**Computer Science Department | 03/10/2025**

**Software Development Document (SDD)**

**Project Title:** Adeyemi Federal University of Education, Ondo (AFUED) School Portal & Result Processing System  
**Prepared By:** Computer Sci Department- AFUED  
**Date:** October 3, 2025

**1. Introduction**

**1.1 Purpose**

The purpose of this project is to design and develop a web-based school portal for Adeyemi Federal University of Education, Ondo. The system will primarily focus on result processing, while also providing features for students, staff, and administrators.

**1.2 Scope**

The system will:

* Allow students to log in, view personal details, register for courses, and check results.
* Enable lecturers to upload, edit, and manage student scores.
* Provide administrative staff with tools for result computation, approval, and transcript generation.
* Offer secure authentication and authorization.
* Maintain records in MongoDB for scalability and flexibility.
* Allows parent to access their wards’ results.

**1.3 Technologies**

* **Frontend:** HTML5, CSS3, JavaScript (React optional)
* **Backend:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** JWT (JSON Web Tokens)
* **Hosting:** Could be deployed on AWS, DigitalOcean, or school servers

**2. System Requirements**

**2.1 Functional Requirements**

* **Authentication & Authorization:**
  + Staff and Student login with role-based access.
* **Student Module:**
  + View personal info, register courses, check results.
* **Lecturer Module:**
  + Upload student scores, manage course results.
* **Admin Module:**
  + Approve results, generate transcripts, manage users.
* **Result Processing:**
  + Automatic grade computation (A–F system), GPA & CGPA calculation.
* **Transcript System:**
  + Students can request transcripts digitally.

**2.2 Non-Functional Requirements**

* **Performance:** Handle up to 5,000 concurrent users.
* **Security:** Encrypted passwords, secure data transfer (HTTPS).
* **Scalability:** Modular design for future expansion (fees, e-learning, library system).
* **Usability:** Mobile-friendly interface.

**3. System Design**

**3.1 Architecture**

* **Client-Server model** with RESTful APIs.
* **Three-tier architecture:**
  1. Presentation Layer (Frontend)
  2. Application Layer (Express backend)
  3. Database Layer (MongoDB)

**3.2 Database Design (MongoDB Collections)**

* **Users** (staff, students, admin)
* **Courses**
* **Results** (studentId, courseId, score, grade, semester, session)
* **Transcripts**
* **Authentication tokens**

**3.3 Use Case Diagram**

* Student → View Results, Register Courses, Request Transcript
* Lecturer → Upload Results, Edit Results
* Admin → Approve Results, Generate Transcript

**4. Implementation Plan**

1. **Phase 1:** User authentication & roles
2. **Phase 2:** Student dashboard (profile, course registration)
3. **Phase 3:** Lecturer dashboard (upload results)
4. **Phase 4:** Admin dashboard (approve, manage results)
5. **Phase 5:** Result processing engine (grades, GPA, transcript)
6. **Phase 6:** Testing & deployment

**5. Testing Strategy**

* **Unit Testing:** Test individual modules (login, result upload).
* **Integration Testing:** Ensure frontend and backend work together.
* **System Testing:** Validate the whole portal with dummy data.
* **User Acceptance Testing (UAT):** Run pilot with students & staff.

**6. Deployment Plan**

* Host backend on **Node.js server**.
* Use **MongoDB Atlas** for cloud database.
* Deploy frontend on school server or cloud hosting (Netlify, Vercel, etc).
* Use CI/CD for updates.

**7. Maintenance & Future Enhancements**

* Bug fixes and security patches.
* Future modules:
  + Fees & Payment System
  + Library Integration
  + Learning Management (E-learning)

**8. References**

* Node.js & Express Documentation
* MongoDB Documentation
* JWT Authentication Best Practices