Student Performance Predictor System

Team:

Mahmoud Osama
Ibrahim Amr
Ahmed Gamal
Abdelrahman Emam
Nedda Hany
Aya Mohamed

---

1. Introduction

The **Student Performance Predictor System** is an intelligent web-based application designed to predict students' exam scores using Machine Learning techniques.
The system integrates **data preprocessing, predictive modeling, analytics, and secure user management** to help students understand their academic performance and receive actionable insights.

2. Project Objectives

- Predict student exam scores accurately.
- Provide AI-driven insights and recommendations.
- Allow users to track performance over time.
- Secure user authentication and data privacy.
- Visualize analytics and trends.

- Build a scalable and modular system architecture.

---

3. System Scope

In Scope

- User registration and authentication.
- Exam score prediction using ML.
- Prediction history storage.
- Analytics and visualizations.
- Profile management.
- Exporting prediction data.

Out of Scope

- Real-time data collection from schools.
- External LMS integration.
- Automatic dataset updates.

4️⃣ Users & Stakeholders

Actor	Description
Student User	Main system user
System Admin	Maintains dataset and models
ML Engine	External logical component

5️⃣ Functional Requirements

Authentication

- Register new users.
- Login with username or email.
- JWT-based session management.
- Logout and token refresh.

Prediction

- Input student academic factors.
- Validate and preprocess data.
- Predict exam score.
- Interpret performance level.
- Generate AI insights.

Data Management

- Save predictions.
- View prediction history.
- Delete records.
- Export predictions to CSV.

Analytics

- View performance statistics.
- Track score trends.
- Compare results with dataset benchmarks.

Profile

- View and update profile.
- Change password securely.

6️⃣ Non-Functional Requirements

- **Security:** JWT, password hashing.
- **Performance:** Cached models and preprocessors.
- **Scalability:** Modular architecture.
- **Usability:** Simple and responsive UI.

- **Reliability:** MongoDB persistence.

7️⃣ System Architecture

The system follows a **Layered Architecture**:

1. Client Layer (Frontend)
2. Backend Layer (Flask APIs)
3. Machine Learning Layer
4. Data Layer (MongoDB + Artifacts)

Each layer communicates through well-defined interfaces, ensuring separation of concerns.

8️⃣ Technology Stack

Layer	Technologies
Frontend	HTML, CSS, JavaScript
Backend	Python, Flask
ML	Scikit-learn, TensorFlow
Database	MongoDB
Auth	JWT
Visualization	Plotly
Deployment	Localhost

9️⃣ Machine Learning Pipeline

Dataset

- Source: StudentPerformanceFactors.csv
- Preprocessed to remove duplicates and missing values.

Preprocessing

- Numerical features: StandardScaler
- Categorical features: OneHotEncoder

Models

- Neural Network (Primary)
- Linear Regression (Benchmark)

Evaluation Metrics

- MAE
- MSE
- RMSE
- R^2

🔟 System Modules Description

◇ Authentication Module

Handles:

- User registration
- Login & JWT generation
- Session validation

◇ Prediction Module

Handles:

- Input validation
- Feature preprocessing

- ML inference
- Result interpretation

◇ **Analytics Module**

Handles:

- Statistics aggregation
- Trend analysis
- Charts generation

◇ **Recommendation & Insights Module**

Generates:

- Study advice
- Attendance warnings
- Personalized improvement plans

◇ **Profile Management Module**

Handles:

- Profile updates
- Password changes
- User activity tracking

---

11. Data Storage Design

MongoDB Collections

Users Collection

- user_id
- username
- email
- password (hashed)
- created_at
- last_login

Predictions Collection

- user_id
- inputs
- predicted_score
- level
- timestamp

---

12. Security Design

- JWT authentication on protected routes.
- Password hashing using SHA-256.
- HTTP-only cookies.
- Input validation on backend.

---

13. System Diagrams Included

- Use Case Diagram (Integrated)
- Sequence Diagrams
- Activity Diagrams
- DFD (Level 0 & 1)
- Block Diagram
- Package Diagram
- Component Diagram
- Architecture Diagram

1️⃣4️⃣ Error Handling & Validation

- Client-side input validation.
- Backend validation with clear error messages.
- Graceful handling of ML and DB failures.

1️⃣5️⃣ Testing Strategy

- Manual testing of APIs.
- Model performance validation.
- Authentication flow testing.
- Prediction accuracy testing.

---

16 Conclusion

The **Student Performance Predictor System** demonstrates a complete integration of **Machine Learning, Web Development, and Data Analytics**.
The project follows best practices in software architecture, security, and ML deployment, making it suitable for academic evaluation and real-world extension.