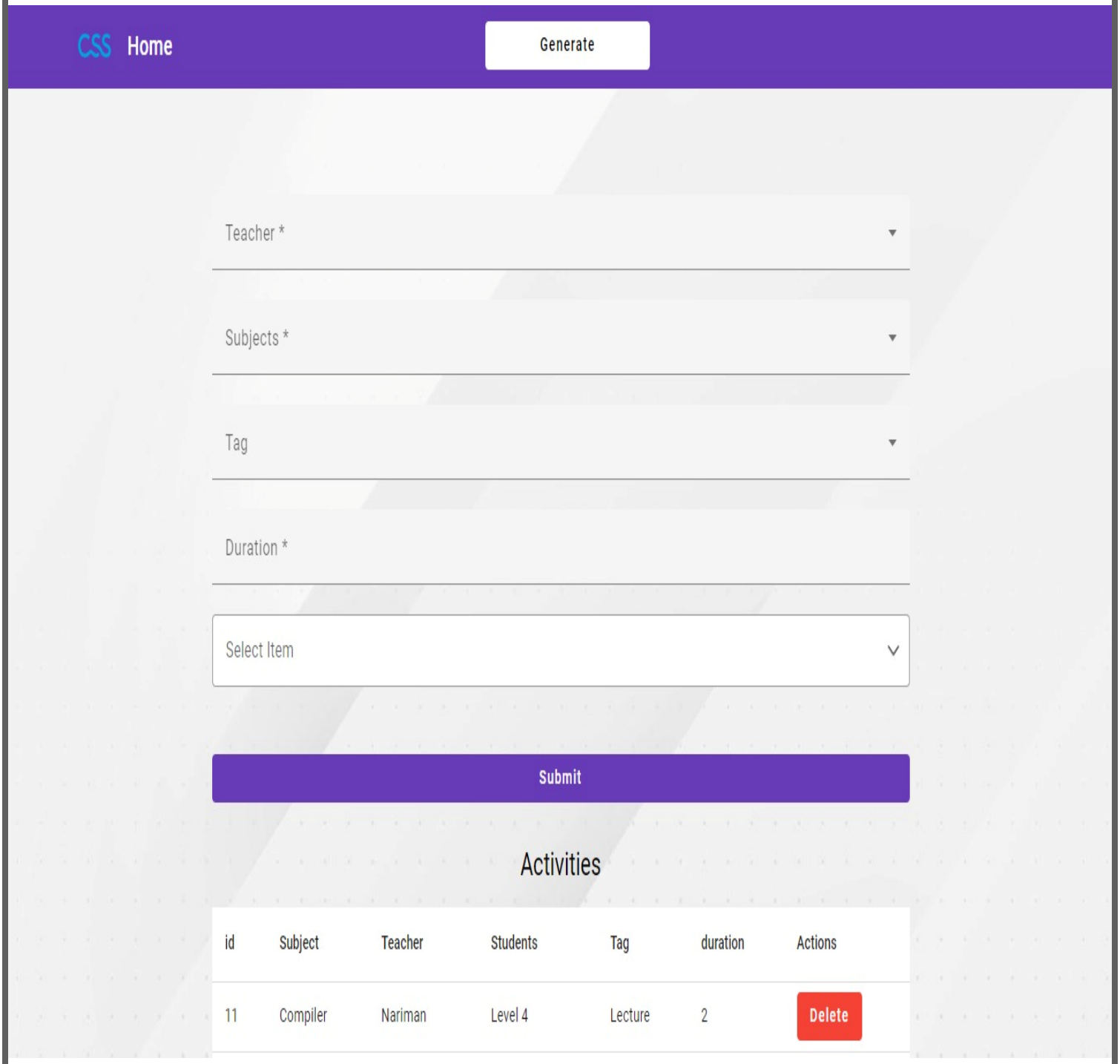


Figure 16 activities page

Table Generated

October 6 University

	Level 4						
	9	10	11	12	1	2	3
Monday	Marketing And Selling Lecture Sherief Mohamed			Advanced DBMS Lecture Hesham Sakr			---
Sunday	B Multimedia Section Ahmed Abdelhafez IS Computer Security Lecture Eman sanad	B Multimedia Section Ahmed Abdelhafez IS Computer Security Lecture Eman sanad	CS Compiler Theory Lecture Nariman IS Computer Security Lecture Eman sanad	CS Compiler Theory Lecture Nariman	CS Compiler Theory Lecture Nariman IS Decision Support System Lecture Asmaa Saad	B Advanced DBMS Section Ahmed Abdelhafez F Computer Security Section Rawan Mostafa IS Decision Support System Lecture Asmaa Saad	B Advanced DBMS Section Ahmed Abdelhafez F Computer Security Section Rawan Mostafa IS Decision Support System Lecture Asmaa Saad
Tuesday	---	G Computer Security Section Yara Tamer	C Advanced DBMS Section Hagar Mohamed G Computer Security Section Yara Tamer	C Advanced DBMS Section Hagar Mohamed	C Multimedia Section Ahmed Abdelhafez F Multimedia Section Afnan Abdelfattah	B Compiler Theory Section Mohamed helmy C Multimedia Section Ahmed Abdelhafez F Multimedia Section Afnan Abdelfattah G Decision Support System Section Hagar Mohamed	B Compiler Theory Section Mohamed helmy G Decision Support System Section Hagar Mohamed
Wednesday	A Computer Security Section Rabab Mahmoud C Compiler Theory Section Magy Makram I Advanced DBMS Section Ahmed Abdelhafez	A Computer Security Section Rabab Mahmoud C Compiler Theory Section Magy Makram I Advanced DBMS Section Ahmed Abdelhafez	Multimedia Lecture Bassem sheta			A Multimedia Section Ahmed Abdelhafez	A Multimedia Section Ahmed Abdelhafez
Thursday	H Decision Support System Section Hagar Mohamed	A Advanced DBMS Section Ahmed Abdelhafez F Compiler Theory Section Mohamed helmy H Decision Support System Section Hagar Mohamed	A Advanced DBMS Section Ahmed Abdelhafez F Compiler Theory Section Mohamed helmy H Computer Security Section Yara Tamer	CS Computer Security Lecture Eman sanad H Computer Security Section Yara Tamer I Decision Support System Section Hagar Mohamed	CS Computer Security Lecture Eman sanad I Decision Support System Section Hagar Mohamed	CS Computer Security Lecture Eman sanad	---
	Generated BY CSS 6.8.7. Generated on 6/15/23 2:37 PM						

Figure 17 Table Generated

8.2 project code

```
import { Table, Model, Column, DataType,HasOne,HasMany } from "sequelize-typescript";
import { Subjects } from "./subjects.model";
import {Activities} from './activities.model';

@Table({
  tableName: "teachers",
})
export class Teachers extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;

  @HasOne(() => Subjects)
  subjects!: Subjects;

  @HasMany(() => Activities)
  activities!: Activities[];
}
```

```typescript
import { Table, Model, Column, DataType,BelongsTo,ForeignKey,HasMany } from "sequelize-typescript";
import { Teachers } from './teachers.model';

import {Activities} from './activities.model';

@Table({
  tableName: "subjects",
})
export class Subjects extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;

  @ForeignKey(() => Teachers)
  @Column({
    type: DataType.INTEGER,
    allowNull: false,
    unique: true,
     onDelete: 'CASCADE',
  })
  teacher_id!: number;

  @BelongsTo(() => Teachers)
  teacher!: Teachers;

  @HasMany(() => Activities)
  activities!: Activities[];
}
```

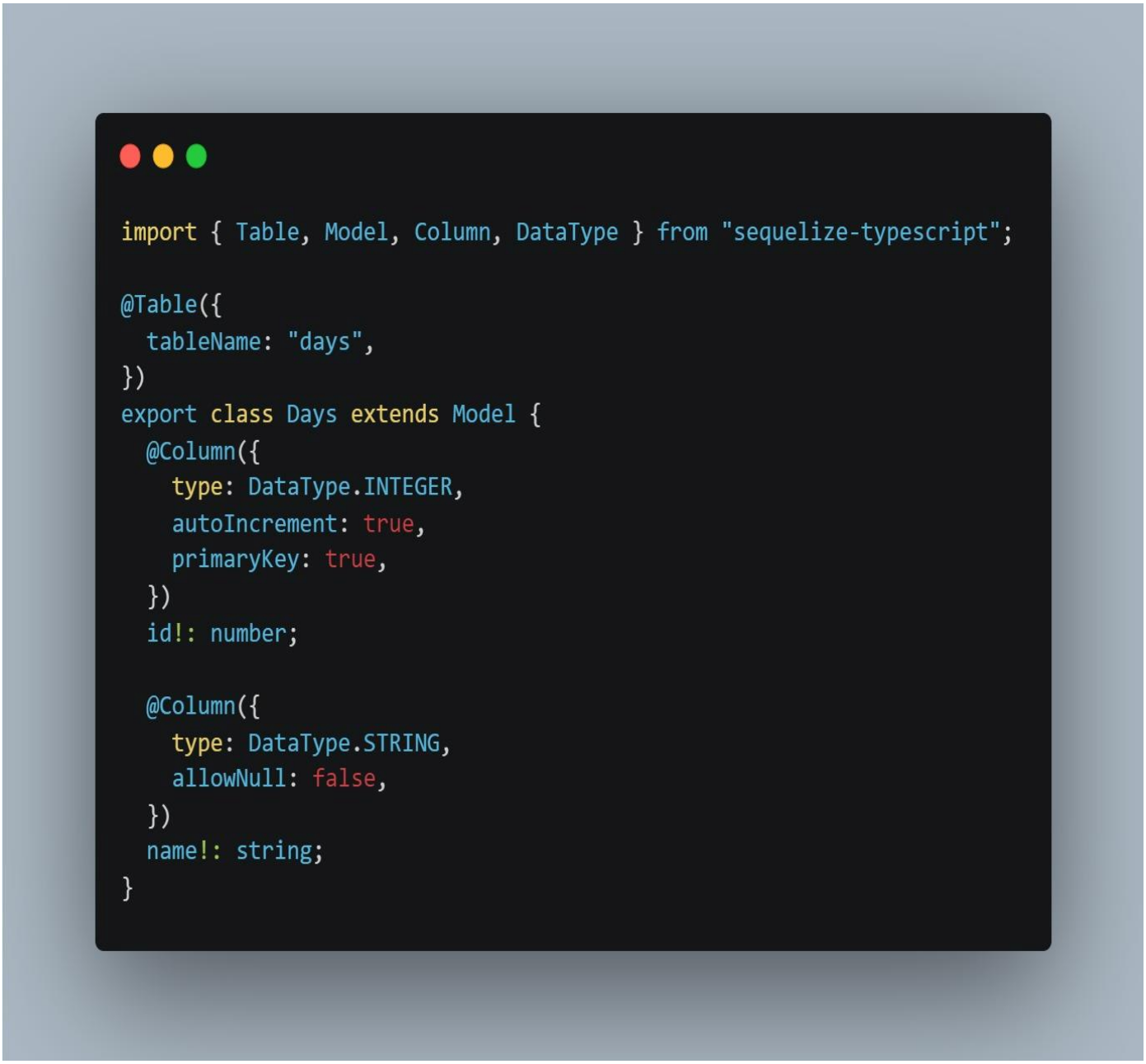
```
import {
  Table,
  Model,
  Column,
  DataType,
  ForeignKey,
  BelongsTo
} from "sequelize-typescript";

import { Groups } from "./groups.model";

@Table({
  tableName: "subgroups",
})
export class Subgroups extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @ForeignKey(() => Groups)
  @Column({
    type: DataType.INTEGER,
    allowNull: false,
    onDelete: 'CASCADE',
  })
  group_id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;

  @Column({
    type: DataType.INTEGER,
    allowNull: false,
  })
  students_no!: number;

  @BelongsTo(() => Groups)
  group!: Groups;
}
```

```ts
import { Table, Model, Column, DataType,HasMany } from "sequelize-typescript";

import { Groups } from "./groups.model";


@Table({
  tableName: "levels",
})
export class Levels extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;

  @Column({
    type: DataType.INTEGER,
    allowNull: false,
  })
  students_no!: number;

  @HasMany(() => Groups)
  groups!: Groups[];
}
```

```
import { Table, Model, Column, DataType } from "sequelize-typescript";

@Table({
  tableName: "hours",
})
export class Hours extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;
}
```

```
import { Table, Model, Column, DataType } from "sequelize-typescript";

@Table({
  tableName: "halls",
})
export class Halls extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;

  @Column({
    type: DataType.INTEGER,
    allowNull: false,
  })
  capacity!: number;
}
```

```
import {
  Table,
  Model,
  Column,
  DataType,
  ForeignKey,
  BelongsTo,
  HasMany
} from "sequelize-typescript";

import { Levels } from "./levels.model";
import { Subgroups } from "./subgroups.model";

@Table({
  tableName: "groups",
})
export class Groups extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @ForeignKey(() => Levels)
  @Column({
    type: DataType.INTEGER,
    allowNull: false,
    onDelete: 'CASCADE',
  })
  level_id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;

  @Column({
    type: DataType.INTEGER,
    allowNull: false,
  })
  students_no!: number;

  @BelongsTo(() => Levels)
  level!: Levels;

  @HasMany(() => Subgroups)
  subgroups!: Subgroups[];
}
```



```
import { Table, Model, Column, DataType } from "sequelize-typescript";

@Table({
  tableName: "days",
})
export class Days extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  name!: string;
}
```

```
import {
  Table,
  Model,
  Column,
  DataType,
  ForeignKey,
  BelongsTo,
} from "sequelize-typescript";


import { Teachers } from "./teachers.model";
import { Subjects } from "./subjects.model";

@Table({
  tableName: "activities",
})
export class Activities extends Model {
  @Column({
    type: DataType.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  })
  id!: number;

  @ForeignKey(() => Teachers)
  @Column({
    type: DataType.INTEGER,
    allowNull: false,
    onDelete: 'CASCADE',
  })
  teacher_id!: number;

  @ForeignKey(() => Subjects)
  @Column({
    type: DataType.INTEGER,
    allowNull: false,
    onDelete: 'CASCADE',
  })
  subject_id!: number;

  @Column({
    type: DataType.STRING,
    allowNull: false,
  })
  students!: string;

  @Column({
    type: DataType.INTEGER,
    allowNull: false,
  })
  duration!: number;

  @Column({
    type: DataType.STRING,
    allowNull: true,
  })
  tag!: string;

  @BelongsTo(() => Teachers)
  teacher!: Teachers;

  @BelongsTo(() => Subjects)
  subject!: Subjects;
}
```

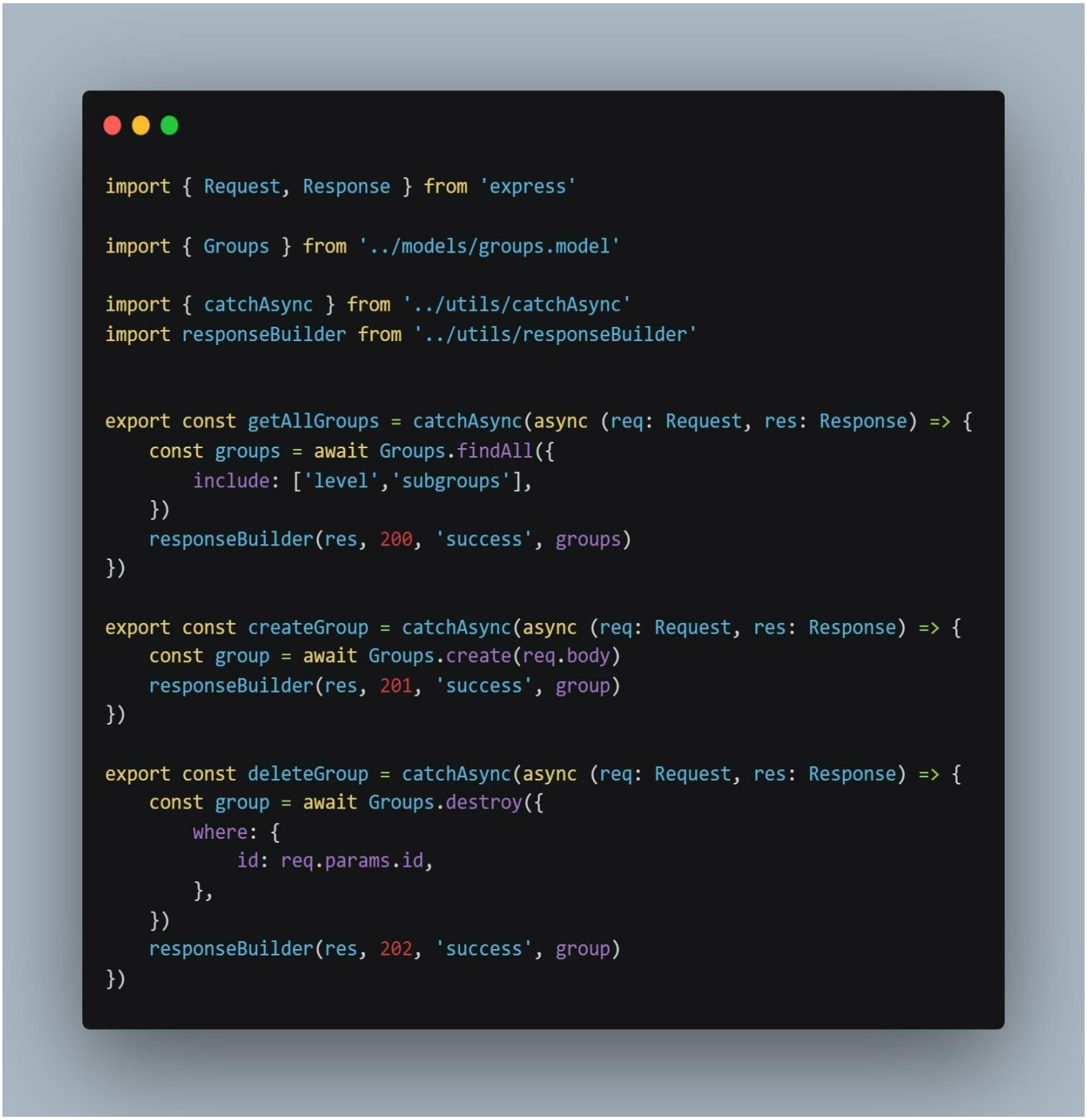
```
import { Request, Response } from 'express'

import { Subjects } from '../models/subjects.model'
import { Teachers } from '../models/teachers.model'

import { catchAsync } from '../utils/catchAsync'
import responseBuilder from '../utils/responseBuilder'

export const getAllSubjects = catchAsync(
   async (req: Request, res: Response) => {
      const subjects = await Subjects.findAll({
         include: [
            {
               model: Teachers,
               include: ['activities'],
            },
            'activities',
         ],
      })
      responseBuilder(res, 200, 'success', subjects)
   }
)

export const createSubject = catchAsync(async (req: Request, res: Response) => {
   const subject = await Subjects.create(req.body)
   responseBuilder(res, 201, 'success', subject)
})

export const deleteSubject = catchAsync(async (req: Request, res: Response) => {
   const subject = await Subjects.destroy({
      where: {
         id: req.params.id,
      },
   })
   responseBuilder(res, 202, 'success', subject)
})
```

```
import { Request, Response } from 'express'

import { Teachers } from '../models/teachers.model';
import { Subjects } from '../models/subjects.model';

import { catchAsync } from '../utils/catchAsync'
import responseBuilder from '../utils/responseBuilder'

export const getAllTeachers = catchAsync(async (req: Request, res: Response) => {
    const teachers = await Teachers.findAll({
        include: [{
            model: Subjects,
            include: ['activities']
        },"activities"]
    })
    responseBuilder(res, 200, 'success', teachers)
});

export const createTeacher = catchAsync(async (req: Request, res: Response) => {
    const teacher = await Teachers.create(req.body)
    responseBuilder(res, 201, 'success', teacher)
});

export const deleteTeacher = catchAsync(async (req: Request, res: Response) => {
    const teacher = await Teachers.destroy({
        where: {
            id: req.params.id,
        },
    })
    responseBuilder(res, 202, 'success', teacher)
});
```

```ts
import { Request, Response } from 'express'

import { Subgroups } from '../models/subgroups.model'
import { Groups } from '../models/groups.model'

import { catchAsync } from '../utils/catchAsync'
import responseBuilder from '../utils/responseBuilder'

export const getAllSubgroups = catchAsync(async (req: Request, res: Response) => {
    const subgroups = await Subgroups.findAll({
        include: {
            model: Groups,
            include: ['level'],
        },
    })
    responseBuilder(res, 200, 'success', subgroups)
})

export const createSubgroup = catchAsync(async (req: Request, res: Response) => {
    const subgroup = await Subgroups.create(req.body)
    responseBuilder(res, 201, 'success', subgroup)
})

export const deleteSubgroup = catchAsync(async (req: Request, res: Response) => {
    const subgroup = await Subgroups.destroy({
        where: {
            id: req.params.id,
        },
    })
    responseBuilder(res, 202, 'success', subgroup)
})
```

```ts
import { Request, Response } from 'express'

import { Levels } from '../models/levels.model'
import { Groups } from '../models/groups.model'


import { catchAsync } from '../utils/catchAsync'
import responseBuilder from '../utils/responseBuilder'

export const getAllLevels = catchAsync(async (req: Request, res: Response) => {
   const levels = await Levels.findAll({
      include: {
         model: Groups,
         include: ['subgroups'],
      },
   })
   responseBuilder(res, 200, 'success', levels)
})

export const createLevel = catchAsync(async (req: Request, res: Response) => {
   const level = await Levels.create(req.body)
   responseBuilder(res, 201, 'success', level)
})

export const deleteLevel = catchAsync(async (req: Request, res: Response) => {
   const level = await Levels.destroy({
      where: {
         id: req.params.id,
      },
   })
   responseBuilder(res, 202, 'success', level)
})
```

```ts
import { Halls } from '../models/halls.model'
import { Request, Response } from 'express'
import { catchAsync } from '../utils/catchAsync'
import responseBuilder from '../utils/responseBuilder'

export const getAllHalls = catchAsync(async (req: Request, res: Response) => {
   const halls = await Halls.findAll()
   responseBuilder(res, 200, 'success', halls)
})

export const createHall = catchAsync(async (req: Request, res: Response) => {
  console.log(req.body);

   const hall = await Halls.create(req.body)
    responseBuilder(res, 201, 'success', hall)
})

export const deleteHall = catchAsync(async (req: Request, res: Response) => {
   const hall = await Halls.destroy({
      where: {
         id: req.params.id,
      },
   })
    responseBuilder(res, 202, 'success',hall)
})
```



```
import { Request, Response } from 'express'

import { Groups } from '../models/groups.model'

import { catchAsync } from '../utils/catchAsync'
import responseBuilder from '../utils/responseBuilder'


export const getAllGroups = catchAsync(async (req: Request, res: Response) => {
    const groups = await Groups.findAll({
        include: ['level','subgroups'],
    })
    responseBuilder(res, 200, 'success', groups)
})

export const createGroup = catchAsync(async (req: Request, res: Response) => {
    const group = await Groups.create(req.body)
    responseBuilder(res, 201, 'success', group)
})

export const deleteGroup = catchAsync(async (req: Request, res: Response) => {
    const group = await Groups.destroy({
        where: {
            id: req.params.id,
        },
    })
    responseBuilder(res, 202, 'success', group)
})
```

```ts
import { Request, Response } from 'express'

import { Activities } from '../models/activities.model';

import { catchAsync } from '../utils/catchAsync'
import responseBuilder from '../utils/responseBuilder'


export const getAllActivities = catchAsync(async (req: Request, res: Response) => {
    const activities = await Activities.findAll({
        include: ['subject','teacher']
    })
    responseBuilder(res, 200, 'success', activities)
})

export const createActivity = catchAsync(async (req: Request, res: Response) => {
    const activity = await Activities.create(req.body)
    responseBuilder(res, 201, 'success', activity)
})

export const deleteActivity = catchAsync(async (req: Request, res: Response) => {
    const activity = await Activities.destroy({
        where: {
            id: req.params.id,
        },
    })
    responseBuilder(res, 202, 'success', activity)
})
```

Chapter 5: results and future work

9 Chapter 5: Results and Conclusion

9.1 Results

The desktop application successfully generated optimized university timetables using the genetic algorithm approach. Users could input course details, room availability, and constraints through the user interface. The application employed the genetic algorithm to find feasible and high-quality timetables that satisfied the input data and constraints.

The generated timetables were visually presented to users, providing an intuitive representation of course assignments, classrooms, and timeslots. Users could easily analyze and evaluate the generated timetables for their university.

The flexibility of the application allowed users to experiment with different optimization options, enabling them to find the best parameter settings for their specific timetabling requirements. The genetic algorithm efficiently explored the solution space and generated diverse timetables, ensuring the algorithm's ability to discover optimal or near-optimal solutions.

9.2 Conclusion

The desktop application built using the Electron framework successfully generated university timetables using a genetic algorithm approach. The application provided a user-friendly interface, allowing users to input data and constraints effortlessly. By employing the genetic algorithm, the application efficiently explored the solution space and produced high-quality timetables that met the university's scheduling requirements. The flexibility of the application allowed users to customize optimization options, enhancing the algorithm's performance and adaptability. The project's results contribute to the field of university timetabling optimization and demonstrate the potential of integrating genetic algorithms into desktop applications for timetable generation.

9.3 Recommendation And Future Work

In the future work, several important directions are identified for the timetable desktop app using Electron and the genetic algorithm. Firstly, scaling the app to serve the entire university is a key objective, allowing it to handle multiple faculties, departments, and courses simultaneously. Secondly, expanding the platform compatibility to include mobile apps for iOS and Android devices is crucial for wider accessibility and convenience. Enhancing the user interface and user experience will be prioritized, refining the design, layout, and navigation based on user feedback and usability testing. Moreover, further optimization of the genetic algorithm will be explored to improve efficiency and accuracy in generating optimized timetables, especially in handling complex scheduling constraints. Integrating the app with existing university systems, such as APIs or data connectors, will streamline data management and ensure up-to-date information. Collaboration features like timetable sharing and real-time collaboration will facilitate better coordination among users. Additionally, automating conflict resolution and integrating with popular timetable management tools will enhance the functionality and usability of the app. These future developments aim to provide a comprehensive and versatile scheduling solution for the entire university community.

10. References

- Alhuniti, Omar, et al. “Smart University Scheduling Using Genetic Algorithms.” Proceedings of the 2020 9th International Conference on Software and Information Engineering (ICSIE), 11 June 2020.
- Erben, Wilhelm. “Timetabling Using Genetic Algorithms.” Artificial Neural Nets and Genetic Algorithms, 25 Oct. 1995, pp. 30–32.
- Karova, M. “Solving Timetabling Problems Using Genetic Algorithms.” 27th International Spring Seminar on Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 2004., 13 May 2004.
- Pandey, Hari Mohan. “Solving Lecture Time Tabling Problem Using Ga.” 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), 5 Feb. 2016.
- Thanh, Nguyen Duc. “Solving Timetabling Problem Using Genetic and Heuristic Algorithms.” Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), 2007.
- Timilsina, Sandesh, et al. “Genetically Evolved Solution to Timetable Scheduling Problem.” International Journal of Computer Applications, vol. 114, no. 18, 2015, pp. 12–17.
- Sigl, B., et al. “Solving Timetable Scheduling Problem Using Genetic Algorithms.” IEEE Transactions on Image Processing, 16 June 2003.

ملخص المشروع

يعد إعداد الجدول الدراسية مشكلة صعبة تتطلب قدرًا هائلاً من الجهد والكثير من الوقت لصنعها يدويًا مرتين في السنة ، وربما أكثر إذا حدث خطاء فى أول مرتين. لذلك من أجل تسهيل قيام الجامعات بوضع الجداول الدراسية مرتين سنويًا ، ولتقليل الجهد المطلوب من صانعي الجدول الدراسية ، وكذلك لتقليل مقدار الوقت الضائع في يوم الطلاب مع ضمان عدم وجود تعارضات في نفس الوقت. مما يعيق مسار تعلمهم ، يجب أن يكون إعداد الجدول آليًا جزئيًا على الأقل.

جميع الخيارات المتاحة لنا والمتعلقة بالمجال في السوق بها عيوب خطيرة لا يمكن تجاهلها. على سبيل المثال ، قد يقوم النظام بعمل جدول به فجوات بين المواد الدراسية قد تصل لساعتين . أو قد يضع النظام جدولاً يفصل المادة الدراسية على يومين مختلفين. كلا النموذجين غير صالح للاستخدام في حالتنا.

العقبة التي يجب معالجتها في مشروعنا هي ان صناعة الجداول مشكلة من النوع NP-hard و التى لايمكن حلها بالخوارزميات العادية دون استهلاك قدر هائل من طاقة المعالجة والوقت. لذلك يجب استخدام خوارزمية من النوع heuristic algorithm او الخوريزميات التجريبية.

بعد تحليل أنظمة مماثلة ، يبدو أن الخوارزمية الجينية genetic algorithms يمكنها أن تولد جدولًا دراسيا فعالا فى فترة زمنية ما بين 0.2 نانوثانية وثانيتين. وهو تحسن كبير مقارنة بعمل الجدول يدويا الذى قد يستغرق يومين على الأقل.

مشروع التخرج

نظام وضع الجداول للفصل الدراسى

إعداد

الاسم: احمد السعيد عبد الصادق	قسم: علوم الحاسب
الاسم: احمد عبد الباسط محمد عبده	قسم: علوم الحاسب
الاسم: احمد على عبد الحميد على جبر	قسم: علوم الحاسب
الاسم: احمد فوزى حسين احمد ابو السعود	قسم: علوم الحاسب
الاسم: على إسماعيل حسن العيسوى	قسم: نظم المعلومات
الاسم: عمر تقى راضى إبراهيم	قسم: نظم المعلومات

تحت إشراف

د/ ممدوح فتح الله

د/ إبراهيم محمد

م/ محمد احمد حلمى

م/ اية إبراهيم الشافى