Project: Requirements analysis and software design

Assessment 1 – Part 1: Requirements Analysis

Course: 32555 Fundamentals of Software Development - Spring 2025

Lab number: Cmp1-07

Tutor name: Dr. Hasan Ali Khattak

Student ID	Student Name
25241807	Nichakamon Ek-Aksorn

Contents

Project Overview of University Application (UniApp)

This project aims to design a university enrolment system that allows students to self-register, log in, and enrol in up to four subjects per semester, with each subject automatically assigned a unique ID, mark, and grade. The system will also save student data for later access and provide administrators with functions to view, categorise, and manage student records.

The objective of Task 1 is to analyse these requirements and present them in a structured form. This includes developing user stories and mapping them into a backlog, creating a UML use case diagram to show actors and their interactions with the system, and designing a UML class diagram to identify the classes, attributes, methods, and relationships. Together, these outputs provide the foundation for later implementation of the system.

Task 1 User-Story Backlog

The following backlog table lists the user stories with unique IDs, actions, results, and their related functions from the class diagram.

ID	User	Action	Result	Function
101	Student	I want to register with my name, email, and password	So that I can enrol	`registerStudent(fName: String, lName: String, email: String, password: String): Student`
102	Student	I want to log in	So that I can access my account	`studentLogin(email: String, password: String): Boolean`
103	Student	I want to enrol in a subject	So that I can study up to 4 subjects	`enrolSubject(): Subject`
104	Student	I want to remove a subject	So that I can adjust my enrolments	`removeSubject(subjectId: String): Boolean`
105	Student	I want to view my enrolment list	So that I can know what subjects I enrolled in	`viewEnrolments(: List<Subject>`
106	Student	I want to change my password	So that I can keep my account secure	`changePassword(oldPass: String, newPass: String): Boolean`
107	Student	I want the system to assign a random mark and calculate the grade for my subject	So that my subject has a mark and grade	`assignRandomMark(): Integer` `calculateGrade(): String`

108	Student	I want the system to validate my email format	So that I use only university emails	`validateEmail(email: String): Boolean`
109	Student	I want the system to validate my password format	So that my password meets security rules	`validatePassword(password: String): Boolean`
110	Student	I want my data registration and enrolment to be saved in a file	So that I don't lose any information	`saveData(): Void` `loadData(): Void`
201	Admin	I want to log in with my admin ID and password	So that I can log in the admin system	`adminLogin(adminId: String, password: String): Boolean`
202	Admin	I want to view all registered students	So that I can manage them	`viewAllStudents(): List<Student>`
203	Admin	I want to view students by grade	So that I can view students by performance	`viewStudentsByGrade(grade: String): List<Student>`
204	Admin	I want to categorize students into PASS/FAIL	So that I can identify failing students	`categorizePassFail(): Map<String, List<Student>>`
205	Admin	I want to remove a student from the list	So that I can delete them from the system	`removeStudent(studentId: String): Boolean`
206	Admin	I want to clear all student data	So that I can reset the system	`clearAllStudents(): Void`

Task 2.1 Use Case Diagram

The use case diagram illustrates the interactions between students, administrators, and the enrolment system.

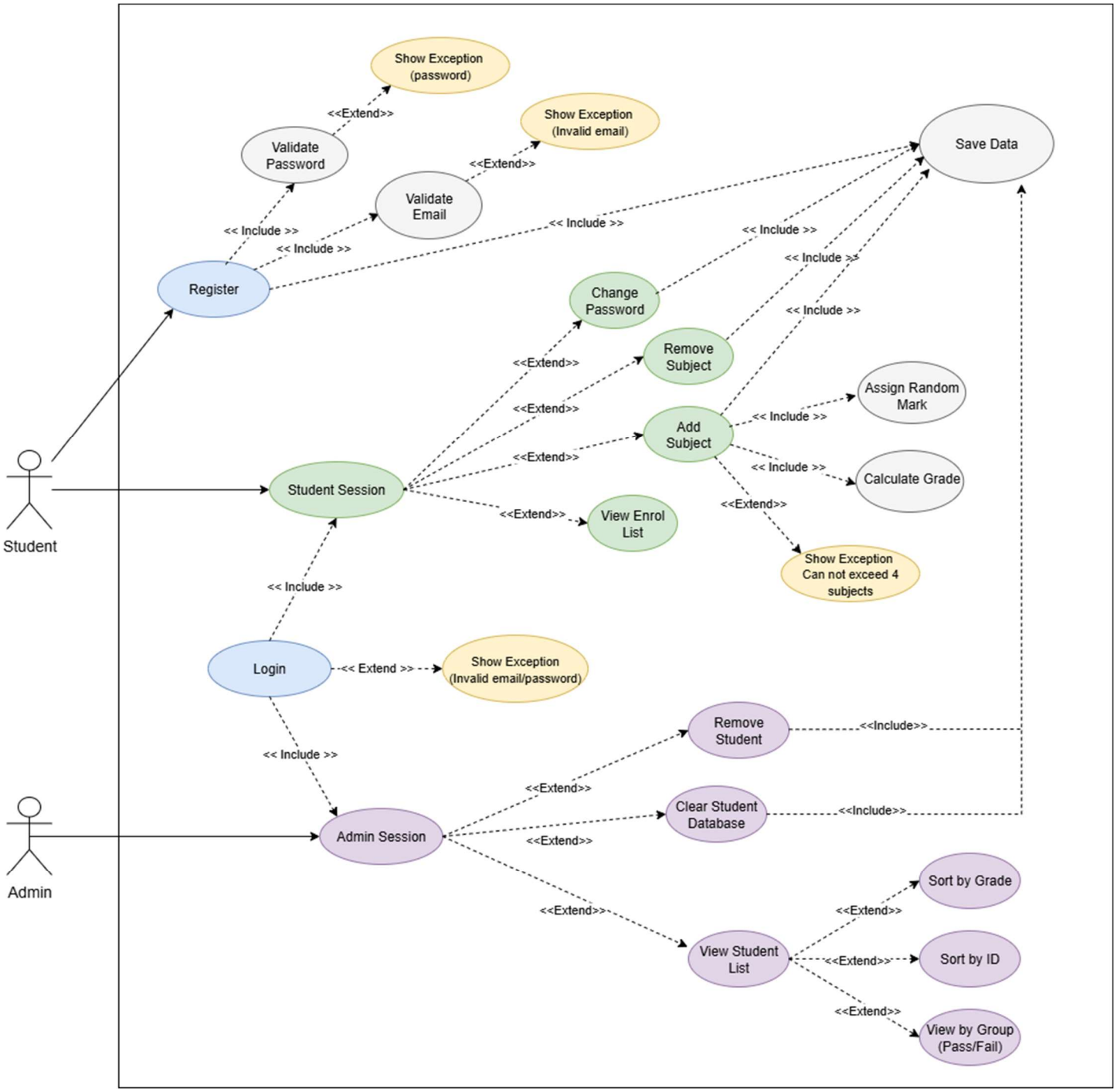

Figure 1: Use Case Diagram of UniApp

Task 2.2 Use Case Diagram Explanation

1. Actors

- **Student:**

  Represents a university student who interacts with the system to register, log in, enrol in subjects, manage their enrolments, and change their password.

- **Admin:**

  Represents a university staff member who accesses the system to manage student data by viewing, categorising, removing, or clearing records.

2. Student Use Cases

- **Register:** A student creates a new account by entering their name, email, and password.
  - Includes **Validate Email** (system checks email format to ensure it ends with @university.com).
  - Includes **Validate Password** (system checks password starts with uppercase, contains ≥5 letters, and ends with ≥3 digits).
  - Includes **Save Data** (new student record stored in students.data).
- **Login:** A student provides valid email and password to access the system.
  - Extends to **Show Exception (Invalid email/password)** if credentials do not match stored records.
- **Student Session**: Represents the student's interaction after successful login, grouping their available actions:
  - **Enrol Subject**: Adds a new subject to the student's enrolment list.
    - Includes **Assign Random Mark** (system generates a mark between 25 - 100).

        - Includes **Calculate Grade** (system converts mark into grade Z, P, C, D, or HD).
        - Includes **Save Data** (updates stored enrolments).
        - Extends to **Show Exception** (Cannot exceed 4 subjects) if student tries to enrol more than 4 subjects.
    - **Remove Subject**: Deletes a subject from the enrolment list. Includes Save Data to update file.
    - **View Enrol List**: Displays the student's current enrolled subjects with marks and grades.
    - **Change Password**: Allows the student to update their password. Includes Save Data.

3. Admin Use Cases

- **Admin Session**: Represents the admin's interaction with the system.
    - **View Student List**: Displays all registered students and their details.
        - Can be organised by **Student ID**, **Grade**, or **Pass/Fail** category.
    - **Remove Student**: Deletes an individual student's record. *Includes* Save Data.
    - **Clear Student Database**: Removes all student records from the file. *Includes* Save Data.

Task 3 Class Diagram

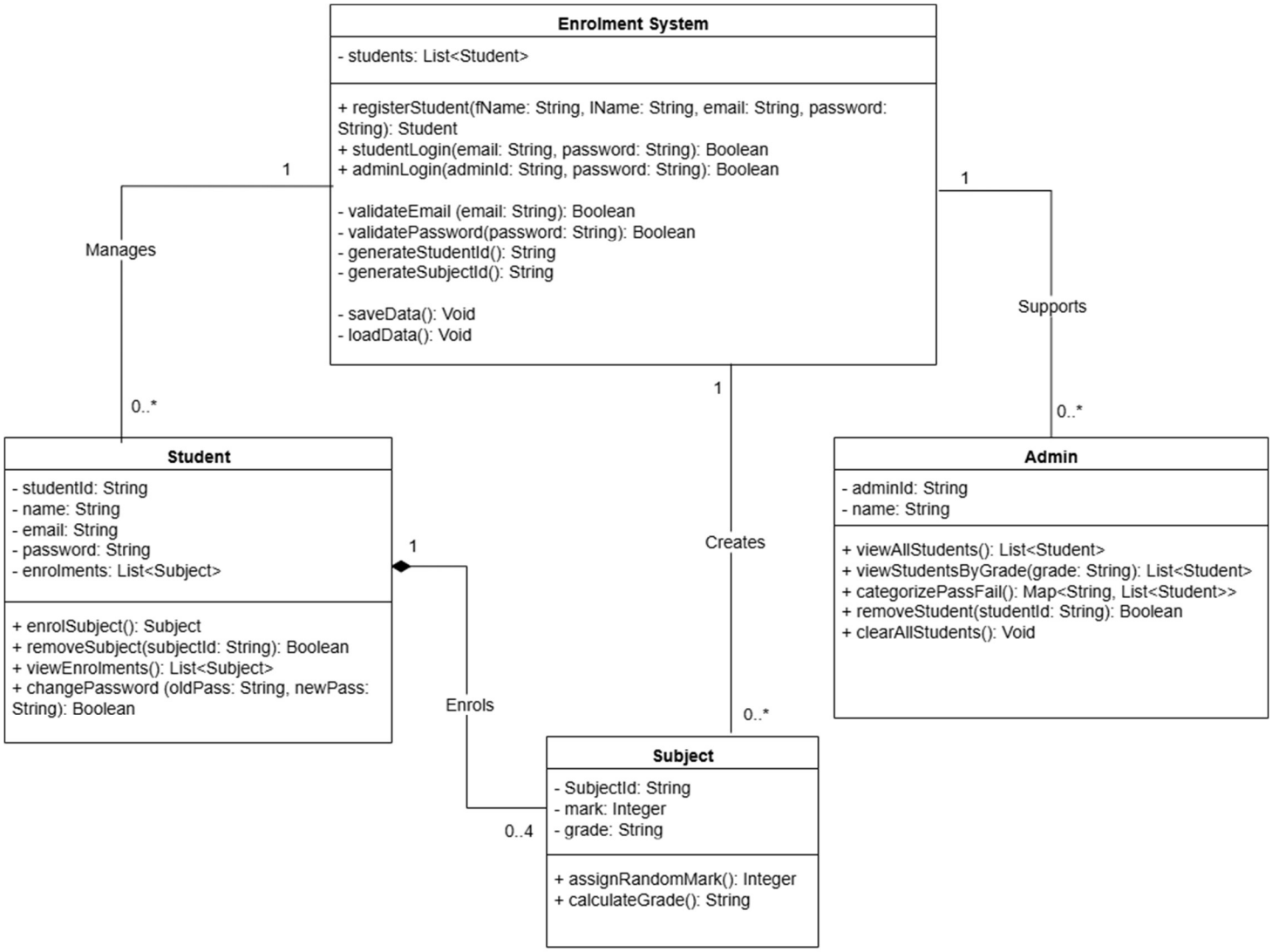

Figure 2: Class Diagram of UniApp