**Software Requirements Specification (SRS)**

**Using Machine Learning for Early Prediction of Student Academic Performance**

Serge Francis Ineza N.

**1. Problem Statement**

Educational institutions often rely on post-hoc academic performance data to identify struggling students. By the time action is taken, it's often too late. This project aims to bridge that gap by developing a machine learning-based system that uses early-semester data such as attendance, assignment submissions, and LMS (e.g., Canvas) activity to predict academic outcomes. The goal is to provide educators with timely risk scores and explanations to enable proactive intervention.

**2. Introduction**

**2.1 Purpose**

This Software Requirements Specification defines the requirements for a Machine Learning-based Student Performance Prediction System that enables early identification of at-risk students through predictive analytics.

**2.2 Scope**

The system will analyze early-term student data to predict academic outcomes, providing educators with actionable insights for timely intervention. The system includes data processing, machine learning model training, risk prediction, and reporting capabilities.

**2.3 Definitions and Acronyms**

* **ML**: Machine Learning
* **LMS**: Learning Management System (e.g., Canvas, Blackboard)
* **SVM**: Support Vector Machine
* **KNN**: K-Nearest Neighbors
* **SHAP**: SHapley Additive exPlanations for model interpretability

**3. Overall Description**

**3.1 Product Perspective**

The system operates as a standalone web-based application that integrates with existing educational databases and LMS platforms to provide predictive analytics for student performance.

**3.2 Product Functions**

* Upload and preprocess academic datasets (grades, attendance, LMS data)
* Train ML models (SVM, Decision Tree, KNN) to predict academic outcomes
* Display individual student risk scores with explanations
* Allow comparison of different models' performance
* Export at-risk student reports

**3.3 User Classes**

* **Educators**: Primary users who identify at-risk students for intervention
* **Data Analysts**: Technical users who train and evaluate ML models
* **Administrators**: System managers who oversee operations and access

**3.4 Operating Environment**

* **Frontend**: Web browsers (desktop and tablet compatible)
* **Backend**: Python-based system with Flask framework
* **Database**: SQL database for data storage
* **ML Framework**: scikit-learn for model implementation

**4. Functional Requirements**

**4.1 Data Management**

* The system shall allow users to upload academic datasets in CSV and Excel formats
* The system shall preprocess uploaded data including grades, attendance, and LMS activity data
* The system shall validate data integrity and handle missing values
* The system shall store processed data securely in the database

**4.2 Machine Learning Capabilities**

* The system shall train Support Vector Machine (SVM) models for academic outcome prediction
* The system shall train Decision Tree models for academic outcome prediction
* The system shall train K-Nearest Neighbors (KNN) models for academic outcome prediction
* The system shall evaluate model performance using accuracy, precision, and recall metrics
* The system shall allow comparison of different models' performance side-by-side

**4.3 Prediction and Analysis**

* The system shall generate individual student risk scores based on early-term data
* The system shall provide explanations for risk predictions using interpretable AI techniques
* The system shall classify risk levels as Low, Medium, High, or Critical
* The system shall suggest intervention recommendations based on risk factors

**4.4 User Interface and Reporting**

* The system shall display student risk scores in an intuitive dashboard interface
* The system shall allow filtering and searching of students by risk level
* The system shall export at-risk student reports in PDF and Excel formats

**5. Non-Functional Requirements**

**5.1 Performance Requirements**

* The system shall provide predictions within 30 seconds for each input
* The system shall support concurrent access by up to 50 users
* The system shall handle datasets with up to 10,000 student records

**5.2 Usability Requirements**

* The system interface shall be accessible on both desktop and tablet devices
* The dashboard shall be user-friendly and require minimal technical expertise

**5.4 Security Requirements**

* All user actions shall be logged for audit purposes

**6. Use Cases**

**Use Case 1: Upload and Preprocess Academic Data**

**Primary Actor**: Educator/Data Analyst  
**Goal**: Successfully upload and prepare student data for analysis

**Preconditions**: User is authenticated and has data upload permissions

**Main Flow**:

1. User navigates to data upload interface
2. User selects CSV or Excel file containing student data
3. System validates file format and the required columns
4. System displays data preview with basic statistics
5. User confirms data upload
6. System preprocesses data (handles missing values, normalizes features)
7. System stores processed data and confirms successful upload

**Alternate Flow**: Large Dataset Processing

* If file size > 50MB, system displays progress bar and processes in chunks
* System provides estimated completion time

**Exception Flow**: Invalid Data Format

* System displays error message specifying format requirements
* System provides sample data template for download
* User must correct file and retry upload

**Use Case 2: Generate Student Risk Predictions**

**Primary Actor**: Educator  
**Goal**: Obtain risk assessment for individual students with explanations

**Preconditions**: Trained ML models are available, and student data exists

**Main Flow**:

1. User accesses student dashboard
2. User searches for or selects specific student
3. System retrieves student's current academic data
4. System generates risk prediction using trained model
5. System creates explanation using SHAP values
6. System displays risk level, confidence score, and key risk factors
7. System provides intervention recommendations
8. User reviews prediction and may add notes for follow-up

**Alternate Flow**: Batch Prediction

* User selects multiple students for simultaneous prediction
* System processes all selected students and displays summary table
* User can drill down into individual student details

**Exception Flow**: Incomplete Student Data

* System identifies missing critical data elements
* System provides prediction with reduced confidence
* System highlights specific data needed for better accuracy

**7. System Architecture Overview**

**7.1 High-Level Components**

* **Data Processing Layer**: Handles upload, validation, and preprocessing
* **Machine Learning Engine**: Trains models and generates predictions
* **Web Interface Layer**: Provides user dashboard and interaction
* **Database Layer**: Stores student data, models, and predictions
* **Reporting Module**: Generates and exports analytical reports

**7.2 Technology Stack**

* **Backend**: Python 3.8+, Flask web framework
* **ML Libraries**: scikit-learn, pandas, numpy
* **Frontend**: HTML5, CSS3, JavaScript, Bootstrap
* **Database**: PostgreSQL or MySQL
* **Visualization**: Chart.js, D3.js for dashboard graphics

**8. Acceptance Criteria**

**8.1 Functional Acceptance**

* All functional requirements are implemented and tested
* Use cases execute successfully with expected outcomes
* ML models achieve minimum 70% accuracy on validation data

**8.2 Performance Acceptance**

* Prediction generation completes within 30-second requirement
* System supports specified user load and data volume
* Interface responds within 3 seconds for standard operations

**8.3 Usability Acceptance**

* Educators can use system with minimal training
* Interface works correctly on desktop and tablet devices

**9. Acceptance Criteria**

**9.1 Use case diagram**

This diagram shows the main actors (Educator, Data Analyst, Administrator) and their interactions with the system through various use cases.

A diagram of a student performance system

AI-generated content may be incorrect.

**9.2 Class Diagram**

This diagram illustrates the main classes, their attributes, methods, and relationships within the ML prediction system.

A diagram of a person's work flow

AI-generated content may be incorrect.