EduReg: A Centralized Student Registration System

Phase 3: System Analysis

By:

Olira Tesgera (1093915)

Yisak Demelash (1093891)

Instructor: Dr. Murad Al-Rajab

Abu Dhabi, United Arab Emirates

November 2, 2025

Table of Contents

1. Activity Diagrams

1.1 Annual Registration and Fee Verification Process

1.2 Assignment Submission and Grading Workflow

1.3 Leave Application and Approval Process

1.4 Class and Section Assignment Process

1.5 Fee Payment and Verification Workflow

2. Class Diagram

2.1 Detailed Class Diagram

2.2 Class Descriptions

3. Sequence Diagrams

3.1 Student Submits an Assignment

3.2 Teacher Grades an Assignment and Updates Gradebook

3.3 Administrator Manages Annual Registration

3.4 Administrator Verifies Fee Payment

3.5 Teacher Takes Daily Attendance

---

\*\*1. Activity Diagrams\*\*

Activity diagrams model the dynamic aspects of the system, illustrating the flow of control from one activity to another. Below are the key workflows for the EduReg system, covering all major functionalities identified in the requirements.

\*\*1.1 Annual Registration and Fee Verification Process\*\*

This diagram models the complete, end-to-end process for a new or returning student to register for an academic year, incorporating the critical external fee payment step.

Workflow Description:

1. The process begins with the Administrator (Director) initiating the annual registration for a user.

2. A decision point checks if the user is New or Existing.

3. If New, the Administrator creates a new user account. If Existing, the Administrator initiates the re-registration process.

4. The system Sends Login Credentials to the user (Student/Parent).

5. The user logs in and Completes/Updates the Online Registration Form, including providing their Fayda ID.

6. The user Uploads Required Documents (transcripts, ID, photo).

7. The user Submits the Completed Application.

8. The Administrator Reviews the Application and Verifies Documents.

9. If documents are rejected, the user is notified and must resubmit.

10. If documents are approved, the system notifies the user to Make Fee Payment via External Service (Telebirr/Bank).

11. The user makes the payment externally and then Uploads the Payment Receipt into the EduReg system.

12. The Administrator Verifies the Payment Receipt against external financial records.

13. If the payment is invalid, the user is notified and must upload a correct receipt.

14. If the payment is verified, the Administrator Approves the Final Registration.

15. The system updates the user's status to Active and Enrolled, and the process ends.

\*\*1.2 Assignment Submission and Grading Workflow\*\*

This diagram illustrates the complete lifecycle of an assignment, from creation to final grading and feedback.

Workflow Description:

1. The workflow is initiated by the Teacher who Creates an Assignment, setting a title, description, due date, and attaching files.

2. The system publishes the assignment, making it visible on the Student Dashboard.

3. The Student Views the Assignment details.

4. The Student Works on the Assignment and Uploads Submission File(s).

5. The system marks the assignment as "Submitted" (or "Late" if past the due date) and records the submission timestamp.

6. The Teacher Reviews the Submitted Work.

7. The Teacher Enters a Grade and Feedback into the system.

8. The system Updates the Student's Gradebook record for that subject.

9. The system Notifies the Student of the new grade and feedback.

10. The Student Views the Grade and Feedback in their portal.

11. The workflow concludes.

\*\*1.3 Leave Application and Approval Process\*\*

This diagram shows the steps involved when any user (Student, Teacher, Staff) applies for leave and how it is processed.

Workflow Description:

1. The process starts when a User (Student, Teacher, or Staff) decides to Apply for Leave.

2. The user fills out the online leave application form, providing start date, end date, and reason.

3. The user Submits the Leave Request.

4. The system routes the application to the designated Approver (based on policy: Director or a delegated Teacher).

5. The Approver Reviews the Leave Application and supporting documents.

6. A decision is made to Approve or Reject the request.

7. If Rejected, the system updates the request status and sends a rejection notification to the user, stating the reason. The process ends.

8. If Approved, the system Updates the User's Attendance Record to reflect the authorized leave.

9. The system then sends an approval notification to the user.

10. The process ends.

\*\*1.4 Class and Section Assignment Process\*\*

This diagram details the administrative process of assigning approved students to class sections.

Workflow Description:

1. The process begins after a student's registration is Fully Approved (documents and fees verified).

2. The Administrator accesses the Class Assignment Module.

3. The Administrator Selects a Grade Level to view all approved, unassigned students.

4. The system displays the list of students and available Sections with their current occupancy.

5. The Administrator chooses an assignment method: Manual, Random, or Balanced (by gender/performance).

6. The system Automatically Suggests Assignments if a non-manual method is chosen.

7. The Administrator Reviews and Confirms the assignments.

8. For each student, the system Updates their Record with the assigned section.

9. The system Generates the Student's Timetable based on the section's schedule.

10. The system Notifies the Student/Parent that the assignment is complete and the timetable is available to view.

11. The process ends.

\*\*1.5 Fee Payment and Verification Workflow\*\*

This diagram focuses specifically on the external payment and internal verification process, a critical feature of EduReg.

Workflow Description:

1. The Parent/Student receives a notification to pay fees after document verification.

2. The user makes a payment using an External Service (Telebirr, Bank).

3. The user then logs into EduReg and navigates to the Fee Payment Section.

4. The user Uploads a Picture/Scan of the Payment Receipt or enters a Transaction ID.

5. The system marks the fee status as "Pending Verification" and notifies the Administrator.

6. The Administrator views the list of pending payments.

7. The Administrator Checks the Receipt/Transaction ID against the school's external financial records.

8. A decision is made: is the payment Valid?

9. If Invalid, the Administrator rejects the payment, provides a reason, and the user is notified to correct it.

10. If Valid, the Administrator marks the payment as "Verified".

11. The system updates the student's financial status and triggers the next step in registration (e.g., becoming eligible for class assignment).

12. The user is Notified of Successful Verification.

---

\*\*2. Class Diagram\*\*

The class diagram models the static structure of the system, showing the system's classes, their attributes, operations (methods), and the relationships between them.

\*\*2.1 Detailed Class Diagram\*\*

[A visual UML diagram would be inserted here showing all classes, attributes, methods, and relationships with multiplicities]

\*\*2.2 Class Descriptions\*\*

\*\*User\*\* (Abstract Base Class)

- Attributes: userID, faydaID, name, email, passwordHash, role, isActive

- Operations: login(), logout(), resetPassword()

- Description: Base class for all system users with common attributes and authentication methods.

\*\*Student\*\* (Inherits from User)

- Attributes: studentID, gradeLevel

- Operations: submitAssignment(), viewTimetable(), applyForLeave()

- Relationships: Associated with one Guardian, enrolled in one Section, has multiple Submissions and Grades

\*\*Teacher\*\* (Inherits from User)

- Attributes: teacherID, specialization

- Operations: createAssignment(), enterGrade(), takeAttendance(), requestResource()

- Relationships: Teaches multiple Subjects, head of one Section, creates multiple Assignments

\*\*Administrator\*\* (Inherits from User)

- Attributes: adminID

- Operations: manageUserAccounts(), initiateRegistration(), verifyPayment(), assignToSection(), generateReports()

- Relationships: Manages all system entities and processes

\*\*Guardian\*\*

- Attributes: guardianID, phoneNumber, relationship

- Relationships: Responsible for one or more Students

\*\*Section\*\*

- Attributes: sectionID, sectionName, academicYear, maxCapacity

- Relationships: Contains multiple Students, has one head Teacher, has scheduled Subjects

\*\*Subject\*\*

- Attributes: subjectID, subjectCode, subjectName

- Relationships: Taught by multiple Teachers

\*\*Assignment\*\*

- Attributes: assignmentID, title, description, dueDate, maxScore

- Operations: updateDetails()

- Relationships: Created by one Teacher, has multiple Submissions

\*\*Submission\*\*

- Attributes: submissionID, submissionDate, filePaths, status

- Operations: uploadFile()

- Relationships: For one Assignment, by one Student

\*\*Grade\*\*

- Attributes: gradeID, score, feedback, gradedDate

- Relationships: For one Submission, for one Student, in one Subject

\*\*AttendanceRecord\*\*

- Attributes: recordID, date, status, remarks

- Relationships: For one Student, recorded by one Teacher

\*\*LeaveRequest\*\*

- Attributes: requestID, startDate, endDate, reason, status

- Operations: submit(), cancel()

- Relationships: Submitted by one User, approved by one User (Administrator/Teacher)

\*\*FeePayment\*\*

- Attributes: paymentID, amount, paymentDate, method, receiptImagePath, status

- Relationships: For one Student, verified by one Administrator

\*\*Boundary Classes (UI):\*\* LoginPage, StudentDashboard, TeacherDashboard, AdminDashboard, RegistrationFormPage, AssignmentPage, GradebookPage

\*\*Control Classes (Logic):\*\* AuthenticationManager, RegistrationManager, AssignmentManager, GradeManager, AttendanceManager, PaymentManager

---

\*\*3. Sequence Diagrams\*\*

Sequence diagrams model the interactions between objects over time, showing how system components collaborate to execute specific use cases.

\*\*3.1 Student Submits an Assignment\*\*

Sequence of Events:

1. Student clicks submit button on AssignmentPage

2. AssignmentPage calls submit(assignmentID, studentID, fileData) on AssignmentManager

3. AssignmentManager creates new Submission object

4. AssignmentManager calls save() on Submission object

5. Submission object confirms save operation

6. AssignmentManager calls logActivity() on ActivityLogger

7. AssignmentManager calls sendNotification() on NotificationManager

8. NotificationManager sends notification to Teacher

9. Success confirmation returned to AssignmentPage

10. AssignmentPage displays "Submission Successful" to Student

\*\*3.2 Teacher Grades an Assignment and Updates Gradebook\*\*

Sequence of Events:

1. Teacher enters score and feedback on GradebookPage and clicks "Save Grade"

2. GradebookPage calls enterGrade(submissionID, score, feedback) on GradeManager

3. GradeManager retrieves corresponding Submission object

4. GradeManager creates new Grade object

5. GradeManager calls save() on Grade object

6. GradeManager calls updateGradebook() on Student object

7. GradeManager calls sendNotification() on NotificationManager

8. NotificationManager sends notification to Student

9. Success message returned to GradebookPage

10. GradebookPage displays "Grade Saved" to Teacher

\*\*3.3 Administrator Manages Annual Registration\*\*

Sequence of Events:

1. Administrator selects "Initiate Re-registration" on AdminDashboard

2. AdminDashboard calls initiateReRegistration(academicYear, gradeLevel) on RegistrationManager

3. RegistrationManager calls findStudentsByGrade(gradeLevel) on Student class

4. List of Student objects returned

5. RegistrationManager calls setStatus("PENDING\_RE\_REGISTRATION") on each Student object

6. Each Student object confirms status update

7. RegistrationManager calls sendBulkNotification() on NotificationManager

8. NotificationManager sends notifications to all Parents/Students

9. Confirmation message returned to AdminDashboard

10. AdminDashboard displays "Re-registration Initiated"

\*\*3.4 Administrator Verifies Fee Payment\*\*

Sequence of Events:

1. Administrator selects pending payment on AdminDashboard

2. AdminDashboard calls getPaymentDetails(paymentID) on PaymentManager

3. PaymentManager retrieves FeePayment and Student objects

4. Payment and student details returned to AdminDashboard

5. Administrator clicks "Verify" after external validation

6. AdminDashboard calls verifyPayment(paymentID, adminID) on PaymentManager

7. PaymentManager updates FeePayment status to "VERIFIED"

8. PaymentManager calls updateRegistrationStatus() on Student object

9. PaymentManager calls sendNotification() on NotificationManager

10. Success result returned to AdminDashboard

\*\*3.5 Teacher Takes Daily Attendance\*\*

Sequence of Events:

1. Teacher opens AttendancePage for specific section and date

2. AttendancePage calls getStudents(sectionID) on AttendanceManager

3. List of Student objects returned

4. Teacher marks attendance and clicks "Save"

5. AttendancePage calls recordAttendance(sectionID, date, attendanceMap) on AttendanceManager

6. AttendanceManager creates and saves AttendanceRecord for each student

7. AttendanceManager checks absence limits against AttendancePolicy

8. For students exceeding limits, AttendanceManager calls sendNotification() on NotificationManager

9. Confirmation returned to AttendancePage

10. Teacher sees "Attendance Saved" message

---

\*\*End of Phase 3 Document\*\*

---