Student Performance Prediction System

1. General description of the system

The Student Performance Prediction System is an intelligent web-based application built using the Django framework. Its primary purpose is to support educational institutions in effectively monitoring and analyzing student progress throughout the academic year. By collecting and processing key performance indicators—such as attendance, assignment scores, participation, and past exam results—the system uses predictive analytics to estimate a student's final grades.

2. System Architecture

This system is built using Django's MVT (Model-View-Template) architecture and is made up of several key components:

2.1. Database

Uses PostgreSQL to store all data
Deployed in containers using Docker and docker-compose for easier setup and scaling

2.2. Backend

Built with Django 4.x to handle logic and data flow
Integrates machine learning models (using scikit-learn) for making grade predictions

2.3. Frontend

Built with HTML and CSS, styled using Bootstrap
Uses JavaScript for interactive features
Designed to work well on both mobile devices and desktops

3. Main Modules of the System

3.1. Authentication Module

Handles user registration and login
Supports different roles (admin and student)
Allows users to manage their profiles

3.2. Dashboard Module

Shows performance data in a clear and organized way
Lets administrators manage courses and assignments
Tracks how students are progressing in real time

3.3. Machine Learning Module

Trains models on historical student data
Predicts student performance
Helps explain which factors influence grades the most

4. Data Models

4.1. User

Builds on Django's default user model
Adds fields like student number and user role (student or admin)
Uses email for communication

4.2. Course

Stores course name, code, and description
Links students to the courses they're enrolled in
Includes assignments tied to each course

4.3. Assignment

Includes title, description, due date, and max points
Connected to specific courses
Stores grades submitted by students

4.4. Grade

Records the score and submission time
Linked to both the student and the specific assignment
Ensures one grade per student per assignment

4.5. Attendance

Tracks whether a student was present on a given date
Tied to specific students and courses
Each entry is unique for that student-course-date combo

4.6. Participation

Records how actively a student took part in a class (low, medium, high)
Can include notes
Also tied uniquely to a student and course on a specific dat

4.7. Prediction

Stores the predicted final score and the model's confidence level
Linked to a specific student and course

5. Functionality of the System

5.1. For Administrators

5.1.1. Course Management

Create, view, edit, and delete courses

5.1.2. Assignment Management

Add new assignments to courses
Set due dates and max scores
Track assignment status (upcoming, active, or completed)

5.1.3. Student Management

View the full list of students
Filter them by course
See detailed profiles

5.1.4. Academic Performance Tracking

Enter grades for assignments
Mark attendance
Log student participation levels

5.1.5. Analytics and Forecasting

Run the prediction algorithm to estimate grades
View predictions by student or course
Explore statistics and performance trends

5.2. For Students

5.2.1. View Personal Info

See personal profile, enrolled courses, and GPA

5.2.2. Monitor Progress

Check assignment grades
View attendance records
See predicted final grades

5.2.3. Notifications

Get alerts for new assignments or grades
Receive warnings for low attendance

6. Machine Learning Module

6.1. Technologies and Libraries

- ➔ Python 3.9
- ➔ scikit-learn for machine learning
- ➔ pandas for data handling
- ➔ numpy for math operations
- ➔ joblib for saving/loading models

6.2. Algorithms Used

RandomForestRegressor: predicts the final grade (0–100)
LogisticRegression: estimates the chance of passing the course

6.3. Model Inputs

avg_grade: current average grade in the course
attendance_rate: percentage of classes attended
participation_level: student's level of activity in class (0–2)

6.4. Model Outputs

predicted_score: estimated final grade
confidence: how sure the model is about the prediction

6.5. Model Training Process

1. Load past student data
2. Split into training and test sets
3. Normalize values with StandardScaler
4. Train models (Random Forest & Logistic Regression)
5. Evaluate how well they perform
6. Save the models for future use

6.6. Forecasting Process

1. Load the saved model and scaler
2. Gather current student data
3. Preprocess it
4. Predict final grades
5. Estimate the chance of passing
6. Save the results in the system

7. System Access Scheme

7.1. Home Pages

```
/: Login
/signup/: Registration
/confirm/: Confirmation after signup
```

7.2. General Pages

```
/dashboard/: Main dashboard (different for each user type)
/notifications/: User alerts and updates
/support/: Help and support page
/settings/: User preferences
```

7.3. Administrative Pages

```
/dashboard/admin/courses/: Manage courses
/dashboard/admin/students/: Manage students
/dashboard/admin/run-predictions/: Run prediction tool
/dashboard/course/{course_id}/: View course details
/dashboard/student/{student_id}/: View student details
```

7.4. Functional Pages

```
Add assignments: /dashboard/course/{course_id}/add-assignment/
Add grades:
/dashboard/assignment/{assignment_id}/student/{student_id}/add-grade/
Mark attendance: /dashboard/course/{course_id}/attendance/
Track participation: /dashboard/course/{course_id}/participation/
```

8. Integration and Deployment

8.1. Docker Containers

web: The Django app running on Gunicorn
db: PostgreSQL database
nginx: Serves static files and routes web requests

8.2. Configuration

Uses environment variables to manage settings:
`DEBUG, SECRET_KEY, DATABASE, SQL_* for connection details`

8.3. Static Files

CSS for styling
JavaScript for user interactions
Icons and other graphics

9. Instructions for Starting the System

9.1. Requirements

Docker and docker-compose
Git for downloading the code

9.2. Setup Steps

1. Clone the repository
2. Create a .env file with necessary settings
3. Run docker-compose up
4. A superuser is created automatically
5. Access the app via browser at http://localhost

10. Conclusion

The Student Performance Prediction System is a comprehensive solution for educational institutions that helps administrators to track student performance and make informed decisions based on machine learning predictions. The modular architecture of the system ensures flexibility and scalability, and the intuitive interface allows users to effectively interact with the system.