ODTÜ METU

KUZEY KIBRIS KAMPUSU
NORTHERN CYPRUS CAMPUS

SNG 201 – INTRODUCTION TO SOFTWARE ENGINEERING

TEAM PROJECT - COURSE TIMETABLE GENERATOR

Ahmet Kerem İnce 2690659

Berrak Yıldırım 2690964

Taha Turkay Aktaş 2640274

Onur Pınarbaşı 2640662

Damla Yıldız 2640746

Table of Contents

1. General Description of the System:

This is a course schedule generator. It is a time-saving tool for people who want to generate course schedules. It helps students by allowing them to choose and add courses from set lists while checking for scheduling conflicts at the same time. It helps students to see the wide picture of the basic timetable at the end of the program. It provides a practical solution to the course management process.

2. Requirements:

Functional Requirements

1. **Load courses from file to program:** It imports course information from a file to program.
2. **Allow the user to select courses as many times as requested:** It allows the user to select as many courses as requested.
3. **Detect and eliminate conflicts:** It detects and prevents course conflicts.
4. **Display timetable:** It shows course schedule in an organized format.
5. **Display added courses:** It makes a list of courses added by the user.

Non-Functional Requirements

1. **Dynamic Suggestions:** It suggests courses that do not conflict.
2. **Usability:** The program should be easy and understandable.
3. **Error Handling:** It shows clear error messages on invalid inputs.
4. **Modularity:** It designs the code structure so that it can be easily updated.
5. **Portability:** It should work on any device with Python installed.
6. **Scalability:** More courses or hours should be able to be added.
7. **Performance:** It process' more than 100 lessons quickly.
8. **Reliability:** It recognizes conflicts accurately in all situations.

3. Personas:

Persona 1: Mustafa Kemal, a University Student

Age: 18

Education Level: Junior

Major: Software Engineering

University: Middle East Technical University NCC

Location: Kalkanlı, TRNC

Background:

Mustafa Kemal is a first-year Software Engineering student at METU NCC. He has great difficulties with time management and is often late for meetings or appointments. He wants to be organized and punctual to prevent this problem from affecting his university life.

As a first-year student, he is not used to his intensive program. He constantly confuses the days and times of his classes. This situation can lead to failing a class, and make him feel very anxious and stressed. To address this issue, he wants to use a scheduling program where he can manually select the course and section number, that he enrolled in and print a timetable for him. This program will allow him to see easily. The only thing he should do is simply enter the course name and section number. Thus he can quickly check the classes without any confusion.

Persona 2: Enver Tilki, a University Lecturer

Age: 45

Education Level: PhD in Computer Science

Position: Prof Associate Lecturer

University: Middle East Technical University NCC

Location: Kalkanlı, TRNC

Background:

Prof Dr Enver Tilki is an associate prof at METU NCC, specializing in computer science at METU Ankara. He has many responsibilities such as giving lectures, grading the final exams, attending meetings, and doing research. He has a very busy schedule that he has to balance and do everything on time. He gives lectures to two different main departments which are software and computer engineering and also he has to teach a course to all other departments in the entire university that is compulsory for everyone.

To keep his demanding schedule organized, he needs a timetable management program.

This timetable management program needs to be simple because he has no time to spend learning it. He only needs to enter the course and section number that he gives to students as input to the program. After the program displays his schedule, he will use it to manage his time, attend meetings, and conduct research more efficiently.

4. Features:

Feature's Name: Course Representation

Description: Course representation is a feature that uses Course class to represent courses with details such as course name, section number, course's start and end hours, and day of the week that the course is held.

Action: It configures the course object's characteristics during creation according to the user's input.

Activation: It activates when a new course object is added.

Output: It prepares the course object with the course's details to use in other features as the output.

Feature's Name: Timetable Initialization

Description: Timetable Initialization is a feature that creates an empty timetable from Monday to Friday, 8.00 AM to 5.00 PM.

Action: It is initialized as a dictionary with weekdays as keys with a list of open slots for hourly ranges included for each day of the week.

Activation: It starts when the program starts.

Output: It prepares an empty timetable that is ready for updates.

Feature's Name: Interactive Course Management

Description: Interactive Course Management is a feature that allows users to add courses interactively while using the update_timetable function.

Action: It updates the timetable according to the user's selection of the course to be added.

Activation: It is called when the user inputs a course to add.

Output: It updates the timetable and displays success messages or there is an error it displays.

Feature Name: Course Display

Description: Course Display is a feature that provides a clear and formatted way of displaying the timetable also the user's final schedule.

Action: It displays free slots and course information in a grid format it also prints the schedule.

Activation: It starts when the user requests to view the timetable.

Output: It displays the timetable that is arranged by time and day.

Feature Name: File Handling for Course Loading

Description: File Handling for Course Loading is a feature that reads the course data from an external file (courses.txt) and creates source objects.

Action: It parses each line of the file to create course objects for valid data.

Activation: It triggers during the program initialization or when the user specifies a file.

Output: It is a list of approved objects for scheduling.

Feature Name: User Input Validation

Description: User Input Validation is a feature that handles errors in user input to ensure valid operations.

Action: It checks for valid input during the course addition.

Activation: It starts when an interactive input is received.

Output: It displays an error message for invalid inputs.

Feature Name: Readable Timetable

Action: It formats the timetable as a grid with labeled rows and columns.

Activation: It starts when the user requests to display the timetable.

Output: It is a structured table with the scheduled courses and time slots.

Feature's Name: Find All Matching Courses

Description: Find All Matching Courses is a feature that finds courses that match with the given input by the user.

Action: It searches the course list for matches and returns them.

Activation: It triggers when the user selects a course to add.

Output: It is a list of matching course objects.

Feature's Name: Final Results Display

Description: Final Results Display is a feature that displays the final timetable.

Action: It summarizes and organizes the user's selected courses and timetable.

Activation: It triggers at the end of the program.

Output: It shows the finalized timetable as output.

Feature's Name: Main Program Control Loop

Description: The Main Program Control Loop is a feature that manages the entire flow of the program, including course selection, conflict handling, and finalization.

Action: It coordinates user interaction and activates different functionalities.

Activation: It runs automatically when the program runs.

Output: It manages the program's inputs.

Feature's Name: Dynamic Updates During Execution

Description: Dynamic Updates During Execution is a feature that allows for real-time addition of courses to the schedule

Action: It updates the timetable interactively according to the user's input.

Activation: During the course addition in the program.

Output: It is a timetable that is updated dynamically.

Feature's Name: Sorting of Schedule

Description: Sorting of schedule is a feature that ensures the user's schedule is displayed in a chronological and organized order.

Action: It sorts courses by day and starting time before displaying them in the final schedule.

Activation: It triggers when displaying the user's final schedule.

Output: It is a neatly sorted schedule for better readability.

Feature's Name: Error Handling for File Operations

Description: Error Handling for File Operations is a feature that handles potential errors when loading courses from a file, such as malformed data or missing files.

Action: It provides feedback for issues like missing or invalid course data.

Activation: It is triggered when the program attempts to load course data from a file.

Output: It logs errors and ensures the program continues gracefully.

Feature's Name: Program Exit Handling

Description: Program Exit Handling is a feature that ensures a clean exit from the program with proper messaging and final updates.

Action: It provides a confirmation prompt before exiting it also saves the if needed the user's progress.

Activation: It is triggered when the user decides to exit the program.

Output: It is a program termination message.

Feature's Name: Help or Instructions Feature

Description: Help or Instructions Feature is a feature that provides a guide on how to use the program, including the expected inputs from the user and available commands as well.

Action: It displays instructions when requested by the user or at the start of the program.

Activation: It is triggered at the beginning of the program or via a specific command.

Output: It displays a list of instructions for using the program.

5. User Stories:

User Story 1: First-Year Student Course Selection

As a first-year student, I want to create a table showing the days and times of my whole semester program.

User Story 2: First-Year Student Course Conflict Prevention

As a first year student, I don't want to clash my lessons, so if I add a lesson at the same time on the same day, I want the system not to add it automatically.

User Story 3: Professor's Schedule Management

As a busy professor, I would like to enter all the lectures I teach into the program so that I can spend time and work efficiently on meetings and my research.

6. Software Architecture:

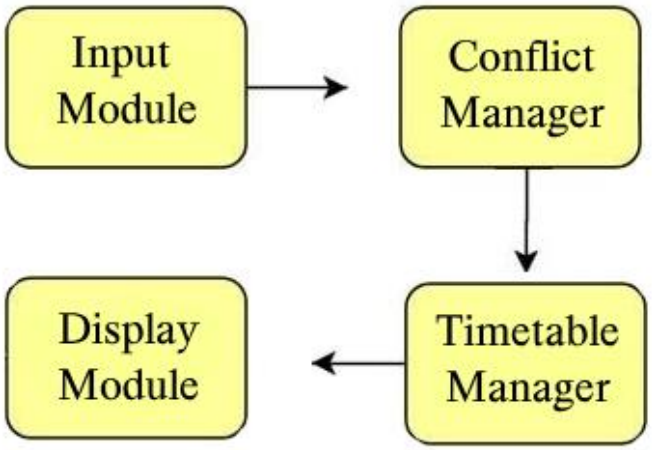
Description

The Timetable Generator app helps users organize their course schedules and makes sure there are no conflicts. Here's how it's designed:

1. **Layers and Components**
    - **Input Layer:** This part lets users add courses and manage their timetable.
    - **Business Logic Layer**: It checks if there are any conflicts in the schedule. If there's a conflict, it shows an error message.
    - **Display Layer:** This part shows the timetable and courses to the user.
2. **Data Management**
    - **Course Class:** This stores details about the course, like the name, section, start and end times, and the day of the week.
    - **Timetable Dictionary:** The schedule is managed using a dictionary.
        - The outer dictionary has the days of the week as keys.
        - The inner list shows hourly slots that are either free or occupied.
3. **Interaction Flow**
    - **Initialization:**
        - The timetable and courses are loaded from a file using load_courses_from_file.
        - A new, empty user schedule is created.
    - **Course Management:**
        - The available courses are shown to the user.
        - The user can select courses, and the system will place it to the appropriate day and hour.
    - **Timetable Update:**
        - The timetable is updated based on the user's choices using the update_timetable function.

- **Final Output:**
    - The timetable is displayed with the display_timetable function, and the final schedule is printed using the print_schedule function

Pictorial Representation:



7. Programming Language and Development Tools:

- **Programming Language:** Python for both frontend and backend.
- **Development Tools:** PyCharm Community Edition.
- **Justification:** Because Python is object-oriented, it is easy to work on and this language model supports many libraries to work with. For the front end, a simple text-based menu is enough for us.

8. Agile Roles:

- **Product Owner:** Hüseyin Sevay
- **Scrum Master:** Berrak Yıldırım
- **Developers:** Ahmet Kerim İnce, Damla Yıldız, Onur Pınarbaşı, Taha Turkay Aktaş

9. Sprints:

Sprint Log (Expected Deliverables):

- Design the Course class with attributes (name, department, start time, end time, day).
- Develop the initialize_timetable function.
- Write the load_courses_from_file function to process course data.
- Implement the is_conflict function for time clash detection.

Scrum Meetings for Sprint 1

Scrum 1

- **Date**: November 19, 2025
- **Time**: 5:00 PM
- **Items we chose from the sprint log**:
  - Design the Course class.
  - Implement the initialize_timetable function.

Scrum 2

- **Date**: December 5, 2025
- **Time**: 5:40 PM
- **Items we chose from sprint log**:
  - Test and validate the initialize_timetable function.
  - Develop the load_courses_from_file function.
  - Implement the is_conflict function

Sprint Reviews

1. Work Completed:

- **Course Class:** We presented the Course class like name, department, start time, end time, and day. Stakeholders confirmed the class properties.
- **Timetable Initialization:** We showed the initialize_timetable function, which sets up an empty timetable structure.
- **Data Loading:** We showed the load_courses_from_file function, successfully importing course data from a .txt file.
- **Conflict Detection:** We reviewed is_conflict function, and explained the logic behind it.

2. Feedback from Stakeholders:

- Stakeholders liked the clear structure of the Course class and the smooth data import process.
- They suggested adding more attributes, like course credits or capacity, for future use.
- They suggested us to focus on the conditions of is_conflict and change it to a less complex one.

3. **Updates to the Product Backlog:**

- We improved the is_conflict.

Sprint Retrospective

What Went Well:

• The transfer of the program of the elements in the txt files was successful.

• The conflict logic of the programs was established.

What Didn't Went Well:

• We haven't implemented the conflict function in this sprint.

What Could Be Improved:

• More courses can be added to the text file and it can be useful for more people

Stage 2: User Interaction and Course Selection

Sprint Initiation Meeting

Due to the decision we made in the previous sprint, we wanted the user to add a new course in each cycle and the courses he could choose were decreasing in each stage. Therefore, to make the user's job easier, we decided to show the current course list to the user in each cycle. We aimed to improve conflict control and complete the course by adding functions and dealing with possible errors.

Sprint Log (Expected Deliverables):

- Create a text-based user interface for course selection.
- Implement the update_timetable function.
- Integrate is_conflict for real-time conflict checking.
- Add error handling for invalid inputs.

Scrum Meetings for Sprint 2

Scrum 3

- **Date**: December 12, 2025
- **Time**: 11:30 AM
- **Items we chose from the sprint log**:
  - Creating the text-based user interface.
  - Begin implementing the update_timetable function.

Scrum 4

- **Date**: December 27, 2025
- **Time**: 7:00 PM
- **Items we chose from the sprint log**:
  - Implementing update_timetable with is_conflict.
  - Adding error handling for invalid inputs

Sprint Reviews

1. Work Completed:

- **Timetable Updates:** We showed the **update_timetable** function, which updates the course list.
- **Conflict Detection:** We improved the function according to the previous feedbacks from stakeholders.
- **Error Problem:** We organized the code again to handle errors.

2. Feedback from Stakeholders:

- Stakeholders liked the changes after the first feedback and the new updater function.
- Suggested improving course filtering but they liked the clear and non-complex structure of the program.

3. Product Backlog Updates:

- Improved error handling.

Sprint Retrospective

What Went Well:

• Time–table logic has been established.

• The conflict problem that we could not solve in the previous sprint has been resolved and we have solved it by using it in our update_timetable function.

What Didn't Went Well:

• The timetable took more time than we expected. Since there was a lot of time between scrums, we spent extra time to remember the project.

What Could Be Improved:

• Filtering can be improved, in another product-focused project, for example, more possibilities can be eliminated at each stage according to credit requirements.

• It can be made into a more secure code by adding error possibilities and conditions.

Stage 3: Timeline Visualization and Completion

Sprint Initiation Meeting

In the last stage, we aimed to present the created timeline and a list of selected courses separately to the user. Finally, we aimed to evaluate what we had done from beginning to end and make our code more organized. We planned to write our report together in our evaluation at our last meeting.

Sprint Log (Expected Deliverables):

- Write the display_timetable function to visualize the schedule.
- Create the print_schedule function for a detailed course list.
- Finalize user prompts for adding new courses or exiting.

Scrum Meetings for Sprint 3

Scrum 5

- **Date**: January 17, 2025
- **Time**: 8:00 PM
- **Items we chose from the sprint log**:
  - Implement the display_timetable function.
  - Begin working on the print_schedule function.

Sprint Reviews

1. Demonstration of Completed Features:

• **Timetable Display:** We have presented the final version of the timetable to the user.

• **Timetable Details:** We have shown the print_schedule function that prints a list of features of the courses selected by the user.

• **User Interaction System:** We have provided clarity on the process of exiting the loop and finishing when requested.

2. Feedback from Stakeholders:

• Stakeholders appreciated the accuracy and solution-oriented nature of the timetable.

Suggested improvements:

• Suggested a feature that can create an additional day-based list in case the timetable becomes more complex.

• Suggested the ability to export the timetable to PDF or other formats.

3. Updates to the Product Backlog:

• Improve the display_timetable function by increasing sorting and filtering.

• Improve the aesthetic design of the timetable by adding borders and color-coded days for better readability.

Sprint Retrospective

What Went Well:

• The printing of the time of courses is clear and understandable.

• The extra courses and time list that the user chose can be pretty useful.

What Didn't Went Well:

• We had to change the previous sprint functions to make them compatible with the print functions.

What Could Be Improved:

• Different types of print options can be added for other needs.

• The table can be made more aesthetic by adding lines around the table.

10. Conclusion:

Goals Achieved:

We have successfully completed all our plans. We have resolved conflicts. We have created a program that can create programs with lessons in .txt files.

Lessons Learned:

We have understood the importance of user feedback and being flexible.

Requirements:

We have met all the requirements. We have met all the requirements by leaving flexibility in the process we determined by prioritizing functionality and user needs.

11. Example Outputs:

Example Output of the Timetable:

Display Example

```
EEE 426 (Section 1) on Tuesday: 14:00-15:00
EEE 212 (Section 1) on Wednesday: 9:00-10:00
EEE 426 (Section 1) on Wednesday: 12:00-13:00
EEE 281 (Section 1) on Wednesday: 10:00-11:00
EEE 471 (Section 1) on Wednesday: 13:00-14:00
EEE 330 (Section 1) on Thursday: 9:00-10:00
EEE 463 (Section 1) on Thursday: 10:00-11:00
EEE 212 (Section 1) on Thursday: 11:00-12:00
EEE 201 (Section 1) on Friday: 8:00-9:00
EEE 301 (Section 1) on Friday: 12:00-13:00
EEE 493 (Section 1) on Friday: 13:00-14:00
EEE 463 (Section 1) on Friday: 14:00-15:00
EEE 445 (Section 1) on Friday: 15:00-16:00

Enter the name of the course you want to add (e.g., EEE 445): CNG 223
Enter the section of the course: 2
CNG 223 has been added to the timetable.
CNG 223 has been added to the timetable.
CNG 223 has been added to the timetable.

Do you want to add another course? (y/n):
```

Timetable Example

```
Enter the name of the course you want to add (e.g., EEE 445): HST 201
Enter the section of the course: 1
HST 201 has been added to the timetable.
Error: HST 201 timing is out of timetable bounds.

Do you want to add another course? (y/n): N

Final timetable:
Hour  Monday              Tuesday             Wednesday           Thursday            Friday
-------------------------------------------------------------------------------------------------------
8:00  Free                CNG 223 (Section 2) Free                MAT 219 (Section 1) Free
9:00  Free                CNG 223 (Section 2) Free                MAT 219 (Section 1) Free
10:00 Free                Free                Free                Free                Free
11:00 CNG 223 (Section 2) Free                Free                CNG 213 (Section 2) Free
12:00 Free                SNG 201 (Section 1) Free                CNG 213 (Section 2) Free
13:00 Free                Free                Free                SNG 201 (Section 1) Free
14:00 CNG 213 (Section 2) Free                Free                Free                Free
15:00 Free                Free                MAT 219 (Section 1) Free                Free
16:00 MAT 219 (Section 1) Free                MAT 219 (Section 1) Free                Free
17:00 HST 201 (Section 1) Free                Free                Free                Free
```

Final Schedule Example

```
Final schedule:
Day       Time                Course (Section)
-------------------------------------------------------------
Monday    11:00 - 12:00       CNG 223 (Section 2)
Monday    14:00 - 15:00       CNG 213 (Section 2)
Monday    16:00 - 17:00       MAT 219 (Section 1)
Monday    17:00 - 18:00       HST 201 (Section 1)
Monday    18:00 - 19:00       HST 201 (Section 1)
Thursday  8:00 - 9:00         MAT 219 (Section 1)
Thursday  9:00 - 10:00        MAT 219 (Section 1)
Thursday  11:00 - 12:00       CNG 213 (Section 2)
Thursday  12:00 - 13:00       CNG 213 (Section 2)
Thursday  13:00 - 14:00       SNG 201 (Section 1)
Tuesday   8:00 - 9:00         CNG 223 (Section 2)
Tuesday   9:00 - 10:00        CNG 223 (Section 2)
Tuesday   12:00 - 13:00       SNG 201 (Section 1)
Wednesday 15:00 - 16:00       MAT 219 (Section 1)
Wednesday 16:00 - 17:00       MAT 219 (Section 1)
```