SkillForge: UML Design Report

Alexandria University
Faculty of Engineering
Computer and Communication Engineering
Specialized Scientific Programs

Course Information

- **Course Code:** CC272
- **Course Name:** Programming II
- **Semester:** Fall 2025/2026
- **Assignment:** Lab 6 - SkillForge UML Design

Team Members

Name	Student ID
Farah Emad	9291
Mariam Elalfy	9228
Salma Wael	9736
Ganna Khaled	9241

1. Use Case Diagram

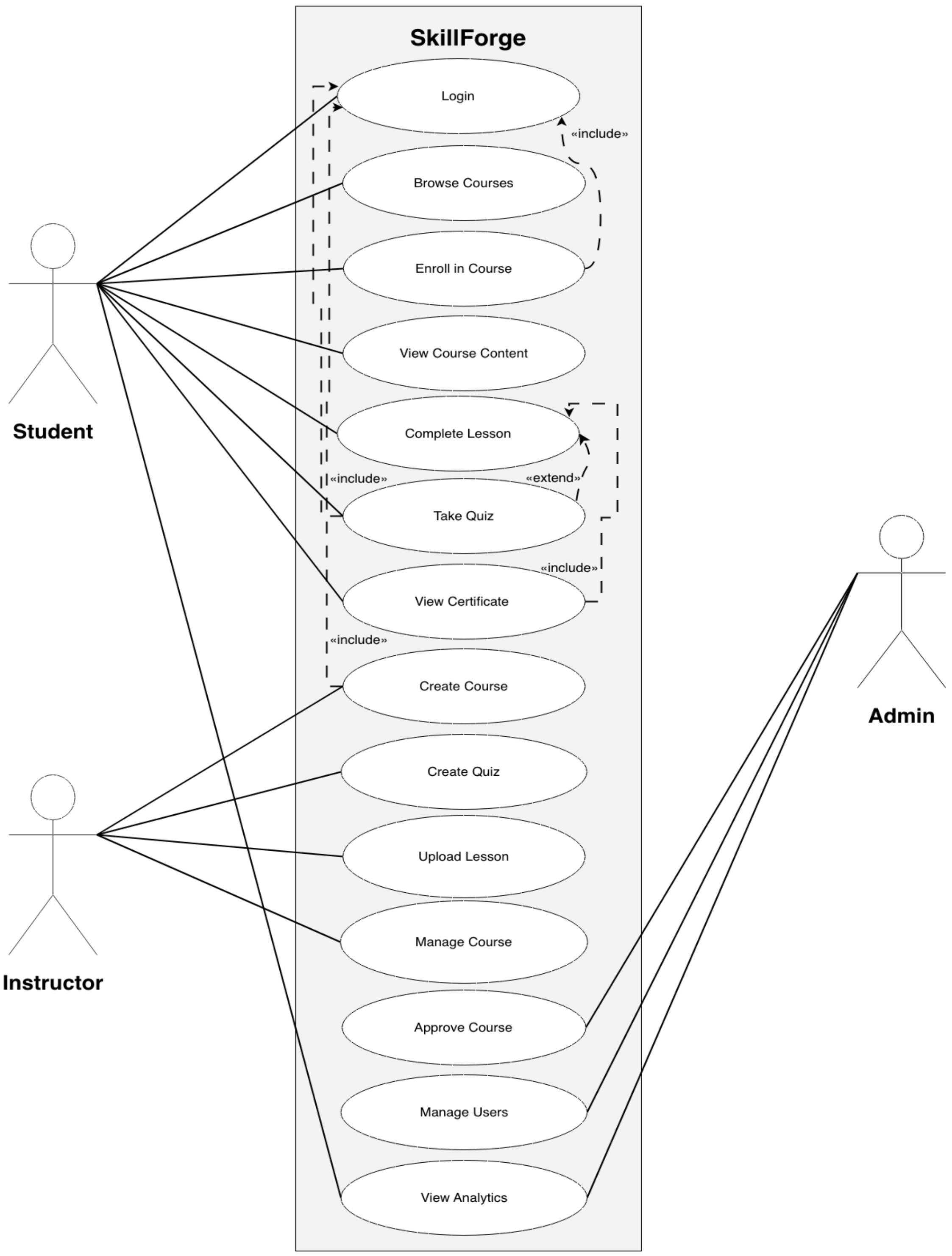

Figure 1: Use Case Diagram for SkillForge

Figure 1: Use Case Diagram for SkillForge

The Use Case Diagram identifies three main actors in the SkillForge platform: Student, Instructor, and Admin, each with distinct responsibilities and interactions with the system. Students can login, browse courses, enroll in courses, view course content, complete lessons, take quizzes, view certificates, and view analytics, with several «include» relationships showing dependencies between use cases (Login is included in Browse Courses and Enroll, Complete Lesson is included in Take Quiz and View Certificate, and Browse Courses is included in Create Course). Instructors can create courses, create quizzes, upload lessons, and manage courses, while Admins have elevated privileges to approve courses, manage users, and view platform-wide analytics. This diagram is essential for understanding the functional requirements and scope of the system by clearly defining what each user type can accomplish within the platform.

---

2. Activity Diagram

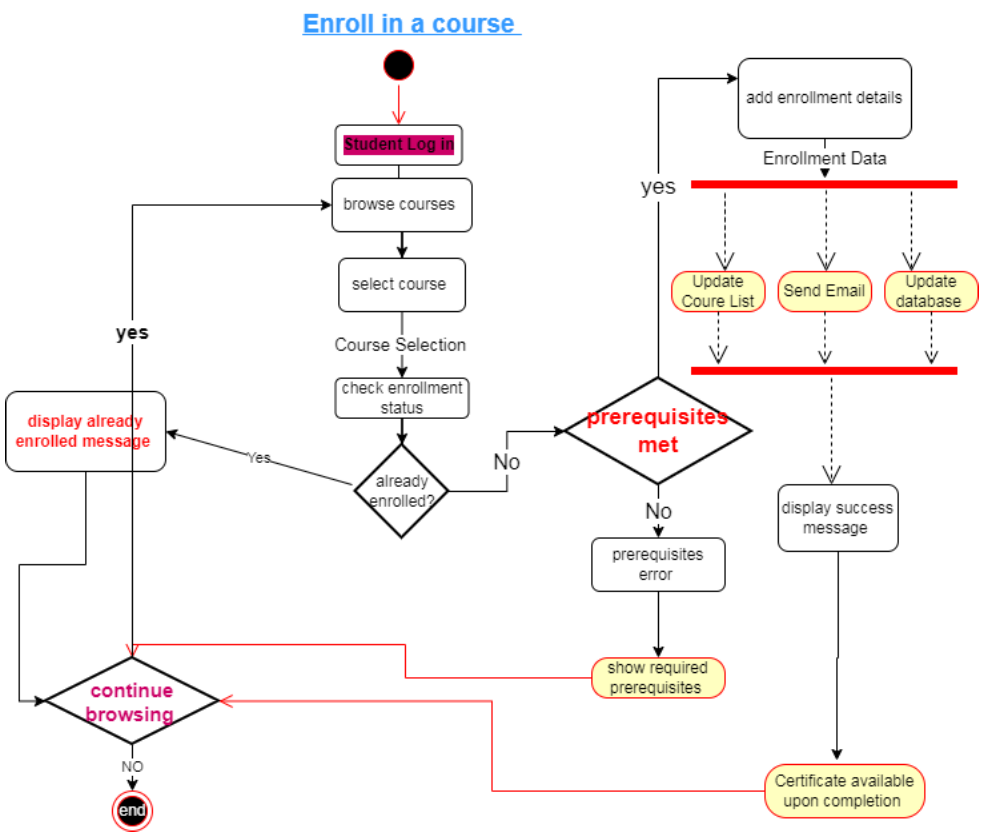

Figure 2: Activity Diagram - Enroll in a Course

The Activity Diagram models the complete workflow for enrolling in a course, beginning with student login and course browsing, followed by course selection and enrollment status validation to prevent duplicate enrollments. The diagram includes critical decision points such as checking if the student is already enrolled and verifying prerequisite requirements, with parallel activities (fork/join) showing synchronized processes for updating the course list, sending enrollment confirmation emails, and updating the database when prerequisites are met. This diagram is important for developers to understand the business logic, error handling scenarios, and the sequence of operations required to implement the enrollment feature correctly.

---

3. Class Diagram

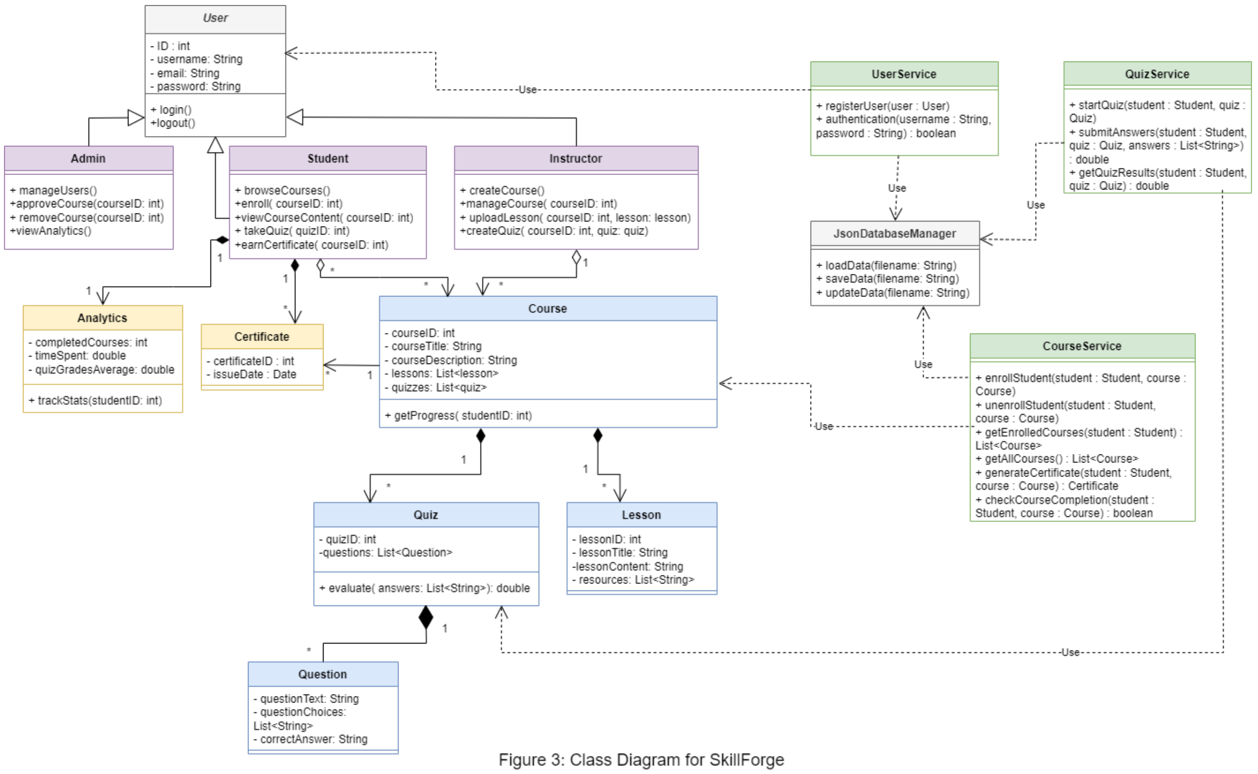
User
- ID : int
- username: String
- email: String
- password: String
+ login()
+logout()
Use
Admin
+ manageUsers()
+approveCourse(courseID: int)
+ removeCourse(courseID: int)
+viewAnalytics()
Student
+ browseCourses()
+enroll(courseID: int)
+viewCourseContent(courseID: int)
+ takeQuiz(quizID: int)
+earnCertificate(courseID: int)
Instructor
+ createCourse()
+manageCourse(courseID: int)
+ uploadLesson(courseID: int, lesson: lesson)
+createQuiz(courseID: int, quiz: quiz)
UserService
+ registerUser(user : User)
+ authentication(username : String, password : String) : boolean
QuizService
+ startQuiz(student : Student, quiz : Quiz)
+ submitAnswers(student : Student, quiz : Quiz, answers : List<String>) : double
+ getQuizResults(student : Student, quiz : Quiz) : double
Use
Use
JsonDatabaseManager
+ loadData(filename: String)
+ saveData(filename: String)
+ updateData(filename: String)
1
1
1
*
*
*
*
*
1
Analytics
- completedCourses: int
- timeSpent: double
- quizGradesAverage: double
+ trackStats(studentID: int)
Certificate
- certificateID : int
- issueDate : Date
*
1
Course
- courseID: int
- courseTitle: String
- courseDescription: String
- lessons: List<lesson>
- quizzes: List<quiz>
+ getProgress(studentID: int)
Use
CourseService
+ enrollStudent(student : Student, course : Course)
+ unenrollStudent(student : Student, course : Course)
+ getEnrolledCourses(student : Student) : List<Course>
+ getAllCourses() : List<Course>
+ generateCertificate(student : Student, course : Course) : Certificate
+ checkCourseCompletion(student : Student, course : Course) : boolean
Use
1
*
1
*
Quiz
- quizID: int
-questions: List<Question>
+ evaluate(answers: List<String>): double
Lesson
- lessonID: int
- lessonTitle: String
-lessonContent: String
- resources: List<String>
1
*
Use
Question
- questionText: String
- questionChoices: List<String>
- correctAnswer: String

Figure 3: Class Diagram for SkillForge

Figure 3: Class Diagram for SkillForge

The Class Diagram represents the static structure of SkillForge with a User base class that generalizes into Student, Instructor, and Admin subclasses, along with core domain classes including Course, Lesson, Quiz, Question, Certificate, and Analytics. The diagram shows various relationship types including inheritance (User to its subclasses), composition relationships (Course contains Lessons and Quizzes, Quiz contains Questions - all shown with filled diamonds indicating that these components cannot exist independently), and associations with proper multiplicities between Student-Course enrollment and Student-Certificate. Service layer classes (UserService, CourseService, QuizService) and JsonDatabaseManager provide separation of concerns by handling business logic and data persistence, making this diagram crucial for understanding the system architecture and guiding the implementation of classes, attributes, methods, and their relationships.

4. Sequence Diagram

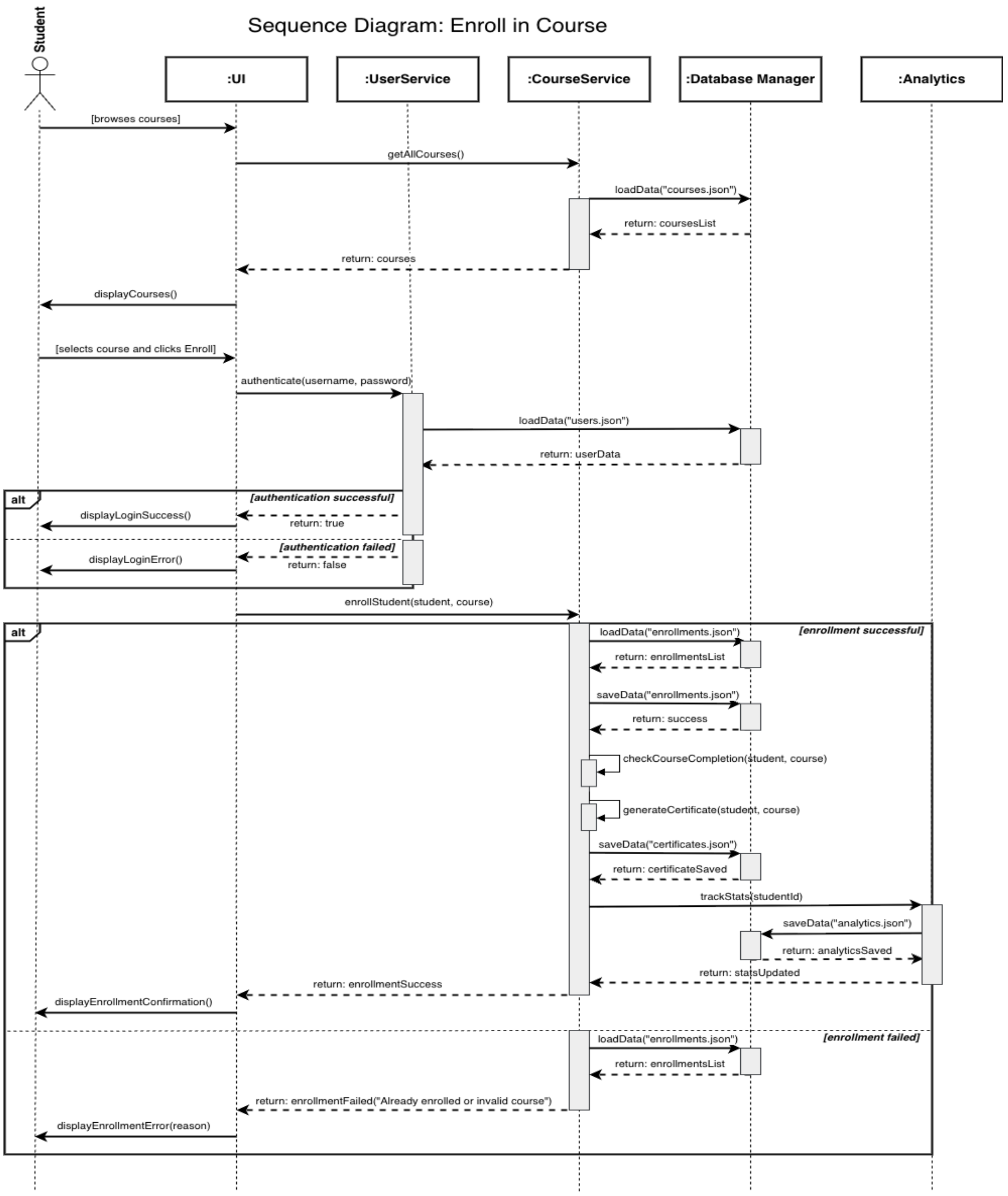
Sequence Diagram: Enroll in Course
Student
:UI
:UserService
:CourseService
:Database Manager
:Analytics
[browses courses]
getAllCourses()
loadData("courses.json")
return: coursesList
return: courses
displayCourses()
[selects course and clicks Enroll]
authenticate(username, password)
loadData("users.json")
return: userData
alt
[authentication successful]
displayLoginSuccess()
return: true
[authentication failed]
displayLoginError()
return: false
enrollStudent(student, course)
alt
loadData("enrollments.json")
[enrollment successful]
return: enrollmentsList
saveData("enrollments.json")
return: success
checkCourseCompletion(student, course)
generateCertificate(student, course)
saveData("certificates.json")
return: certificateSaved
trackStats(studentId)
saveData("analytics.json")
return: analyticsSaved
return: statsUpdated
return: enrollmentSuccess
displayEnrollmentConfirmation()
loadData("enrollments.json")
[enrollment failed]
return: enrollmentsList
return: enrollmentFailed("Already enrolled or invalid course")
displayEnrollmentError(reason)

Figure 4: Sequence Diagram - Enroll in a Course

The Sequence Diagram illustrates the dynamic interaction and message flow between system components (Student, UI, UserService, CourseService, Database Manager, and Analytics) during the course enrollment process, showing the chronological order of operations from browsing courses through authentication to final enrollment confirmation. The diagram includes alternative paths using alt fragments to handle both successful and failed scenarios for authentication and enrollment, with activation bars indicating when each object is processing and dashed return arrows showing data being passed back between components. This diagram is essential for developers to understand the exact sequence of method calls, object collaborations over time, and error handling logic required to implement the enrollment functionality correctly.

Conclusion

This UML design provides a comprehensive blueprint for implementing the SkillForge learning platform through four complementary diagrams that define functional requirements (Use Case), process workflows (Activity), static structure (Class), and dynamic behavior (Sequence). The design emphasizes proper object-oriented principles including inheritance, composition, and separation of concerns through a service layer architecture, ensuring the system is maintainable, scalable, and ready for implementation.

End of Report