CP630 project proposal
Smart Student Performance Analytics Platform
Student: Adeniyi Ridwan Adetunji
Student ID: 245852450

**Introduction**

Every institution faces important challenges in identifying struggling students before it's too late. grading systems provide feedback only after assessments are completed, leaving small time for intervention. Research shows that immature identification of at risk students can improve pass rates by 15-20% through timely intervention.

The Smart Student Performance Analytics Platform is a data oriented enterprise computing application that leverages machine learning to predict student performance, identify at risk students, and provide personalized recommendations. By analyzing binary performance indicators including attendance, quiz scores, assignment grades, participation , and study habits , the platform generates true predictions and actionable insights for educators.

This project will be built with the knowledge gained from the enterprise computing course, The goal is to create a comprehensive enterprise computing solution that addresses a real world educational problem while demonstrating mastery of data processing , enterprise Java technologies, and deployment practices .

Target Users:
- Primary: Teachers and academic advisors for early intervention
- Secondary: Students seeking performance insights and recommendations
- Tertiary: Department heads for analytics and decision-making

**Problem solving and algorithms**
**1. Application data**
Data Representation and Dataset Design .
The dataset contains student performance records with the next structure:.

Input Features (8 attributes):.
- student_id: Unique identifier (Integer).
- attendance_percentage: Class attendance (0-100%).
- quiz_average: Average quiz score (0-100 ).
- assignment_average: Average assignment score (0-100 ).
- midterm_score: Midterm exam score (0-100 ).
- participation_score: Class participation (1-10 scale).
- study_hours_per_week: Self reported study time (0-20 hours).
- previous_gpa: Prior semester GPA (0.0-4 .0).

Target Variables:.
- final_grade: Letter grade (A, B , C , D, F) - Classification target.
- final_score: Numeric score (0-100 ) - Regression target .
- pass_fail: Binary outcome (Pass/Fail) - Risk assessment target .

Dataset Characteristics:.
- Total instances: 1000+ student records .
- Class distribution: Approximately balanced (Pass: ~75%, Fail: ~25%).
- Feature types: Continuous (scores , percentages) and unconditional (grade).
- Format: CSV for naked data , JSON for polished data and models

**2. Models to represent information, knowledge and patterns.**

Model 1: Grade Classification Model .
- Algorithm: logistical Regression / J48 Decision Tree .
- Purpose: Predict letter grade (A/B/C/D/F).
- Expected Accuracy: >85%.

Model 2: Score Prediction Model .
- Algorithm: Linear Regression .
- Purpose: Predict denotive last score (0-100 ).
- Expected MAE: <5 points .

Model 3: Risk Assessment Model.
- Algorithm: Binary logistical Regression .
- Purpose: Identify at risk students (Pass/Fail).
- Expected Accuracy: >90%.

Model 4: Student Clustering Model (Bonus).
- Algorithm: K-Means Clustering.
- Purpose: Group students by performance patterns .
- Clusters: 3 groups (High/Medium/Low performers).

Model Storage: All models serialized in JSON format for easygoing loading in Java services.

**3. Algorithms to solve the problems and compute the models.**

Step 1: Data Loading and Exploration .
-> Load student_data .csv .
-> Descriptive statistics and visualization .

Step 2: Data Preprocessing .
-> Normalize/standardize features .
-> Encode unconditional variables.
-> Split dataset (80% train , 20% test).

Step 3: Model Training.
-> Train Grade Classifier (Logistic Regression).
-> Train Score Predictor (Linear Regression).
-> Train Risk Assessor (Binary Classification).
-> Train Student Clusterer (K-Means).

Step 4: Model Evaluation.
-> Calculate accuracy, precision , recall , F1-score .
-> Cross validation (10 fold).
-> Select best hyperparameters.

Step 5: Model Serialization.
-> Export models to JSON format.
-> Save in models/ directory
Tools: Python 3.9+ with scikit-learn,pandas,numpy, WEKA(optional) for validation

**Proposed System Design**
1. The platform follows a microservices architecture with three Spring Boot services deployed as Docker containers, providing REST APIs for prediction , analytics, and recommendations.

2. **System Architecture**

User Roles and Permissions .
Admin (Role: 1)

- User management (CRUD operations on teachers and students).
- Course management (create courses , assign teachers).
- Enrollment management (enroll students to courses).
- View system wide analytics. - Access all data across the platform .

Teacher (Role: 2)

- View courses they teach.
- View students enrolled in their courses.
- Input/edit performance data for their students only
- Generate predictions for their students

- View analytics for their classes.
- Cannot access different teachers' data.

Student (Role: 3)
- View own performance data (read-only).
- View own predicted grades.
- View personalized recommendations.
- Track own progress.
- Cannot access different students' data .

Database Architecture.

Core Tables:

1 . `users` - Authentication (username , password_hash, role).
2 . `teachers` - Teacher profiles (linked to users).
3. `students` - Student profiles (linked to users).
4. `courses` - Course information (code, name, teacher_id).
5 . `enrollments` - Student Course relationships.
6. `performance` - Performance metrics (by student , by course).
7 . `predictions` - Prediction results.

Key Relationships:
- One teacher teaches multiple courses .
- One course has multiple students (through enrollments).
- One student enrolled in multiple courses .
- Each student has performance data per course.
- Each performance record generates one prediction .

Microservices Architecture.

**Service 1: User & Auth Service (Port 8080).**
- User CRUD operations (admin only).
- Authentication (JWT tokens).
- Role based authorization .

Endpoints:.
- `POST /register` - Admin adds users .
- `POST /login` - User authentication.
- `GET /users` - List users (admin only).
- `PUT /users/{id}` - Edit user (admin only).
- `DELETE /users/{id}` - Delete user (admin only).

**Service 2: Course & Enrollment Service (Port 8081)**
- Course management.
- Enrollment operations .
- Teacher course assignments .

Endpoints:.
- `POST /courses` Create course (admin only).
- `GET /courses/teacher/{teacherId}` - Teacher's courses .

- `POST /enrollments` - Enroll student (admin only).
- `GET /enrollments/course/{courseId}` Students in course .

**Service 3: Performance Service (Port 8082)**

- Input performance data. - Update existing records .
- Retrieve performance by student/course.

Endpoints:.

- `POST /performance` - Save performance (teacher only, own students).
- `GET /performance/{studentId}/{courseId}` Get data.
- PUT /performance/{id}` - Update data (teacher only)

**Service 4: Prediction Service (Port 8083)**

- Load ML models.
- Generate predictions.
- Save prediction results .

Endpoints:.

- `GET /predict/grade` - Predict grade .
- `GET /predict/score` - Predict score .
- `GET /predict/risk/{studentId}` - Assess risk .

**Service 5: Analytics Service (Port 8084).**

- Class statistics .
- Performance trends.
- Grade distributions.

Endpoints:.

- `GET /analytics/class/{courseId} - Class stats (teacher for own courses).
- `GET /analytics/student/{studentId}` - Student progress.
- `GET /analytics/system` - System wide (admin only).

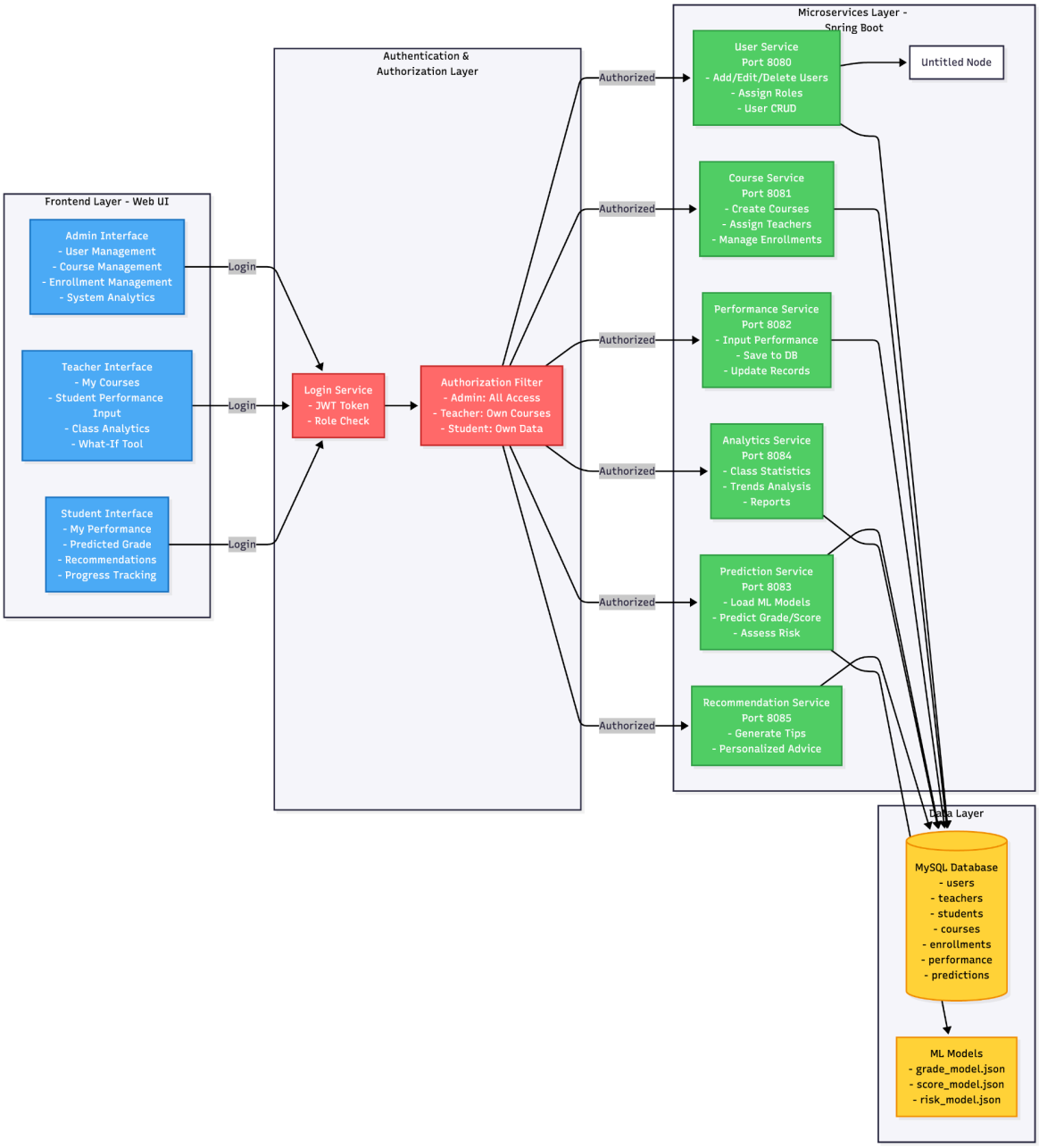
**Service 6: Recommendation Service (Port 8085)**.

- Generate study recommendations .
- Suggest resources .
- Personalized advice.

Endpoints:.

- `GET /recommend/{studentId}` - Get recommendations

## System Architecture Diagram




Service Components:.
Prediction Service (Spring Boot).
- Load ML models from JSON files .
- REST API endpoints for grade/score/risk prediction.
- Batch prediction capability.

Analytics Service (Spring Boot).
- Calculate class statistics .
- Track student progress over time .
- Generate performance reports .

Recommendation Service (Spring Boot).
- Generate personalized study recommendations .
- Suggest resources based on faint areas .
- Rule based testimonial engine .

Frontend (Web Application).
- Teacher Dashboard: student list , at risk alerts , class statistics, prediction form.
- Student Portal: view predictions, recommendations, progress tracking.
- Technologies: HTML5 , CSS3 (Bootstrap 5), JavaScript, Chart.js .

Database:.
- MySQL for production (student records, predictions).
- H2 for development/testing .
- Redis for caching (optional)

3. Platform and tools to be used in the project.
Backend Technologies:.
- Java 11+.
- Spring Boot 2.7+.
- Spring Framework (Web , Data JPA , REST).
- Maven 3.8+ for build management.
- MySQL 8.0 / H2 Database .

Machine Learning:.
- Python 3.9+ (scikit-learn , pandas , numpy).
- Weka API (optional for validation).
- Google Colab for experimentation .

Frontend:.
- HTML5, CSS3, JavaScript.
- Bootstrap 5 for susceptible design .
- Chart.js for data visualization .

Deployment:.
- Docker 24 .0+ for containerization.
- Docker Compose for orchestration.
- Embedded Tomcat (Spring Boot).

Development Tools:.
- Eclipse / VS Code.
- Git + GitHub for version control.
- Boomrang/Postman for API testing

**Project plan and schedule**

| Task ID | Description | Due date | Lead |
|---|---|---|---|
| 1 | Project research & team up | Day 7 of week 9 | Adeniyi Ridwan |
| 2 | Project Proposal | Day 6 of week 10 | Adeniyi Ridwan |
| 3 | Data Collection and Prepocessing | Day 6 of Week 11 | Adeniyi Ridwan |
| 4 | Model Training and Validation | Day 6 of week 11 | Adeniyi Ridwan |
| 5 | Prediction Development | Day 2 of week 12 | Adeniyi Ridwan |
| 6 | Analytic and Recommendation | Day 6 of week 12 | Adeniyi Ridwan |
| 7 | Database Setup and Integration | Day 2 of week 13 | Adeniyi Ridwan |
| 8 | Docker Development | Day 4 of week 13 | Adeniyi Ridwan |
| 9 | Frontend Development | Day 6 of week 13 | Adeniyi Ridwan |
| 10 | Testing and fixing | Day 2 of week 14 | Adeniyi Ridwan |
| 11 | Documentation and demo video | Day 4 of week 14 | Adeniyi Ridwan |
| 12 | Final submission | Day 6 of week 14 | Adeniyi Ridwan |

**References**

1. Romero , C., & Ventura, S . (2020). "Educational Data Mining and Learning Analytics: An Updated Survey ." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(3). 2. Kotsiantis , S. B . (2012). "Use of Machine Learning Techniques for Educational Proposes ." Artificial Intelligence Review, 37(4), 331-344 .
3. Gray , G ., McGuinness , C., & Owende , P. (2014). "An Application of Classification Models to Predict Learner Progression ." IEEE International Advance Computing Conference .
4. UCI Machine Learning Repository - Student Performance Dataset: https://archive .ics.uci.edu/ml/datasets/student+performance.
5. Spring Boot Documentation: https://spring.io/projects/spring-boot.
6. Docker Documentation: https://docs.docker.com/. Scikit learn Documentation: https://scikit-learn.org/.
7. Weka 3: Machine Learning Software in Java: https://www.cs.waikato.ac.nz/ml/weka/
8. CP630OC Labs and Assignments

**Appendices**

Additional Features to be implemented:.
Security Enhancements:.

- JWT based authentication for API endpoints .
- Password encryption (BCrypt) for user accounts .
- HTTPS for web component access .

Role based access control (Teacher vs. Student).

- Performance Optimization:. Redis caching for steady prediction queries .
- Database connection pooling.
- Lazy loading and pagination for massive datasets .
- API rate limiting to prevent abuse.

Innovative Deployment:.

- Docker Compose for multi-container orchestration .
- Health checks for all microservices .
- Centralized logging.
- Kubernetes deployment configuration .

Other Features:.

- Real time email notifications for at risk students .
- PDF report generation (JasperReports).
- Data export (CSV, Excel).
- Batch prediction for smooth classes.
- Charts and dashboards (Chart.js).
- Mobile responsive design.

Massive Data Consideration:

- Ascendible architecture supports massive datasets .
- Can integrate with Apache Spark for distributed processing.
- Model versioning and A/B testing capability

Future works:

- NoSQL database(Cassandra) for big data later on, but cassandra java API will be used instead of JPA
- Natural Language Processing(Transformers): we can implement by using python flask microservices with transformers to analyze student feedback text using hugging face to generate personalized study guides and recommendations or analyze student comments