Project Based Evaluation

Project Report
Semester-IV (Batch-2023)

Course-Enrollment Management System

Supervised by:
Dr. Susama Bagchi

Submitted By:
Yuvraj Gupta, 2310991326 (G16)
Krati Arora, 2310991331 (G16)
Vedant Sareen, 2310991318 (G16)
Prachi Singla, 2310991321 (G16)

Department of Computer Science and Engineering Chitkara University Institute of Engineering & Technology, Chitkara University, Punjab

INDEX

1. Introduction

In any academic institution, managing student course enrollments manually can lead to significant challenges such as data inconsistency, delays in processing, and a lack of transparency. With the increasing number of students and courses, a robust and automated system is essential to streamline operations. The Course Enrollment Management System is developed to overcome these problems by offering a digital, reliable, and easy-to-use solution to handle student and course records, manage enrollments, and track academic data effectively.

This system has been developed using Java programming with a command-line interface (CLI). It employs the principles of Object-Oriented Programming (OOP) and persists data using CSV (Comma-Separated Values) files, making it simple yet powerful for academic use cases.

1.1 Purpose

This system aims to automate and simplify the process of managing students, courses, and enrollments in academic institutions. The document outlines the functional and non-functional requirements, system design, and implementation details that serve as a guide for the development and evaluation of the software.

The main intention is to provide a centralized, command-line interface-based application that maintains accurate and persistent records of student-course interactions while showcasing fundamental concepts of Java and Object-Oriented Software Engineering.

1.2 Document Conventions

Convention	**Meaning**
Menu Selections	Navigation options or buttons users need to select (e.g., "Manage Students" → "Add Student").
ID Naming	Student IDs, Course Codes, and Enrollment IDs follow specific structured formats (e.g., STU001, CSC101).
Date Format	All dates are represented in the format YYYY-MM-DD.
Error Messages	Shown in **bold** and inform users about input validation errors, missing data, or processing failures.

Table 1.1 : Convention for the document

1.3 Intended Audience and Reading Suggestions

This document is intended for the following audience groups:

- **Faculty Evaluators**: To assess the student's understanding of software engineering principles and application of object-oriented concepts.
- **Students/Developers**: To understand the architecture, functionality, and development approach used in the Course Enrollment Management System.
- **Academic Institutions**: To review and adapt this system as a model for automating their own student-course enrollment processes.

Reading Suggestions:

- Start with the **Introduction** to understand the overall goal of the system.
- Refer to the **Problem Definition and Requirements** to grasp the scope and technical constraints.
- Explore the **Proposed Design / Methodology** section for diagrams and structure.
- Use the **Results** section to validate the functionalities via CLI screenshots and CSV outputs.

1.4 Project Scope

The scope of the **Course Enrollment Management System** is to provide a lightweight yet effective command-line tool for academic institutions to manage students, courses, and their enrollments. This system allows administrators to:

- Add, view, update, and delete student records.
- Add, view, and manage course information.
- Enroll students into specific courses and manage those enrollments.
- Store and retrieve data from **CSV files** for persistent storage.
- Interact with the system using a **Java-based CLI** without relying on any third-party database or GUI.

The system is designed primarily for **academic learning and prototyping**, focusing on demonstrating Java programming skills and software engineering design principles. While it is functional, it can be further extended to include graphical interfaces, authentication mechanisms, and database integration for real-world deployment.

2. Overall Description

2.1 Product Perspective

The Course Enrollment Management System is a **standalone, command-line based application** developed in Java. It is built as an educational tool that demonstrates the integration of core Java programming with real-world problem-solving through Object-Oriented Software Engineering principles. It does not rely on a database management system; instead, it uses **CSV format in .txt files** for persistent storage, making it lightweight and easy to deploy.

This system is **modular** in nature, with separate components responsible for managing students, courses, and enrollments. Each component can be modified or extended without affecting the rest of the system, thanks to the use of object-oriented design patterns.

2.2 User Class and Characteristics

There is single type of user for Course Enrollment Management System: The **Administration Department**. They can:

- Create/Read/Update/Delete students.
- Create/Read/Update/Delete enrollments.
- Create/Read/Update/Delete courses.
- View Results sorted by different attributes
- Search across data with the user keyword

The system maintains real-time updates on course enrollment management. Administrators require deeper access and control over course offerings and academic scheduling.

2.3 Design and Implementation Constraints

The design and development of the **Course Enrollment Management System** will be guided by the following constraints:

- **Use Case Diagram**

  Illustrates the interaction between users (students, faculty, administrators) and the system functionalities such as course enrollment, withdrawal, course addition, and schedule management.

- **Class Diagram**

  Defines the structure of the system by showing system classes such as Student, Course, Instructor, Enrollment, and their relationships (associations, inheritances).

- **Sequence Diagram**

  Describes the sequence of operations for major use cases like "Enroll in Course," "Drop Course," "View Course," showing interactions between the system and users over time.
- **Activity Diagram**

  Models the workflow of key activities such as the course registration process, approval workflows for special enrollments, and withdrawal from courses.
- **Component Diagram**

  Represents the physical components (like database servers, application modules, user interface modules) and how they are organized and interact within the system.
- **State Diagram**

  Depicts the states of important system objects and how they transition based on events.
- **Deployment Diagram**

  Outlines the physical deployment of software components across hardware nodes, including the server hosting the database, application server, and client machines accessing the system.

2.4 Operating Environment

- **Programming Language**: Java 17+
- **Platform**: Windows/Linux/MacOS (CLI)
- **Storage Format**: CSV (Comma-Separated Values)
- **IDE**: IntelliJ IDEA / Eclipse / VS Code
- **Execution Environment**: Java Virtual Machine (JVM)

The major features of the course management system as shown in below **Flowchart and ER diagram.**

Our **Flowchart** defines the Class Diagram and Activity Diagram. We can’t define state diagram as there are not any functions which need some secondary or tertiary allowance at any stage. The changes done are directly committed so there is no indefinite stage in-between.

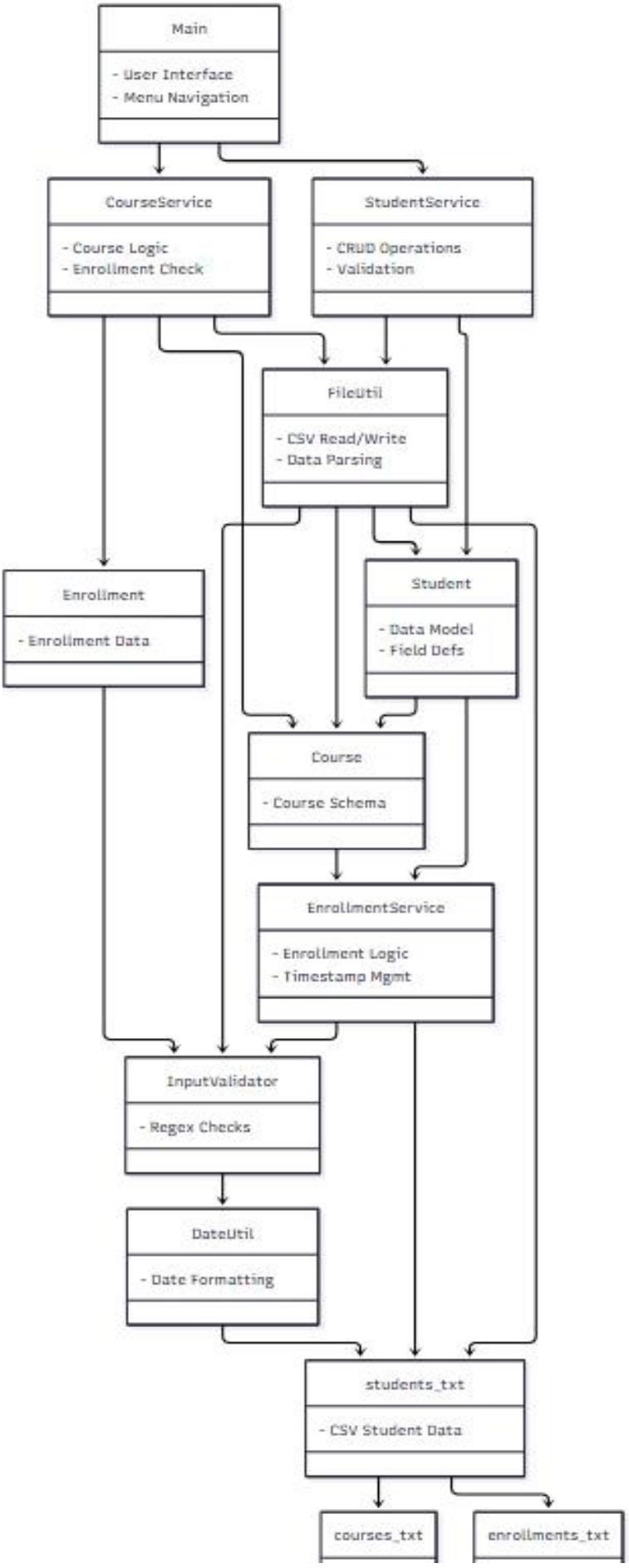

Figure 2.1: Flowchart of the App

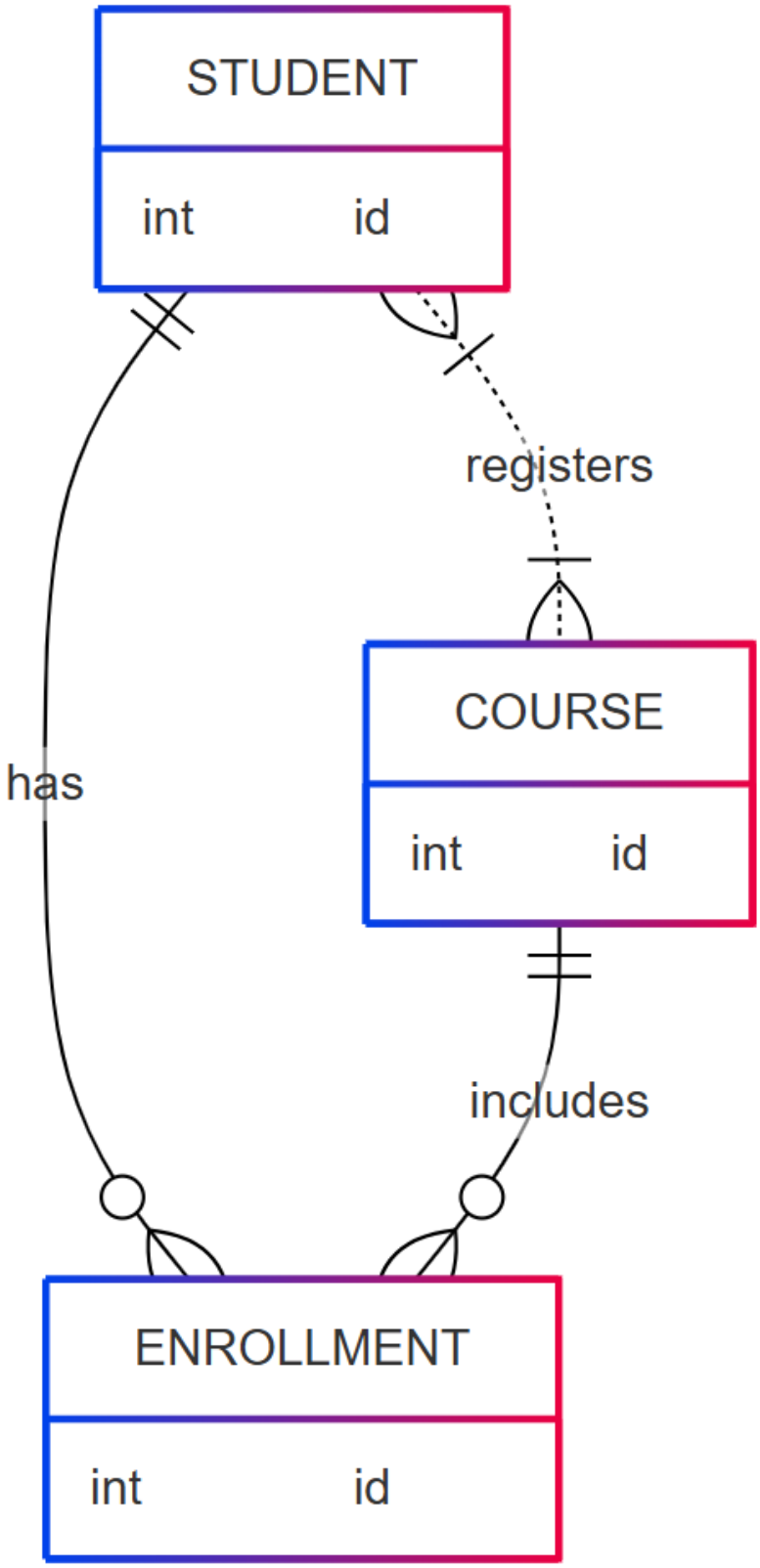

Figure 2.2 : Entity Relationship Diagram

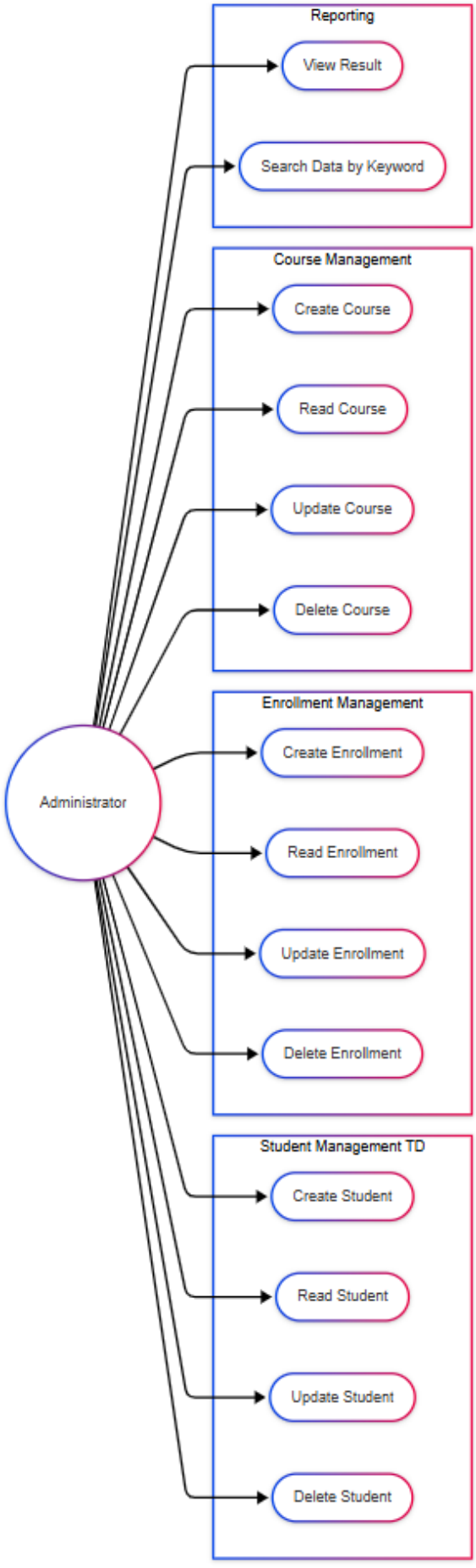

Figure 2.3: Use Case Diagram

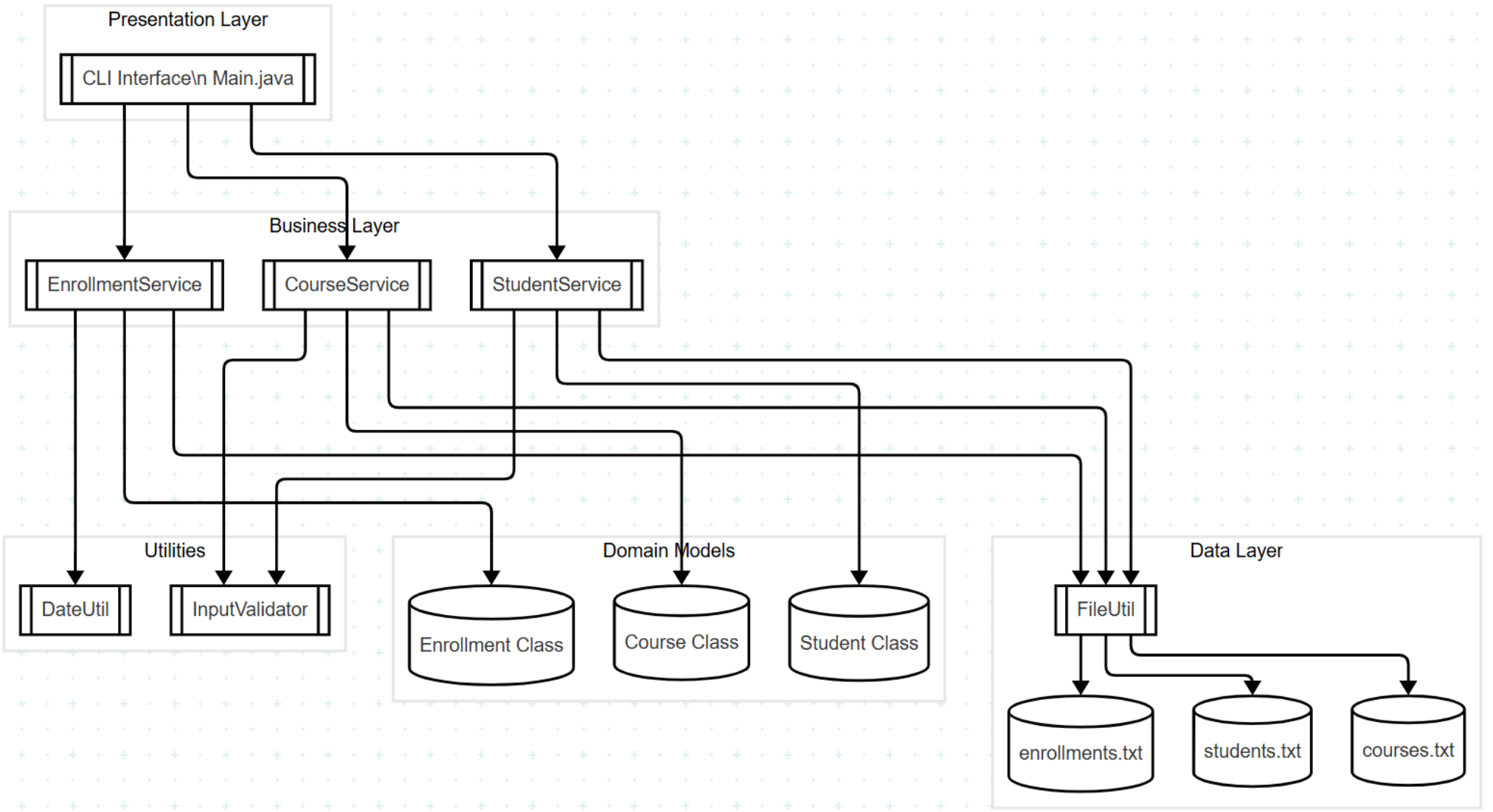

Figure 2.4: Component Diagram for the App

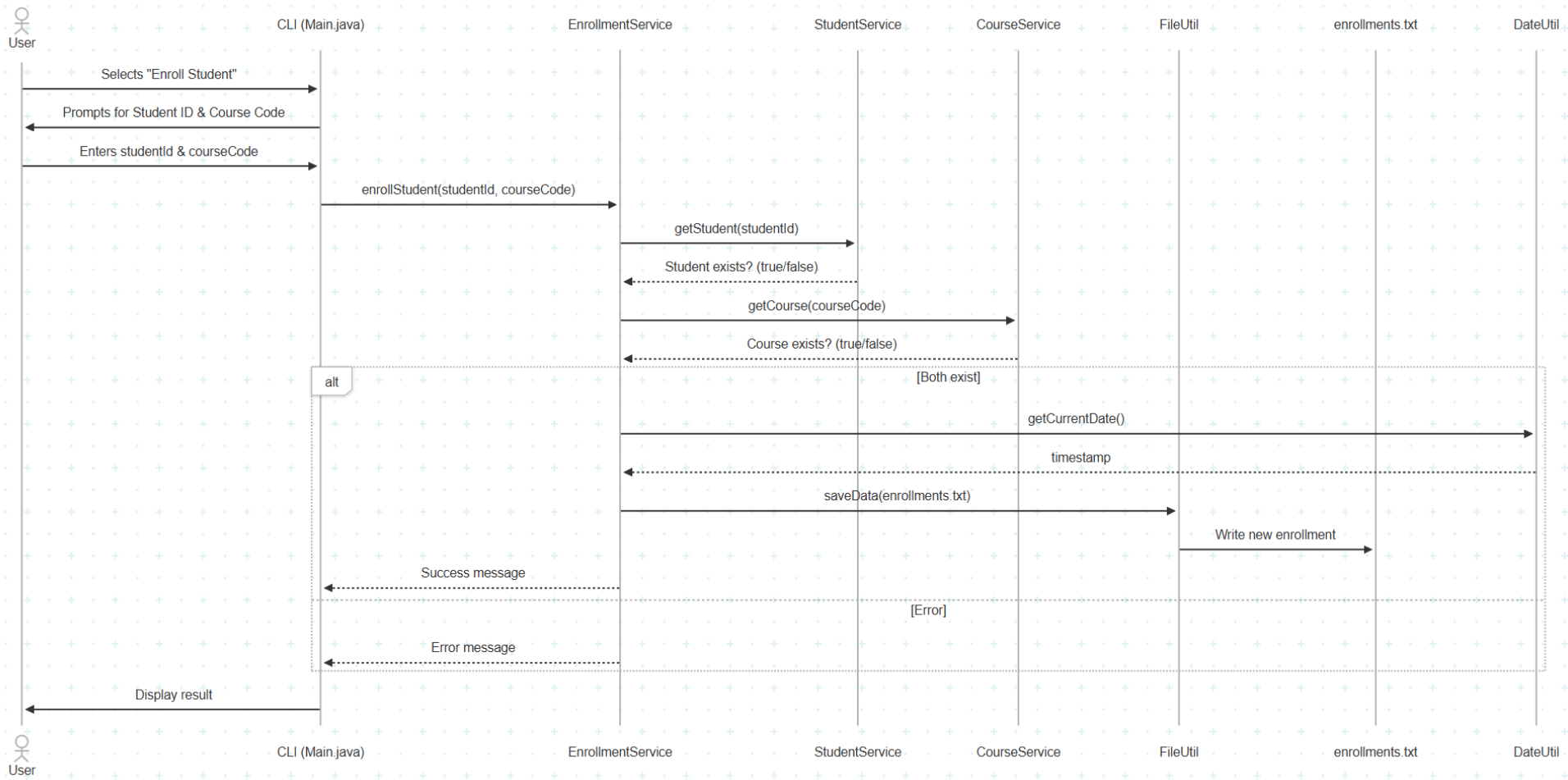

Figure 2.5: Sequence Diagram for Course Enrollment

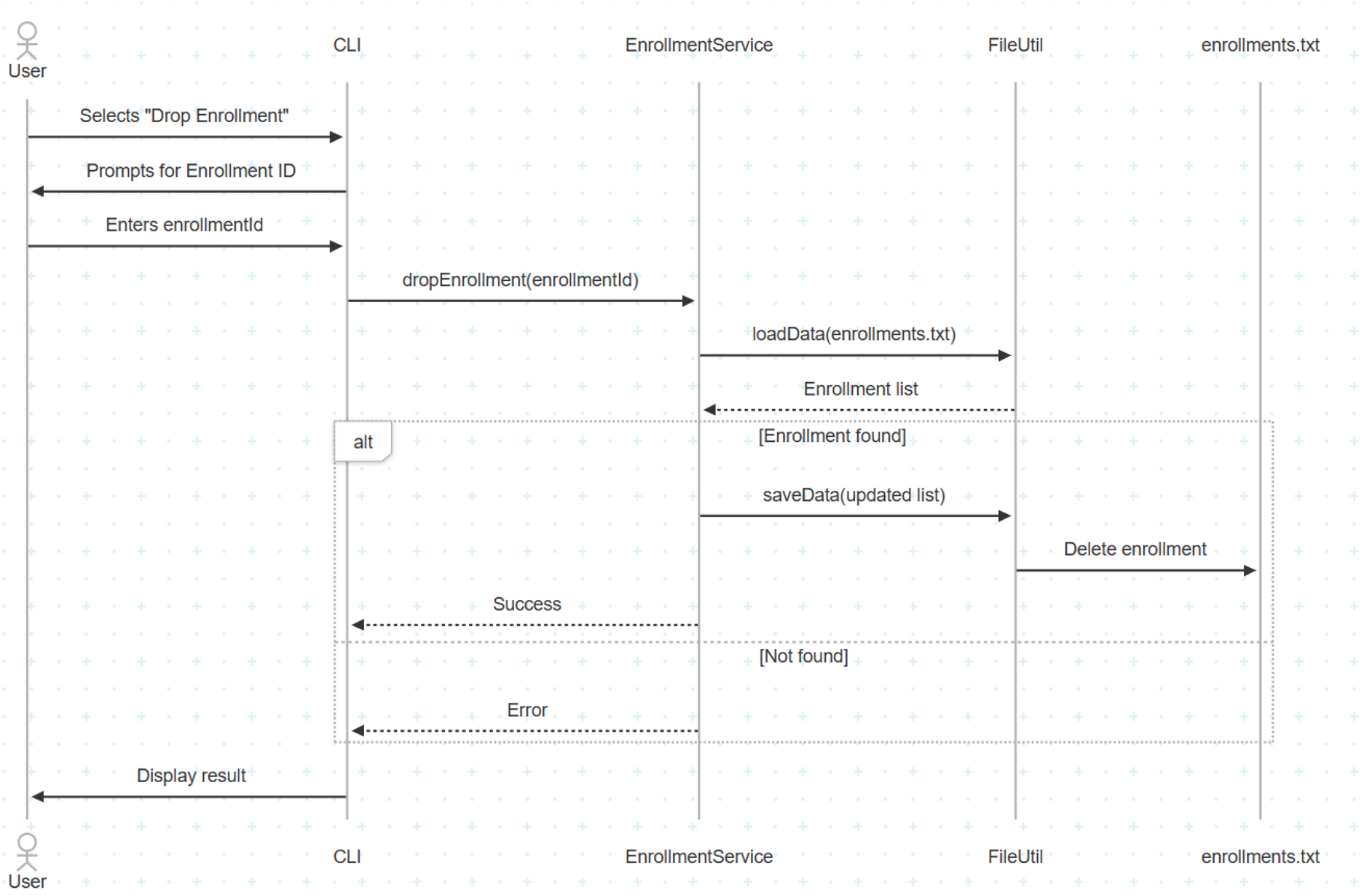

Figure 2.6: Sequence Diagram for Dropping a Course

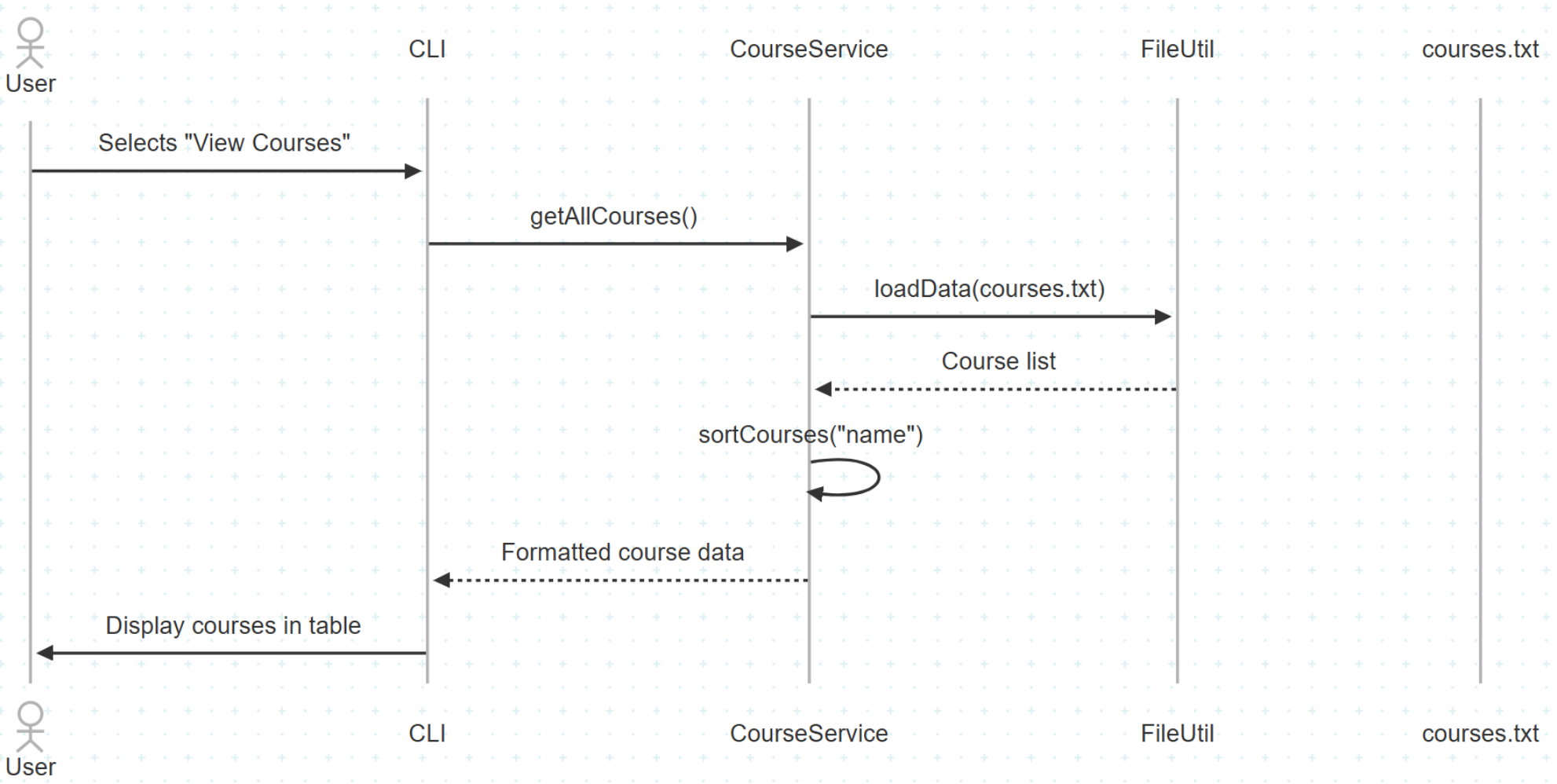

Figure 2.7: Sequence Diagram for Viewing the Course

3. System Features

3.1 Product Features and Functional Requirements

- Add, view, update, and delete student records.
- Add, view, update, and delete course information.
- Enroll students in specific courses and track enrollments.
- Persistent storage using CSV files for students, courses, and enrollment data.
- Modular and object-oriented design to promote scalability and reusability.
- User-friendly command-line interface for interacting with the system.
- Built entirely in Java, showcasing the use of classes, inheritance, exception handling, and file operations.

3.2 Description and Priority

The **Course Enrollment Management System** maintains information about courses, student enrol, class schedules, instructor assignments, and academic records. This project holds a **high priority** because efficient course management is essential for smooth academic operations and student success in any educational institution.

3.3 Stimulus/Response Sequences

- Search for available courses by department, semester, or instructor.
- Display a detailed list of available courses, allowing students to **enroll** in a selected course.
- Drop or **withdraw** from an existing course enrollment.
- View and **manage** the student's enrolled course list and schedule.
- Administrative staff can **add**, **update**, or **remove** courses.
- Instructors can **view enrolled students** and manage course materials.

3.4 Functional Requirements

- Students should be able to:
    - Search, view, and enroll in available courses.
    - Drop or withdraw from enrolled courses.
    - View their enrolled course schedule.

- Instructors should be able to:
    - View courses they are assigned to.
    - View the list of students enrolled in their courses.
- Administrators should be able to:
    - Add new courses and assign instructors.
    - Update course information (schedule, capacity, prerequisites).
    - Remove or archive courses no longer offered.

3.5 Nonfunctional Requirements

- **Performance Requirements**: The system should efficiently handle course and student data, even with increasing entries, ensuring quick file access and response time in the terminal.
- **Availability**: The system should be available anytime it is run on a compatible terminal, without relying on external servers or internet connectivity.
- **Correctness**: The application must accurately store and retrieve course and student information, ensuring correct enrollment data and preventing duplicate or invalid entries.
- **Maintainability**: The system should be easy to update, allowing future enhancements such as improved data validation or added features with minimal changes to the existing code.
- **Usability**: The system should be simple to use via the command-line interface, with clear prompts and instructions to guide users through different operations like adding students, enrolling in courses, and viewing records.

4. External Interface Requirements

The Course and Student Management System runs in a **terminal-based environment** with a **text-based interface**. Users interact with the system through a menu-driven interface that allows them to perform tasks such as adding courses, registering students, and managing enrollments. The system uses **plain .txt files with comma-separated values** to store data instead of a database. This setup is lightweight, making it suitable for academic prototypes and small-scale environments.

4.1 User Interfaces

- **Front-end software**: The system uses a **command-line interface (CLI)** developed in a general-purpose programming language (such as Python, C++, or Java as per your project). There is no graphical user interface (GUI); users interact with the system via terminal prompts and menu options.

- **Back-end software**: No dedicated database system is used. Instead, the data is stored in **plain text (.txt) files** in **comma-separated value (CSV)** format, which allows simple data storage and retrieval without a database server

4.2 Hardware Interfaces

- The project is designed to run on systems with a **basic operating system** such as **Windows** or **Linux**.
- No special hardware is required. The application works efficiently on a **standard personal computer** with a keyboard and display to interact via the terminal.

4.3 Software Interfaces

Operating System	We have chosen the Windows operating system for its best support and user friendliness.
Storage	To save all three types of records, we have chosen .txt format files in which we are storing CSV data.

Table 4.1 : List of Software Interfaces

4.4 Communication Interfaces

This project is a standalone terminal-based application. It does not rely on any network or web-based communication. All data is stored and managed locally through text files without the need for internet or browser-based interfaces.

5. Test Cases

Files after data insertion through Code :

```
courseCode,courseName,credits,instructor
"123","DSA",4,"Rajendra"
"200","BEE",2,"Nitish Rajput"
"201","OOSE",4,"Susama Baghchi"
"202","DSA",6,"Aakash"
```

Figure 5.1 : Course.txt file

```
enrollmentId,studentId,courseCode,enrollmentDate
"100-123","100","123",1744867168522
"100-200","100","200",1746112913967
"101-202","101","202",1746112904748
```

Figure 5.2 : Enrollments.txt file

```
studentId,firstName,lastName,email,phoneNumber
"100","Krati","Arora","krati@gmail.com","1234567890"
"101","Prachi","Singla","prachi@gmail.com","9876543210"
"103","Yuvraj","Gupta","yuvraj@hotmail.com","6789012345"
```

Figure 5.3 : Students.txt file

6. References

- **Java Documentation** - https://docs.oracle.com/en/java/
- **Data Structures and Algorithms in Java** by Robert Lafore
- **Effective Java** by Joshua Bloch
- **Java Collections Framework** - https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html