SOFTWARE DESIGN DOCUMENT

for

SYLLABUS PROJECT

Prepared by **Team 10**

Canberk ÇOBAN	20210602022
Ahmet Eren SIRCAN	20210602057
Ege PELİNDAĞ	20200602040
Hakan UZUN	20200602051
Mehmet GÜLBAHAR	20210602210

Lecturer: **İlker Korkmaz**

December 2, 2023

Contents

1 Introduction

The Syllabus Application is a standalone desktop application designed to manage course syllabuses. The application allows users to efficiently create, edit, delete, and store syllabuses using a user-friendly graphical interface. The application is compatible with Windows systems and allows users to select Turkish or English for syllabus creation. The development is centered around the Java programming language, with JavaFX used for the graphical user interface.

According to the introduction section following requirements are met:

Functional Requirement 1: Users should be able to install the desktop application through an installer without requiring additional installations.

Non-Functional Requirement 1: The application shall be installed on a computer which has only JVM as a desktop application.

Non-Functional Requirement 3: The program shall support the Windows operating system.

2 Software Architecture

The Syllabus Application features a hybrid approach of several well-known software architectures for syllabus management. Built with JavaFX for the graphical user interface and Java for the backend. At its core, the application follows a modular architecture, separating key functionalities into distinct components. Since the application was developed with JavaFX, it also has influences from MVC architectural design. The model layer handles data management and storage, capturing the essence of syllabus content and supporting version control functionalities. The view layer, implemented using JavaFX, focuses on presenting a user-friendly interface for syllabus creation and interaction. The controller layer acts as the intermediary, managing user inputs, processing requests, and coordinating communication between the model and view.

2.1 Structural Design

As there is a single user for the application, the structural design is simplified. The application revolves around the Syllabus class, representing the course syllabuses. Each instance of the Syllabus class encapsulates information about a specific course. Course and Syllabus. These classes are purposefully structured to effectively handle and encapsulate the inherent relationship between courses and syllabuses within the application.

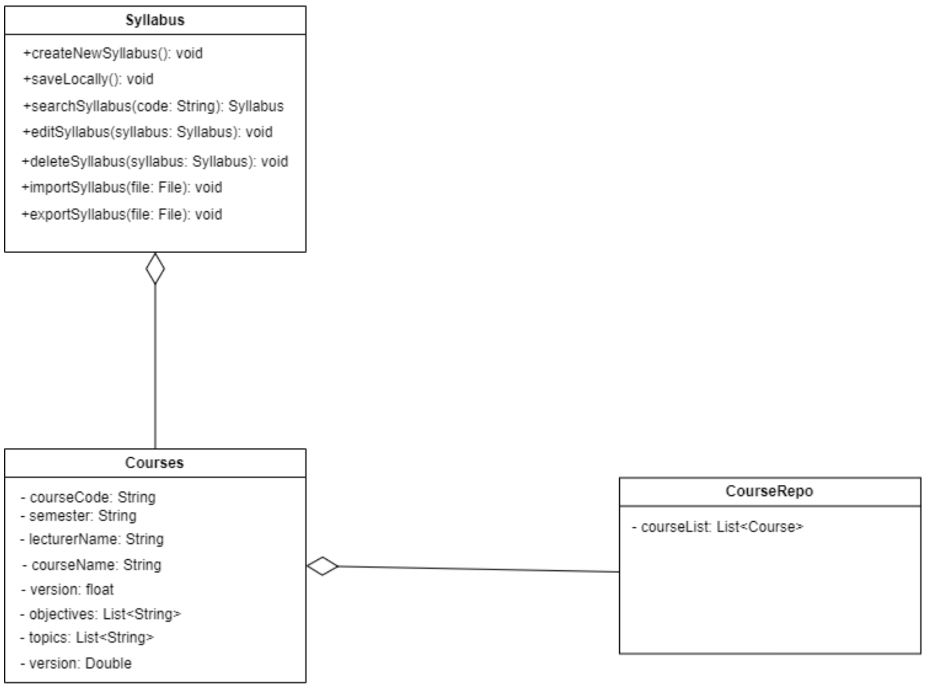

Syllabus
+createNewSyllabus(): void
+saveLocally(): void
+searchSyllabus(code: String): Syllabus
+editSyllabus(syllabus: Syllabus): void
+deleteSyllabus(syllabus: Syllabus): void
+importSyllabus(file: File): void
+exportSyllabus(file: File): void
Courses
- courseCode: String
- semester: String
- lecturerName: String
- courseName: String
- version: float
- objectives: List<String>
- topics: List<String>
- version: Double
CourseRepo
- courseList: List<Course>

Figure 1: Class Diagram

As illustrated in Figure 1, the Syllabus class serves as the primary entity, encompassing vital information such as course details, objectives, topics, version control details, and syllabus name. Each instance of the Syllabus class represents an individual course within the system.

Here's a simple example for creating a new **syllabus** object Since the syllabus contains many attributes, it should be taken into consideration that this representation **does not show** all the attributes:

> Syllabus SE302Syllabus = new Syllabus(courseCode = ”Se - 302”,Semester = ”Fall 2023”, lecturerName = ”İlker Korkmaz”, courseName = ”Principles of Software Engineering”, version = 1.0);

These meet the following requirements:

> **Functional Requirement 2:** The application should facilitate the creation of new syllabi via the program's interface, allowing users to save them locally.
> **Functional System Requirement:** Users will have the capability to create new syllabi using the program's interface or save syllabi prepared with the template to the system database.

2.2 Behavioral Design

2.2.1 Creating and Deleting Syllabuses

Requirement 2 emphasizes the app's user-friendly approach to creating a new syllabus. Clicking the "New Syllabus" button opens a blank template for users to fill in the necessary details. After completing this, the app saves the information into a secure JSON file, neatly organized in a database for easy retrieval. To enhance functionality, the **SyllabusRepo** class now includes a syllabus list storing course names and codes. Users can easily locate the syllabus using the unique course code through the **searchSyllabus** method. For deletion, users search for the course, view the syllabus, and click the delete button, initiating the **deleteSyllabus** method to seamlessly remove it from the database, simplifying syllabus management.

2.2.2 Editing Syllabuses

During the editing operation, users follow a specific sequence. Initially, users are required to utilize the **searchSyllabus** method, providing the course code they intend to edit. If the program successfully locates it will display the relevant information to the user. If the user wishes to proceed with the edit, an "Edit" button will be available at the top left of the page. Upon clicking it, the entire syllabus will become editable with the **editSyllabus** method. This method allows user to edit desired parts of the syllabus via GUI. Finally, the save button saves the filled form to the database and creates a new copy of the syllabus with a new incremented version number.

2.2.3 Searching Syllabuses

Requirements specify that the user should be able to search syllabuses via course code. Each Course has a unique course code. The **CourseRepo** class contains **courseList** to store the name and the code of the courses added into the system. User shall be able to access to desired syllabus via **searchSyllabus** method. This method takes course code as a parameter to search in **courseList** to find the corresponding syllabus.

2.2.4 Storing Syllabuses

Upon activating the **"Next"** button, the system will systematically record the relevant information on the corresponding page. Subsequently, upon executing the **"Save"** button, all data will be carefully stored within an **SQLite database**. Each course is assigned a **unique** identifier known as the **courseCode**, facilitating streamlined access and retrieval of specific course-related information. This structured approach ensures a straightforward and efficient management of data within the system.

2.2.5 Version Control

The initial iteration of a syllabus, upon its first saving, is assigned the version number **"1.0"**. Subsequent modifications to the syllabus, whether through edits or the creation of a new instance, trigger an incremental increase in the version number. The **version** attribute associated with each syllabus serves the purpose of tracking changes made to syllabuses within the system. Specifically, when a syllabus undergoes editing or a new one is created, the version number experiences an **increment of 0.1** to accurately reflect the evolving state of the syllabus.

2.2.6 Import/Export to JSON

The **export** method is designed to convert syllabus data from the **SQLite database** into a structured **JSON** format, capturing comprehensive details for versatile data storage. In contrast, the import method plays a vital role in processing **JSON** input data, adapting it to a suitable format for seamless integration into the SQLite database. This involves mapping and organizing the **JSON** data into the corresponding SQL schema, ensuring compatibility and integrity with the existing database structure. The import method ensures the accurate and efficient insertion of information into the SQLite database, facilitating smooth data integration.

3 Graphical User Interface

The interface was developed with SceneBuilder. The GUI is the user-friendly face of our project, making it easy for users to manage course content. With a straightforward menu and quick tools, it's designed for simplicity and convenience for the user. It handles key tasks like syllabus creation, syllabus searching, syllabus editing, syllabus deletion, and import/export functionalities.

User Interface Menu:

- **Syllabus**
  - **New Syllabus**
  - **Search Syllabus**
    - **Edit Syllabus**
    - **Delete Syllabus**
  - **Import/Export Syllabus**
- **Help**
  - **Instructions**
  - **About**
- **Exit**

A simple main menu is used for the main scene with the following functionalities: **New Syllabus**, **Search Syllabus**, **Edit Syllabus**, **Delete Syllabus**, and **Import/Export Syllabus**.

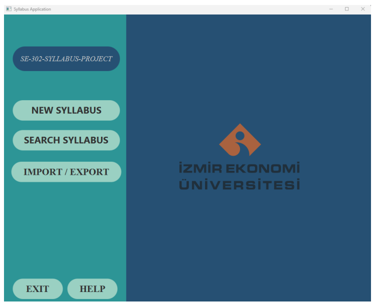
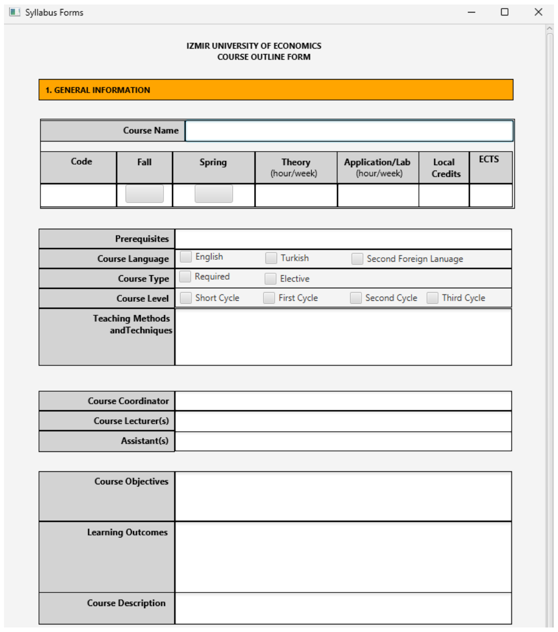


Figure 2: Main Menu

The Syllabus Application provides an easy-to-fill syllabus template for users to fill the course syllabuses. A syllabus is divided into five sections to fill the template step by step. Page 1 will look as follows.



Syllabus Forms

IZMIR UNIVERSITY OF ECONOMICS
COURSE OUTLINE FORM

1. GENERAL INFORMATION

Course Name	

Code	Fall	Spring	Theory (hour/week)	Application/Lab (hour/week)	Local Credits	ECTS

Prerequisites				
Course Language	☐ English	☐ Turkish	☐ Second Foreign Lanuage	
Course Type	☐ Required	☐ Elective		
Course Level	☐ Short Cycle	☐ First Cycle	☐ Second Cycle	☐ Third Cycle
Teaching Methods andTechniques				

Course Coordinator	
Course Lecturer(s)	
Assistant(s)	

Course Objectives	
Learning Outcomes	
Course Description	

Figure 3: Syllabus Form

By filling out the form users can save syllabuses into the system via **database storage**. In this way, existing syllabuses shall be displayed using the **search feature** of the system. The **search feature** will allow users to search through **Course codes** to find the syllabuses they search for. The **version control system** shall provide the version of each syllabus and previously created/edited syllabuses to be displayed to the user. After searching for a specific course there will be an **edit button** to change desired parts of a syllabus via the edit page of Interface. Since these figures are from the early development stage. GUI may change during the further development period of the application.

Functional Requirement 2: The application should facilitate the creation of new syllabi via the program's interface, allowing users to save them locally.

Functional Requirement 3: The application should be able to display the syllabi templates. So that the user can fill in the templates.

Functional Requirement 7: The user should be able to display the desired syllabus programs by searching via course code.

Non-Functional Requirement 5: The program shall have an info section that explains the application.

Non-Functional System Requirement: The Syllabus Project must feature a simple graphical user interface to enhance user productivity and reduce complexity.