Skill Forge — UML Analysis & Design
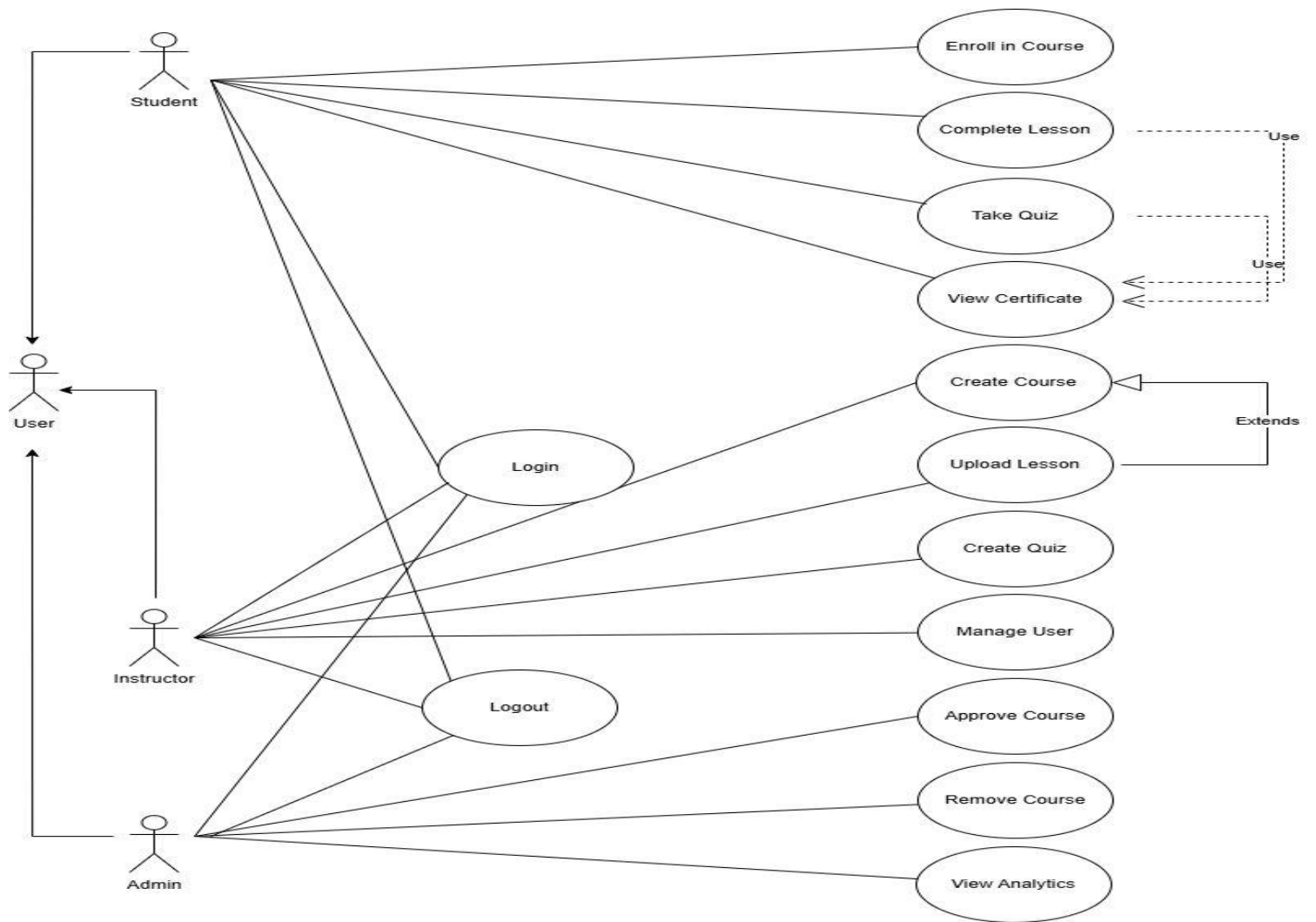
Programming II – Lab 6

Prepared by:

Mohamed Bahig – 9424

Shaden Rafik - 9641

Sara Hany Rakha - 9315

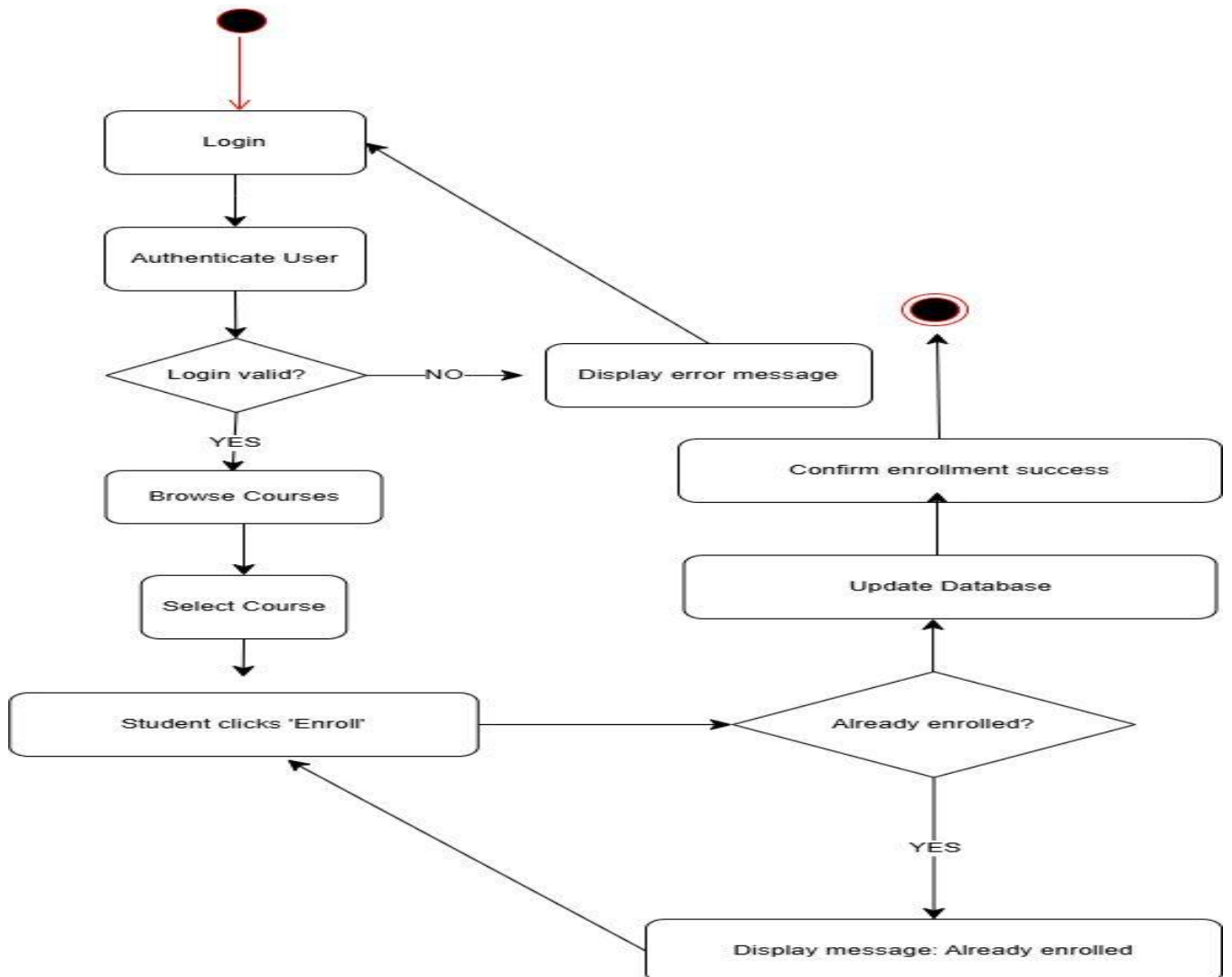Hayat Tarek - 9565

1. Use Case Diagram

- Label/Title: SkillForge Platform Use Case Diagram

This diagram maps the system's functional requirements by showing the interactions between the main actors (Student, Instructor, Admin) and the key use cases. It is essential for defining the system's scope and clearly outlining the specific tasks available to each user role.

2. Activity Diagram (Use Case: Enroll in Course)

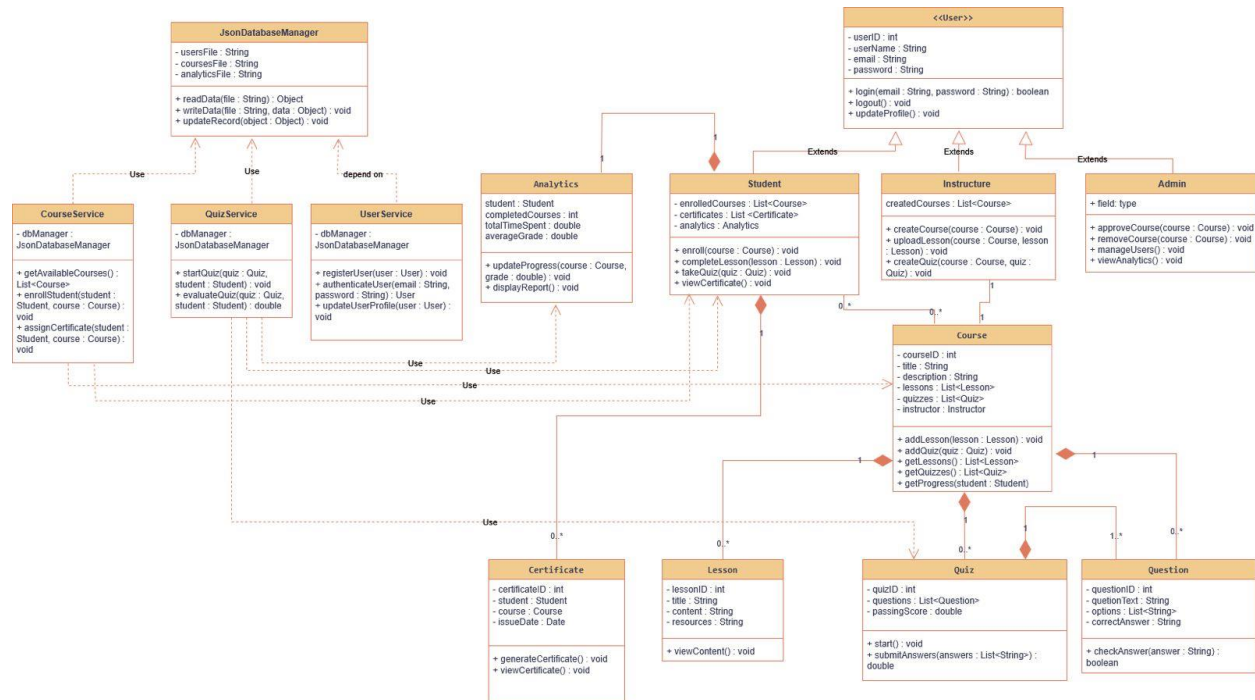• Label/Title: Activity Diagram for "Enroll in Course"

This diagram illustrates the step-by-step workflow and decision logic involved in the process of a student enrolling in a course. It is vital for understanding the sequence of actions, such as login validation and the "Already enrolled?" check.

3. Class Diagram

• Label/Title: SkillForge Platform Class Diagram

This diagram models the static structure of the SkillForge system, detailing the core classes (User, Course, Lesson, Quiz, Service classes) along with their attributes, methods, and relationships.

JsonDatabaseManager
- usersFile : String
- coursesFile : String
- analyticsFile : String

+ readData(file : String) : Object
+ writeData(file : String, data : Object) : void
+ updateRecord(object : Object) : void

<<User>>
- userID : int
- userName : String
- email : String
- password : String

+ login(email : String, password : String) : boolean
+ logout() : void
+ updateProfile() : void

Use — Use — depend on — Extends — Extends — Extends

Analytics
- student : Student
- completedCourses : int
- totalTimeSpent : double
- averageGrade : double

+ updateProgress(course : Course, grade : double) : void
+ displayReport() : void

Student
- enrolledCourses : List<Course>
- certificates : List <Certificate>
- analytics : Analytics

+ enroll(course : Course) : void
+ completeLesson(lesson : Lesson) : void
+ takeQuiz(quiz : Quiz) : void
+ viewCertificate() : void

Instructure
createdCourses : List<Course>

+ createCourse(course : Course) : void
+ uploadLesson(course : Course, lesson : Lesson) : void
+ createQuiz(course : Course, quiz : Quiz) : void

Admin
+ field: type

+ approveCourse(course : Course) : void
+ removeCourse(course : Course) : void
+ manageUsers() : void
+ viewAnalytics() : void

CourseService
- dbManager : JsonDatabaseManager

+ getAvailableCourses() : List<Course>
+ enrollStudent(student : Student, course : Course) : void
+ assignCertificate(student : Student, course : Course) : void

QuizService
- dbManager : JsonDatabaseManager

+ startQuiz(quiz : Quiz, student : Student) : void
+ evaluateQuiz(quiz : Quiz, student : Student) : double

UserService
- dbManager : JsonDatabaseManager

+ registerUser(user : User) : void
+ authenticateUser(email : String, password : String) : User
+ updateUserProfile(user : User) : void

Course
- courseID : int
- title : String
- description : String
- lessons : List<Lesson>
- quizzes : List<Quiz>
- instructor : Instructor

+ addLesson(lesson : Lesson) : void
+ addQuiz(quiz : Quiz) : void
+ getLessons() : List<Lesson>
+ getQuizzes() : List<Quiz>
+ getProgress(student : Student)

Use — Use — Use — Use

0..* — 0..* — 0..* — 1..* — 0..*

Certificate
- certificateID : int
- student : Student
- course : Course
- issueDate : Date

+ generateCertificate() : void
+ viewCertificate() : void

Lesson
- lessonID : int
- title : String
- content : String
- resources : String

+ viewContent() : void

Quiz
- quizID : int
- questions : List<Question>
- passingScore : double

+ start() : void
+ submitAnswers(answers : List<String>) : double

Question
- questionID : int
- quetionText : String
- options : List<String>
- correctAnswer : String

+ checkAnswer(answer : String) : boolean

4. Sequence Diagram (Use Case: Enroll in Course)

• Label/Title: Sequence Diagram for "Enroll in Course"

This diagram shows the dynamic collaboration among objects over time for the Enroll in Course use case. It details the message flow and the exact order of method calls (e.g., getAvailableCourses(), enrollStudent()) between system components like the Student object, CourseService, and JSONDatabaseManager.