CS4013 – Group 11

Fabian Hartnett Ferreira – 24432024

Mikael Eramian - 24427233

Karen Kotowska - 24403148

Samuel Wilson – 24426881

Functional Requirements

Admin can:

- Log in using a username and password
- Create and manage modules, rooms and student groups
- Create and edit timetable slots (day, time, room, group, session type)

Lecturer can:

- Log in
- View their timetable

Student can:

- Log in
- View their timetable specific to their programme / group

Rationale:

- **The admin role is the only role that can modify data. This reduces the risk of inconsistent changes from multiple roles**
- **Lecturers and Students can only view timetables relevant to them. Allowing Lecturers to edit their own slots is something that could be added but we didn't to avoid conflict resolution between Admins and Lecturers**

Architecture – MVC

Model:

- User
- Admin
- Lecturer
- Student
- Programme
- Module
- Room
- Timetable
- TimetableSlot
- StudentGroup
- Subgroup
- SessionType

The model has all of the core data and business logic of the system. The classes store the information needed for making and viewing timetables. By keeping model separate, it allows it to be independent of UI decisions.

View:

- ConsoleUI
- StudentMenu
- LecturerMenu
- AdminMenu

The view menu contains menus specific to the role a user is logged in as. It is responsible for displaying information to the user. View interacts with the rest of the system through controllers. This can allow for changes like changing from console UI to GUI in the future without needing to make changes to the controller or model. The GUI would just call the same controller methods.

Controller:

- UserController: Login and role detection
- AdminController: Creation of modules, rooms, programmes, student groups and modifying timetable slots.

- TimetableController: Viewing timetables for students/lecturers

Controllers handle user actions and contains the main logic of the system. The classes accept input and process it which then updates the model. The separation ensures that changes to logic do not affect the user interface.

Repository:

- TimetableRepository - Interface
- CSVTimetableRepository – Loading/saving timetables from/to CSV files.

Responsible for loading and saving data from CSV files

Util:

- CSVUtil
- WeeksPattern

Overall the MVC Architecture helps ensure that our system is properly structured and scalable in the future.

Data Model

Classes:

User:

- Fields: id, name, password, role
- Used for authentication and authorisation

Admin, Lecturer and Student:

- Extends User
- Lecturer is associated with Modules and TimetableSlots
- Student belongs to a StudentGroup which is indirectly apart of a Programme

Programme:

- Represents a degree course (CS Year 1)
- Associated with Modules and StudentGroups

Module:

- Fields: code, title, weeks, lectureHours, labHours, tutorialHours, year, semester
- Linked to one or more timetableSlots

Room:

- Fields: number, capacity, type (lecture room, lab)

StudentGroup and Subgroup:

- StudentGroup represents a group of students that do the same programme
- Subgroup represents the lab / tutorial group the StudentGroups belong to. (EG. Lab 2D)

Timetable:

- Represents the complete Timetable
- Contains a collection of TimetableSlots

TimetableSlot:

- Fields: module, day, startTime, endTime, semester, weeks, room, lecturer, studentGroup, subgroup, type
- 1 scheduled session in the timetable (Monday, 10:00, 11:00, Semester2, week 3, Room CSG024, Lecture, CS4013, Michael English)

SessionType: (Enum)

- Lecture, Lab, Tutorial

WeeksPattern:

- Represents the teaching weeks for which a timetable slot runs.

Rationale:

Users and roles: We used a base User class that lets Admin, Student and Lecturer extend it. This avoids repeating code and lets each role have different behaviours

Programme, Module, Groups: We separated these because they represent different concepts. A Programme defines what students study. A Module represents the subject taught. StudentGroups and Subgroups make scheduling easier by grouping students rather than handling every individual.

Timetable and TimetableSlot: A Timetable is made up of TimetableSlots. A TimetableSlot object represents a single session. All relevant information can be added here

Relationships:

- A **Programme** has many **Modules**
- A **Programme** has many **StudentGroups**
- A **StudentGroup** has many **Students**
- A **Timetable** has many **TimetableSlots**
- A **TimetableSlot** has 1 **Module**, **Room**, **SessionType**, **StudentGroup**, **Lecturer**

Rationale:

- A programme has many modules because all students in that programme take those modules
- A timetable has many slots because each slot is 1 scheduled class
- A slot links to exactly 1 module, room, group and lecturer because classes don't use multiple rooms or lecturers at one time.

Data Storage / CSV:

- Admins.csv - stores admin user account
- Lecturers.csv: Stores lecturer accounts
- Students.csv: Stores student account and links them to their programme and group
- Modules.csv: Stores all module details
- Programmes.csv: Stores programme information
- Rooms.csv: Stores room information
- Timetables.csv: Stores timetable information (which timetable belongs to which programme or group)

- SessionTimetable.csv: Stores individual timetable slots with their day, time, room, room, group etc.

Rationale:

- We chose to use multiple small CSV files to serve a specific purpose rather than one large file.
- This allows testing to be more efficient as each file can be opened and checked individually. Changing data also affects only the file that is changed and not the other files. (You can make changes to rooms without affecting students for example)