A

PROJECT REPORT ON

STUDENT PERFORMANCE INDICATOR

SUBMITED BY

MISS. UJWALA SRINIWAS KUSMA

SUBMITED TO

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN PATIAL FULLFILLMENT OF DEGREE

MASTERS OF COMPUTER APPLICATIONS (SEM-I)

UNDER THE GUIDANCE OF

Dr. Shveti Chandan

Through,

Sadhu Vaswani Institute of Management and Studies for Girls,

Koregaon Park, Pune-411001

2024-25

CERTIFICATE

SVIMS
Sadhu Vaswani Institute of Management Studies
BELIEVE AND ACHIEVE

This is to certify that the mini project report entitled, "**Student Performance Indicator**" being submitted herewith in partial fulfillment of requirement of the award of the degree of **Masters in Computer Applications** (SEM-I) to Savitribai Phule Pune university, Pune is the result of original project completed by Ujwala Sriniwas Kusuma under my supervision and guidance and to the best of my knowledge and belief, the work embodies in this project has not found earlier the basis for the award of degree of any similar title or any other university or examining body.

Date:

Place: Pune

Dr. Shveti Chandan	**Dr. Neeta Raskar**	**Dr. B. H Nanwani**
Project Guide	**HOD**	**Director**

Examiner 1 sign　　　　Examiner 2 sign

DECLARATION BY STUDENT

To

The Director

SVIMS, Koregaon Park, Pune

I undersigned declare that this project titled "**Student Performance Indicator**" Written submitted by me to SPPU, Pune, in partial fulfillment of the requirement and award of the degree of Masters of Computer Applications (MCA-I) under the guidance of Dr. Shveti Chandan, is my original work.

I further declare that to the best of my knowledge and belief, this project has not been submitted to this or any other University or Indian institute for award of any degree.

Place: Pune

Date:

(Ujwala Sriniwas Kusma)

ACKNOWLEDGEMENT

I extend my sincere gratitude to DR. B. H. Nanwani, Dr. Neeta Raskar and Dr. Shveti Chandan for allowing me to carry out the study and for their constant encouragement, valuable suggestions, and guidance during the research work.

I extend my special thanks to Prof. Samiksha yeole and Prof. Neeta Raskar for their kind co-operation and inspiration.

I extend my special gratitude to my dearest family members and friends who encouraged and motivated me to complete the project report.

Place: Pune
Date:

Ujwala Sriniwas Kusma

INDEX

CHAPTER 1
INTRODUCTION

1.1 Client/Organization Profile

Name: Student Performance Indicator

Location: Thane, India

About the Organization:

The **Student Performance Indicator** is a machine learning-based web application designed to predict a student's percentage in a particular subject based on various input scores. This project aims to provide a reliable tool that can assist educators, students, and academic institutions in understanding and tracking academic performance.

The system utilizes predictive modeling techniques, taking into account different factors such as previous academic records, attendance, and other relevant input variables to forecast a student's potential outcome. By doing so, it offers an early indication of how a student might perform, allowing for timely interventions and support.

In recent years, educational institutions have increasingly recognized the importance of data-driven insights to improve student outcomes. As academic pressures grow, students and educators alike benefit from tools that provide personalized predictions and feedback. With this context in mind, the **Student Performance Indicator** serves as a valuable resource for:

- **Students**: Helping them identify areas that need improvement and guiding their study efforts accordingly.
- **Teachers**: Providing them with insights into students' performance trends, enabling more targeted support.
- **Parents**: Offering them a clear view of their child's academic progress.

This project is built using a comprehensive machine learning pipeline, starting from data collection and preprocessing, to model training, and finally, deployment. The application is deployed on Hugging Face Spaces, making it easily accessible, and is containerized using Docker to ensure it runs seamlessly across various environments.

The goal of this project is to offer a scalable, user-friendly solution that can enhance student performance tracking and prediction while contributing to the overall improvement of academic success in schools and universities.

1.2 Need of the Project

In today’s education system, there is a growing emphasis on the importance of tracking student performance. While traditional evaluation methods such as exams and quizzes provide a snapshot of student achievement, they do not always give a clear prediction of future performance or identify areas where students may need additional support. The **Student Performance Indicator** project aims to fill this gap by leveraging machine learning to provide more precise, data-driven predictions of student outcomes.

Key Reasons for the Need of the Project:

Early Identification of At-Risk Students:

- One of the main reasons for developing this project is to identify students who are at risk of underperforming before it becomes a critical issue. Early prediction enables teachers and administrators to intervene with appropriate measures, such as additional tutoring or counseling, to help students improve their performance before the final evaluation.

Data-Driven Decision Making:

- Educational institutions often rely on intuition or historical data to assess student performance, which can sometimes be subjective or limited. This project enables data-driven decision-making by analyzing various factors such as previous scores, attendance, and study habits. It offers more accurate and personalized predictions, leading to better planning and resource allocation.

Efficient Resource Allocation:

- By predicting the academic performance of students, schools can efficiently allocate their resources. For example, they can assign extra support to students who are likely to underperform or streamline their mentoring programs to focus on students who need the most assistance. This results in a more targeted approach to education management.

Personalized Learning:

- Every student learns differently, and their academic progress can vary greatly depending on various internal and external factors. This project allows for the creation of personalized learning strategies based on predicted performance. Teachers can customize their teaching methods, offering personalized feedback to students based on their predicted outcomes.

Better Communication Between Teachers, Students, and Parents:

- The prediction model enables teachers to communicate more effectively with both students and parents. By providing data-driven predictions, teachers can give clear guidance on where improvements are needed, helping students focus on the right areas for academic growth. Parents, in turn, gain a clearer understanding of their child's academic trajectory.

Proactive Academic Support:

- Instead of waiting for poor results at the end of a semester or academic year, this project allows for continuous monitoring of student performance. Proactive measures can be taken to support students, offering remedial classes or alternative learning materials in real time to improve outcomes.

1.3 Scope and Feasibility of the System

Scope of the System

The **Student Performance Indicator** project offers a wide range of applications and benefits for educational institutions, students, and teachers. The system’s predictive model, which forecasts student performance based on multiple factors, can be expanded and integrated into various educational contexts. Its utility goes beyond basic grade prediction, providing comprehensive insights that can help improve teaching methods, resource allocation, and student outcomes.

Key Areas of Application:

Academic Performance Prediction:

The system provides predictions of student percentages in specific subjects based on input features such as previous scores, attendance, and study habits. This can be used to gauge future performance, enabling students and teachers to take proactive steps in areas that need improvement.

Customized Learning Plans:

The system can be extended to generate personalized learning recommendations based on predicted performance. For example, students who are at risk of underperforming in specific subjects can receive targeted resources or extra tutoring to address weaknesses.

Institution-Wide Implementation:

Educational institutions can scale the system to monitor and predict the performance of all students, regardless of grade level or subject. This will help administrators allocate resources efficiently, such as assigning additional teachers or mentors where needed.

Teacher and Parent Insights:

The system can be adapted to create dashboards that offer teachers and parents insights into student performance trends. This information can be used to create more effective communication strategies between teachers, students, and parents, fostering a collaborative approach to student success.

a) **Integration with Learning Management Systems (LMS)**:

The system can be integrated with existing Learning Management Systems (LMS) used by schools or universities to provide real-time updates on student performance and predictive insights directly through the institution's digital platform.

b) **Adaptability to Other Educational Levels**:

Though initially developed for high school or university-level students, the system is flexible enough to be adapted for use in other educational contexts, such as middle school students or adult learners in continuing education programs.

c) **Real-Time Monitoring and Reporting**:

The system can be enhanced to provide real-time monitoring of students' progress throughout the semester. Institutions can generate reports that highlight at-risk students or showcase trends that might require institutional intervention.

Feasibility of the System

The development and deployment of the **Student Performance Indicator** project are highly feasible, given the availability of modern machine learning frameworks and cloud-based deployment platforms. The system leverages existing technologies like Flask for web development, Docker for containerization, and Hugging Face Spaces for deployment, ensuring scalability and ease of use.

Technical Feasibility:

- **Technology Stack**: The project is built using a proven technology stack, including Python for programming, Flask for web framework, and Scikit-Learn for machine learning. These tools are widely used and have extensive community support, making them reliable for both development and maintenance.
- **Machine Learning Algorithms**: The system currently uses **Linear Regression**, which is computationally efficient and provides accurate predictions. The model can be easily upgraded or replaced with more complex algorithms (such as Random Forest or Gradient Boosting) as the project expands.
- **Containerization and Deployment**: Using Docker for containerization ensures that the system can be deployed in any environment without compatibility issues. The deployment on Hugging Face Spaces offers a flexible cloud-based solution for easy access and scalability.

- **Data Management**: The system efficiently handles student data through Pandas and NumPy, which allow for smooth data preprocessing and manipulation. The current model is built to handle moderate datasets, but with minor modifications, it can scale to handle larger datasets as needed.

Economic Feasibility:

- **Cost of Development**: The cost of development is minimal, as the system is built using open-source libraries and frameworks. The use of cloud-based deployment platforms like Hugging Face Spaces reduces infrastructure costs.
- **Maintenance Costs**: Maintenance costs are expected to remain low due to the use of Docker, which simplifies version control and system updates. Regular updates to the machine learning model can be made without requiring significant additional resources.

Operational Feasibility:

- **User Adoption**: The system is designed with a simple and intuitive user interface, which ensures that educators, students, and administrators can easily interact with the application. This makes it feasible for institutions with varying levels of technical expertise to adopt and integrate the system into their academic processes.
- **Integration with Existing Systems**: The system can be integrated with existing databases or Learning Management Systems (LMS) used by educational institutions, making it easy to import historical student data for better performance predictions.

Legal and Ethical Feasibility:

- **Data Privacy**: The system will handle sensitive student data, so ensuring data privacy and protection is crucial. All data will be anonymized, and strong security measures will be implemented to comply with educational data privacy regulations such as FERPA (Family Educational Rights and Privacy Act).
- **Fairness and Bias**: The machine learning model will be regularly tested to ensure that it does not introduce any biases in predictions based on gender, race, or socioeconomic background. Ethical AI practices will be followed to ensure fairness in student performance predictions.

1.3 Operating Environment – Hardware and Software:

Hardware Requirements

Development Environment:

- **Processor**: Intel Core i5 (or equivalent) and above
- **RAM**: 8 GB minimum (16 GB recommended)
- **Storage**: 256 GB SSD or higher
- **Graphics Card**: Not mandatory (but a GPU like NVIDIA CUDA-enabled cards is recommended for larger datasets)

Production/Deployment Environment:

- **Cloud Hosting**: Hugging Face Spaces (cloud-based)
- **Processor**: At least 2 vCPUs
- **RAM**: 4 GB or more
- **Storage**: 50 GB (expandable as needed)
- **Docker Compatibility**: Required for containerization

Software Requirements

Development Software:

- **Operating System**: Windows 10, macOS (Catalina or higher), or Linux (Ubuntu 20.04 LTS or higher)
- **Programming Language**: Python 3.7 or higher
- **Web Framework**: Flask
- **Libraries**:
  - Pandas
  - NumPy
  - Scikit-Learn
  - Matplotlib/Seaborn (optional for visualization)
- **IDE**: PyCharm, VS Code, or Jupyter Notebook
- **Version Control**: Git (GitHub for repository management)

Deployment Software:

- **Containerization**: Docker
- **Cloud Platform**: Hugging Face Spaces
- **Web Server**: Gunicorn (for production environment)

1.6 Detailed Description of Technologies Used

The **Student Performance Indicator** project employs a variety of modern technologies, including programming languages, libraries, frameworks, and tools. These technologies are critical for building, deploying, and maintaining the machine learning web application. Below is a detailed description of the key technologies used:

a) Python

Python is the primary programming language used for developing this project. It is chosen due to its simplicity, readability, and extensive support for data science and machine learning through various libraries. Python offers numerous advantages, such as a large developer community, support for multiple platforms, and the availability of powerful libraries for data manipulation, machine learning, and web development.

b) Flask

Flask is a lightweight, micro web framework for Python that is used to build the web application for the Student Performance Indicator. Flask allows developers to quickly create web applications by providing essential components like routing, request handling, and templating, while remaining flexible and extensible. Flask is preferred due to its simplicity and ease of use, making it ideal for small to medium-sized applications.

Key Features of Flask:

- Minimalistic design with no default database or form validation.
- Provides extensions for adding features like authentication, databases, and more.
- Suitable for building RESTful APIs.

c) Scikit-Learn

Scikit-Learn is a widely used machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It is built on top of NumPy, SciPy, and Matplotlib, and provides various algorithms for classification, regression, clustering, and dimensionality reduction. For the **Student Performance Indicator** project, Scikit-Learn is used to develop the **Linear Regression** model, which predicts a student’s performance based on input data.

Key Features of Scikit-Learn:

- Simple and efficient tools for predictive data analysis.
- Supports both supervised and unsupervised learning.
- Easy integration with other Python libraries like Pandas and NumPy.

d)Pandas

Pandas is a Python library designed for data manipulation and analysis. It provides data structures like DataFrames, which make it easy to work with structured data, such as student performance records. In this project, Pandas is used for data cleaning, data manipulation, and handling input data in the form of CSV files. It also helps with organizing and preparing the data for machine learning models.

***Key Features of Pandas**:*

- Provides data structures like Series and DataFrames for handling structured data.
- Efficient data manipulation and aggregation tools.
- Ability to read and write data from/to multiple formats, such as CSV, Excel, and SQL databases.

e) NumPy

NumPy is the fundamental package for numerical computing in Python. It provides support for working with large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. In the **Student Performance Indicator** project, NumPy is used for numerical calculations, data preprocessing, and handling array operations that feed into the machine learning model.

***Key Features of NumPy**:*

- Provides a high-performance multidimensional array object.
- Offers tools for performing mathematical operations on arrays.
- Supports a wide range of functions for linear algebra, random number generation, and Fourier transformations.

f) Docker

Docker is a platform used to containerize the application, ensuring that it runs the same way across different environments, whether in development, testing, or production. By using Docker, the **Student Performance Indicator** project can be packaged into a container that includes all dependencies, such as libraries and runtime configurations, which simplifies deployment and enhances portability.

***Key Features of Docker**:*

- Allows applications to run in isolated containers, ensuring consistency across different environments.
- Simplifies the deployment process by packaging the application and its dependencies into a single container.
- Provides scalability and efficient resource management for cloud-based applications.

g) Hugging Face Spaces

Hugging Face Spaces is the platform used to deploy the **Student Performance Indicator** web application. Hugging Face Spaces supports easy hosting of machine learning models and apps built with Streamlit, Gradio, or Flask. It is particularly useful for public deployment of machine learning projects, offering a simple and free way to showcase the application.

***Key Features of Hugging Face Spaces**:*

- Provides an easy way to host and deploy machine learning applications.
- Supports web applications built with Streamlit, Gradio, and Flask.
- Offers free hosting with the ability to scale up as needed.

h) Gunicorn

Gunicorn (Green Unicorn) is a Python WSGI HTTP server for running web applications in production environments. In this project, Gunicorn is used to serve the Flask application in a multi-threaded environment, ensuring the application can handle multiple requests efficiently.

***Key Features of Gunicorn**:*

- Handles multiple simultaneous connections efficiently.
- Supports various worker configurations to optimize performance based on server resources.
- Lightweight and easy to integrate with Flask applications.

Chapter- 2 Proposed System

2.1 Proposed System

The proposed **Student Performance Indicator** system is designed to predict a student's subject percentage based on various input scores, offering educational institutions, teachers, and students a powerful tool to analyze and improve academic performance. The system leverages machine learning algorithms to forecast outcomes, enabling proactive intervention and personalized learning strategies. The proposed solution aims to provide an intuitive, scalable, and efficient system that can be accessed through a web-based application.

Key Objectives of the Proposed System:

Accurate Prediction of Student Performance:

- The system will predict a student's percentage in specific subjects by analyzing input features such as previous grades, study hours, attendance, and other academic factors. The prediction is made using a machine learning model trained on historical data to identify patterns and trends.

Personalized Learning Insights:

- Based on predicted performance, the system will offer personalized insights, helping students understand which subjects they are likely to excel in or struggle with. These insights can guide students in focusing their efforts on subjects that need improvement.

User-Friendly Web Interface:

- The system will feature a user-friendly web application built using the Flask framework. This interface allows students, teachers, and administrators to input data, view predictions, and analyze performance metrics with ease.

Real-Time and Scalable Predictions:

- The system will allow users to get real-time predictions by simply entering a student’s data into the web interface. The platform is designed to be scalable, capable of handling data from multiple students, and can be deployed both locally and in cloud environments.

Comprehensive Data Analysis:

- The system will include tools for analyzing the dataset, providing visualizations and insights into overall academic performance trends. This can help schools and teachers make data-driven decisions to improve student outcomes at both individual and group levels.

Seamless Deployment with Docker:

- The system is containerized using Docker, ensuring that it runs smoothly across different environments. This allows the application to be deployed in a cloud environment (such as Hugging Face Spaces) or run locally without compatibility issues.

Flexibility for Future Expansion:

- The system is designed with flexibility in mind, allowing for future integration of additional features such as more complex machine learning models, larger datasets, or advanced predictive analytics for different educational levels.

How the System Works:

Data Input:

- Users (students, teachers, or administrators) can input relevant academic data such as previous scores, study habits, attendance records, and other metrics into the system through the web interface.

Prediction Process:

- The machine learning model, built using **Linear Regression** (or other algorithms), processes the input data and predicts the student's percentage in specific subjects. This is achieved by analyzing relationships between input variables and past performance trends.

Output and Analysis:

- The system will display the predicted percentage along with additional performance insights. Teachers can use these insights to guide their teaching strategies, while students can use them to improve study habits and focus on weak areas.

Deployment:

- The system is hosted using **Hugging Face Spaces**, which provides a scalable cloud environment. Additionally, the project is containerized using Docker, ensuring that it can be easily deployed and maintained across various platforms.

Advantages of the Proposed System:

- **Improved Academic Performance**: By predicting student performance early, teachers and institutions can provide targeted interventions to improve student outcomes.
- **Data-Driven Insights**: The system offers valuable data analysis that can help educators make informed decisions regarding curriculum design and resource allocation.
- **Accessibility**: The system is accessible through a web interface, making it easy for students, teachers, and administrators to use, regardless of technical expertise.

- **Flexibility and Scalability**: The system can handle multiple students and datasets and can be easily scaled to accommodate more users and features as needed.

2.2 Objectives of the System

The **Student Performance Indicator** system is designed to serve as a powerful tool for educational institutions, teachers, and students by providing predictions and insights into academic performance. The system's objectives are focused on improving the understanding of student outcomes and facilitating data-driven decision-making to enhance overall educational effectiveness. The key objectives are outlined below:

Predict Student Performance with High Accuracy

- The primary objective of the system is to predict a student's subject-wise percentage based on input features such as previous exam scores, attendance, study habits, and other academic factors. Using machine learning models, the system aims to deliver accurate predictions that can help students and teachers understand potential outcomes and take proactive measures.

Enable Early Detection of Academic Struggles

- By predicting a student's performance before final exams or evaluations, the system enables early identification of students who might struggle with certain subjects. This early detection allows teachers to intervene and offer additional support to improve student outcomes in critical areas.

Provide Personalized Learning Insights

- The system will offer personalized insights based on the predicted results. Students will be able to see which subjects they are excelling in and which areas need improvement. Teachers can use this information to create customized learning plans and help students focus on their weaknesses.

Offer a User-Friendly and Accessible Interface

- The system will provide a simple and intuitive web-based interface that allows users (students, teachers, or administrators) to input data, view predictions, and analyze performance trends easily. This objective is to ensure that the system can be used without requiring advanced technical skills.

Facilitate Data-Driven Decision Making

- The system's predictive insights allow educators to make informed, data-driven decisions. Teachers can use the system to assess which students need additional attention, and schools can use the aggregated data to adjust curricula, allocate resources, or design targeted interventions for improving academic performance.

2.3 User Requirements of the System

Student Requirements

- **Account Creation**: Students can create and manage their accounts.
- **Data Input**: Students can input academic data (e.g., previous scores, attendance).
- **Performance Prediction**: Students can view predicted performance in subjects.
- **Personalized Insights**: The system offers suggestions for improvement.
- **User-Friendly Interface**: Easy-to-navigate web interface.

Teacher Requirements

- **Data Management**: Teachers can manage and view student data.
- **Performance Analysis**: Teachers can analyze trends and identify patterns.
- **Actionable Insights**: Recommendations for student interventions.
- **User-Friendly Interface**: Easy access to data and insights.

Administrator Requirements

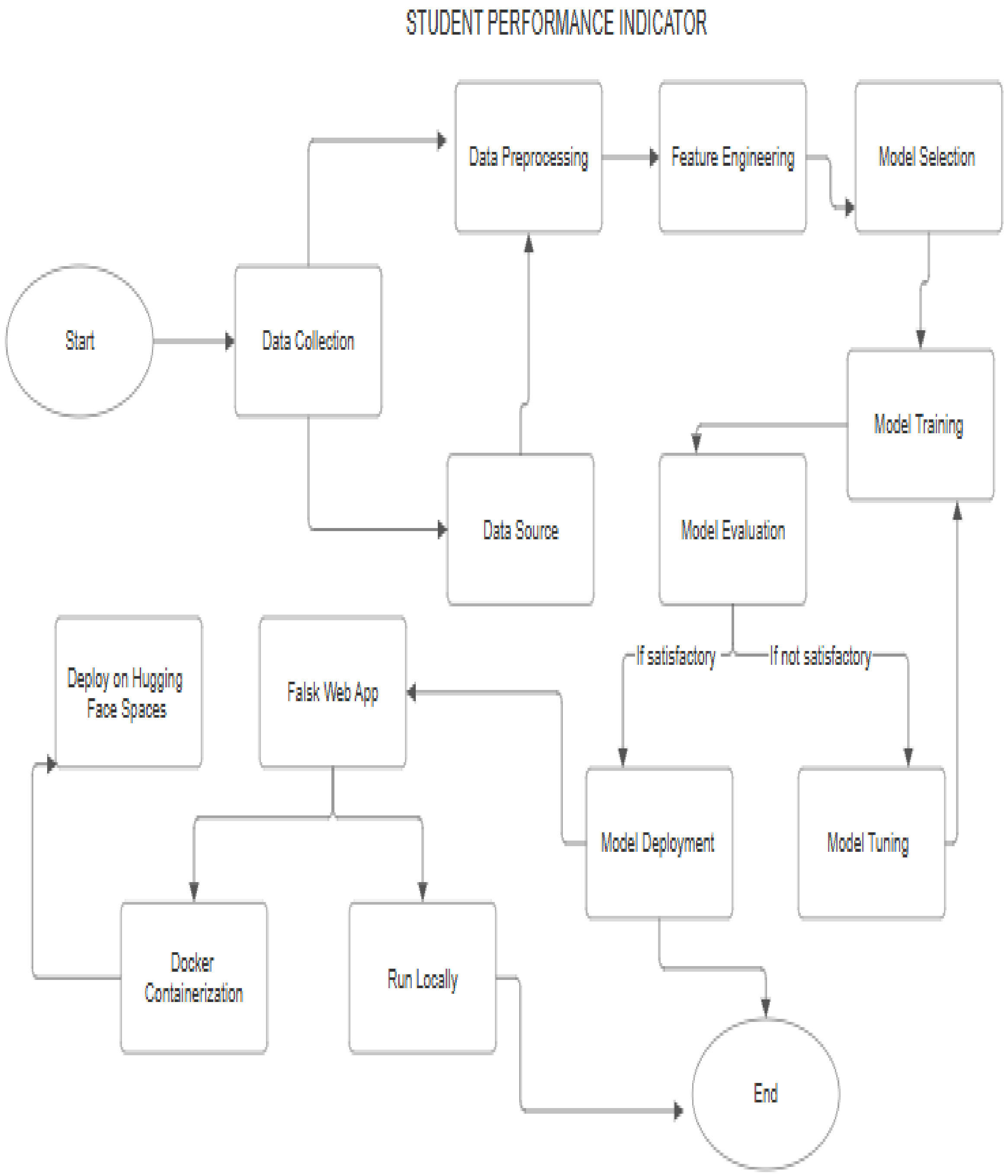
- **System Configuration**: Administrators can configure and manage user roles.
- **Data Security**: Ensure privacy and security of student data.
- **User Management**: Create/manage student and teacher accounts.
- **Report Generation**: Generate performance reports and system analytics.
- **System Monitoring**: Monitor system performance and handle issues.

General System Requirements

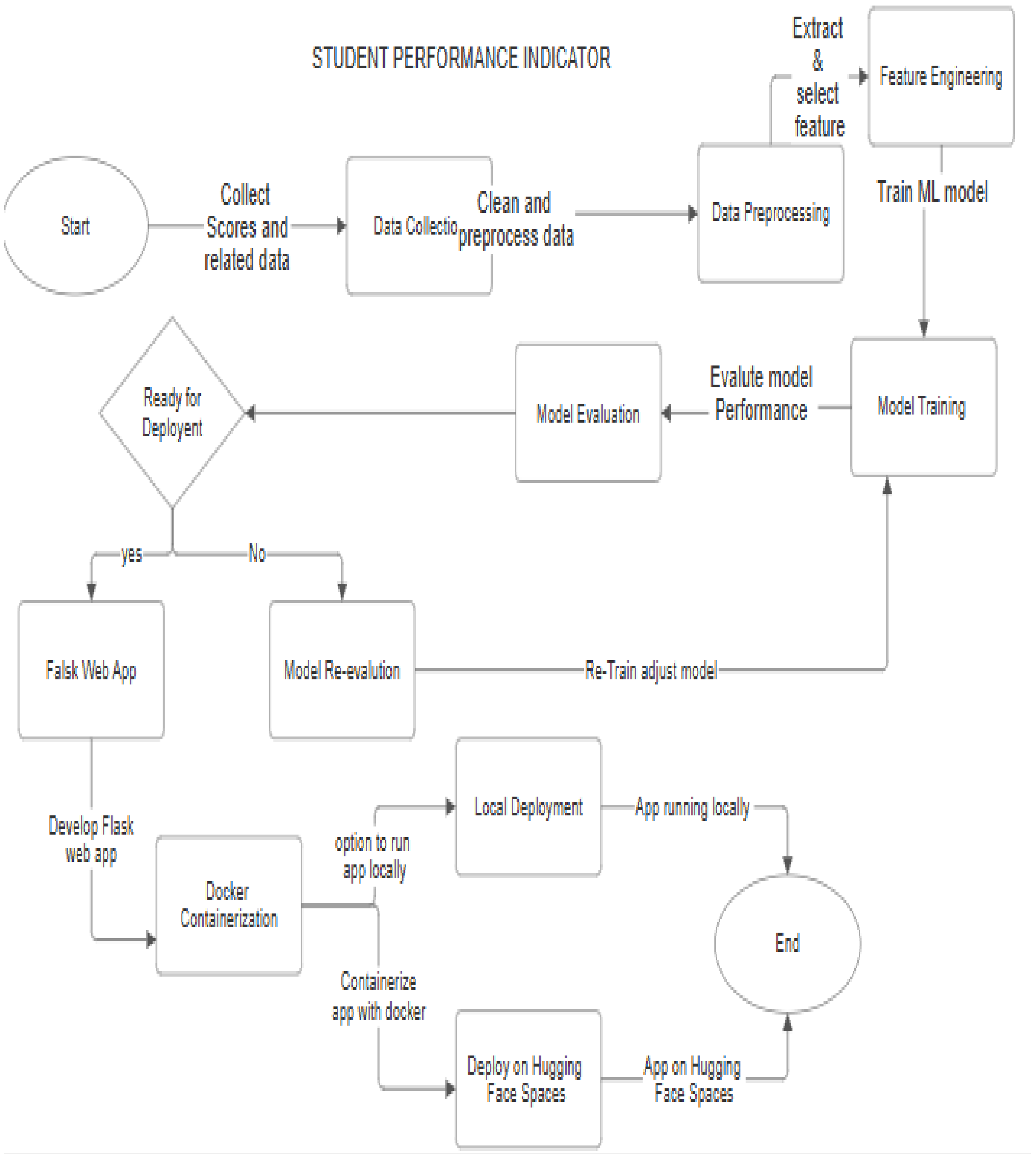
- **Cloud Deployment**: Hosted on a cloud platform (e.g., Hugging Face Spaces).
- **Containerization**: Docker for deployment consistency.
- **Secure Authentication**: Username/password and two-factor authentication.
- **Mobile Compatibility**: Accessible on mobile devices.
- **Performance and Scalability**: Handles large user bases and data.

Chapter-3 Analysis & Design

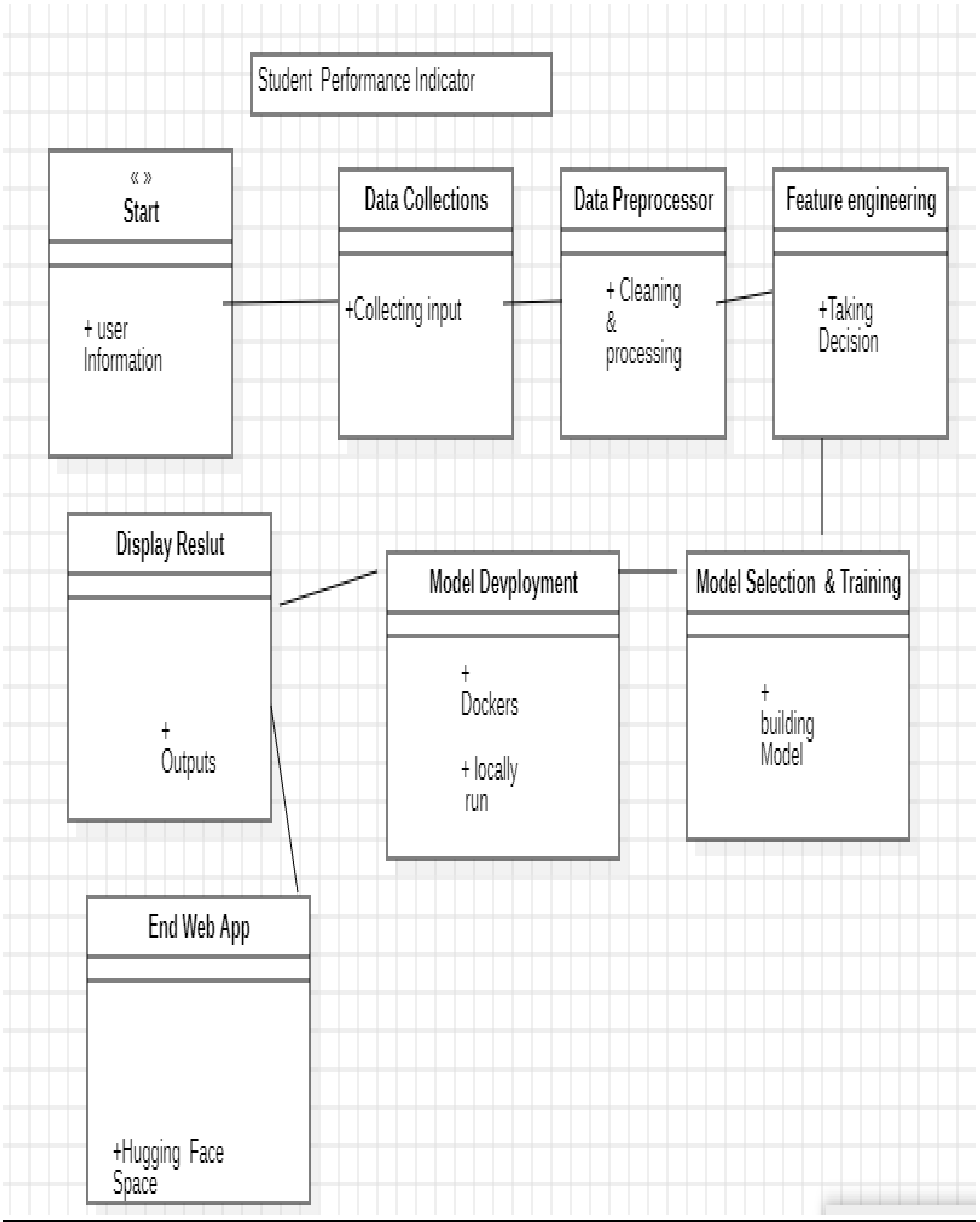
3.1 DFD



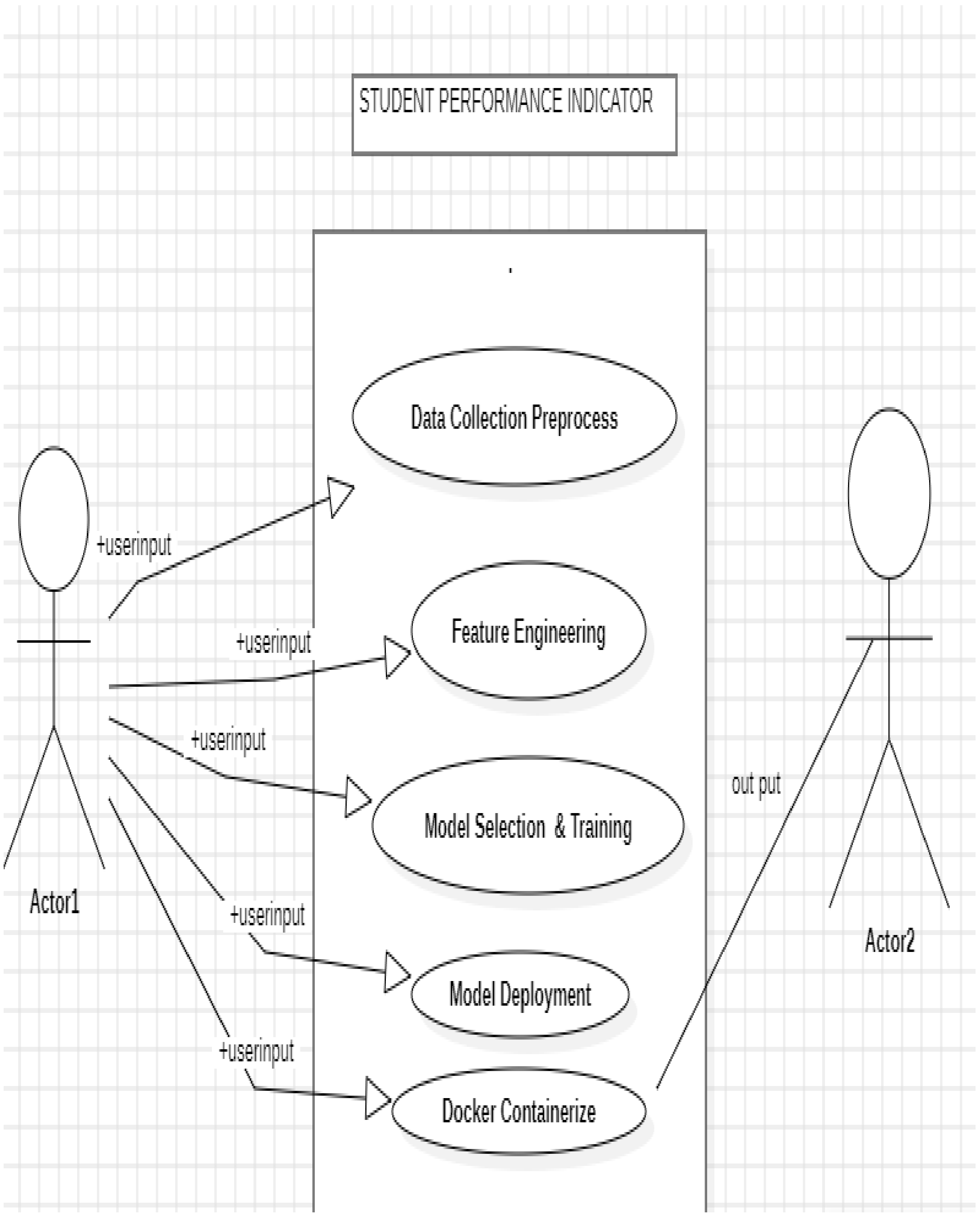
3.2 ER Diagram



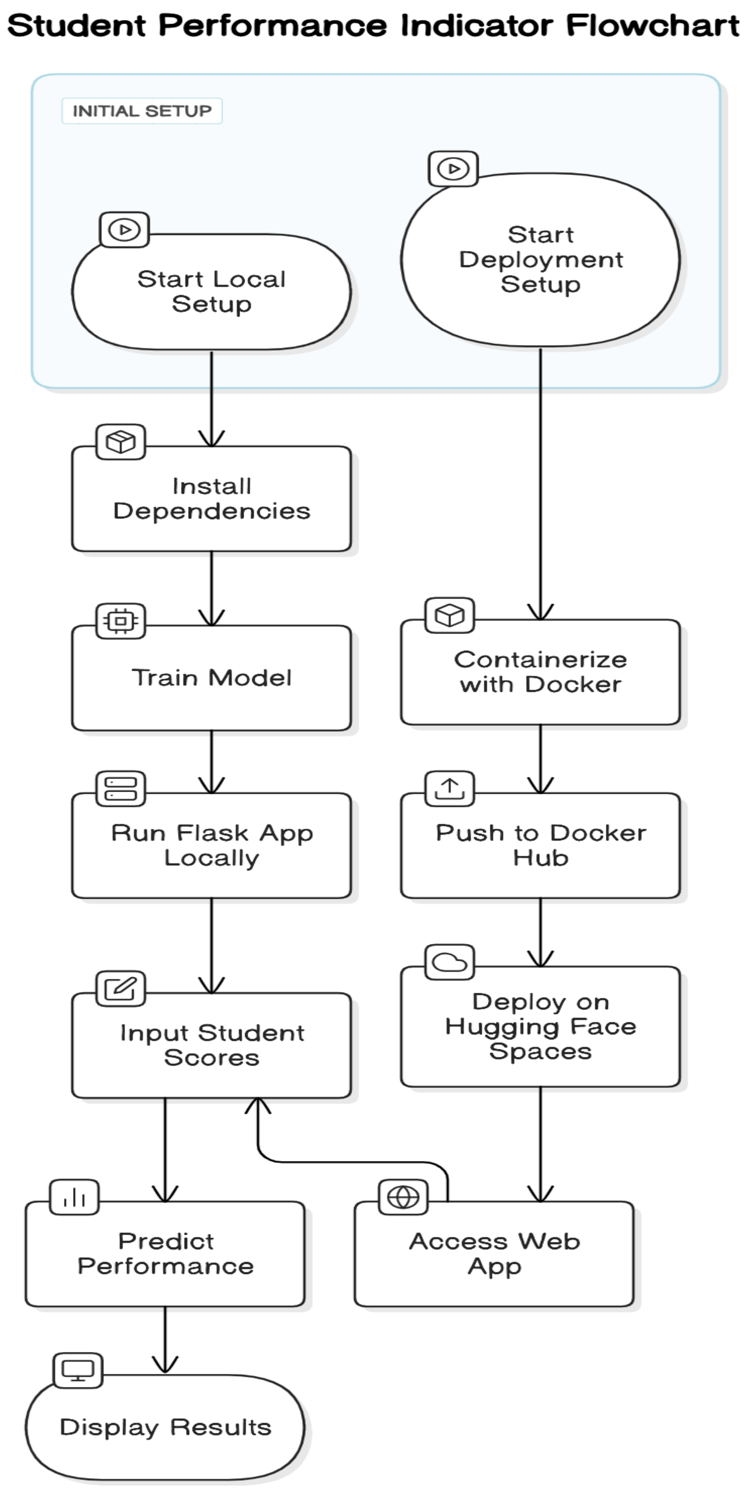
3.3 Class Diagram



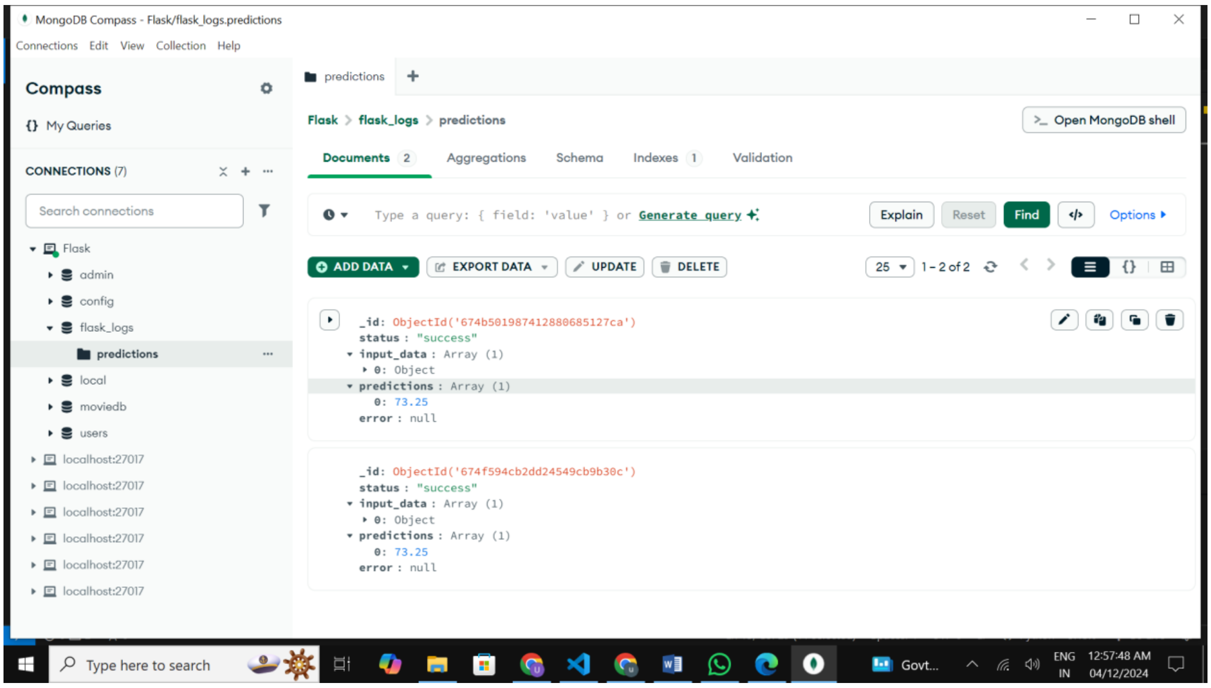
3.4 Use Case Diagram



3.5 Activity Diagram



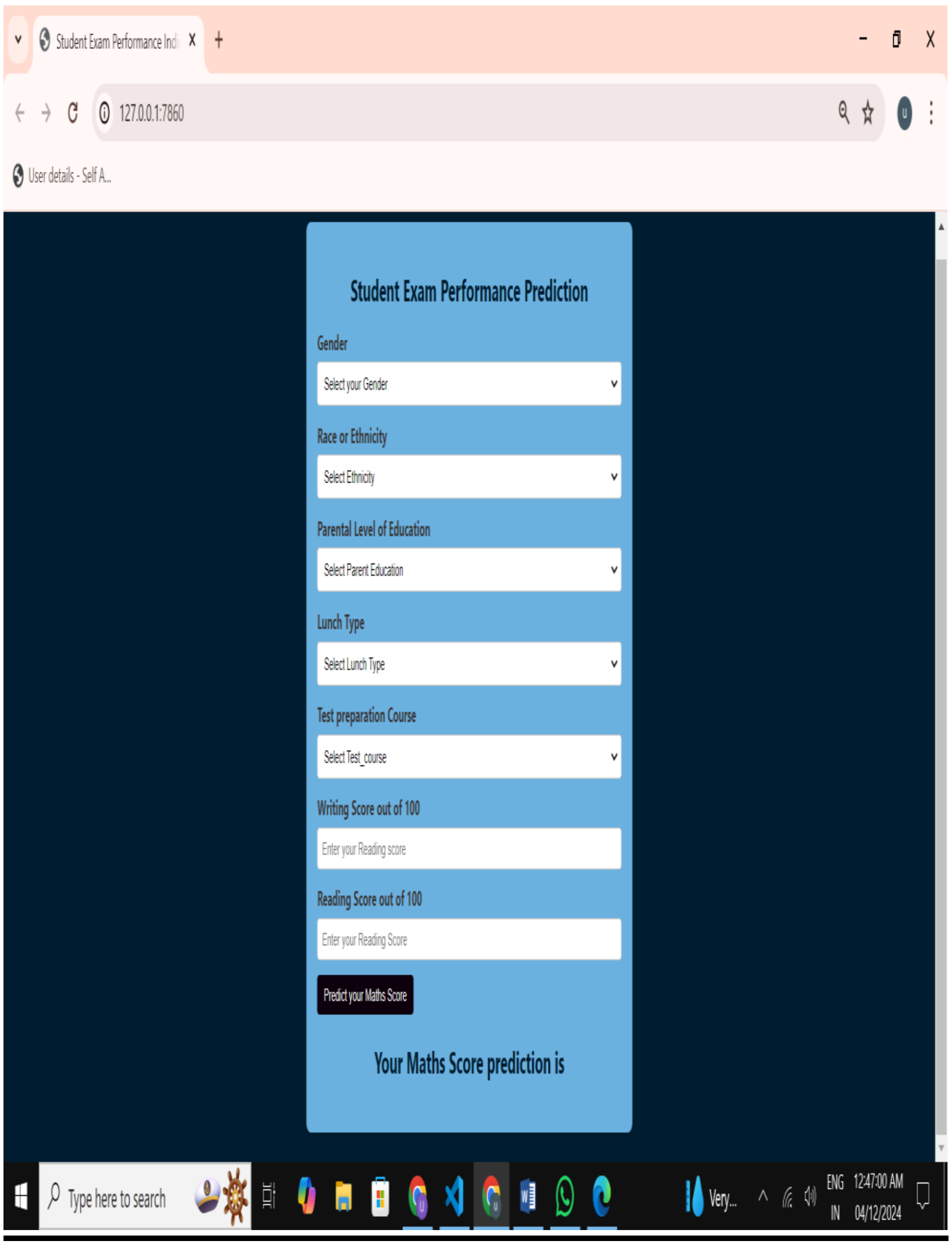
3.6 Table specification



```
_id: ObjectId('674b501987412880685127ca')
status : "success"
input_data : Array (1)
  0: Object
predictions : Array (1)
  0: 73.25
error : null

_id: ObjectId('674f594cb2dd24549cb9b30c')
status : "success"
input_data : Array (1)
  0: Object
predictions : Array (1)
  0: 73.25
error : null
```

Chapter-4 User Manual



Student Exam Performance Prediction

Gender

Select your Gender

Race or Ethnicity

Select Ethnicity

Parental Level of Education

Select Parent Education

Lunch Type

Select Lunch Type

Test preparation Course

Select Test_course

Writing Score out of 100

Enter your Reading score

Reading Score out of 100

Enter your Reading Score

Your Maths Score prediction is

4.2 Limitations of the System

While the **Student Performance Indicator** system provides valuable insights into student performance, there are certain limitations to be aware of:

Limited Data Input

- The system relies on the accuracy and completeness of the input data, such as past scores, attendance, and study habits. Missing or incomplete data can impact the accuracy of the performance predictions.

Dependency on Historical Data

- The machine learning model is trained on historical data, which may not always reflect the most up-to-date or comprehensive trends. If the data used for training is outdated or biased, the predictions may not be as accurate.

Subjectivity in Data

- Some inputs, such as study habits or participation in class, are subjective and can be difficult to quantify accurately. This may lead to variations in the predictions, as these factors are not always consistent or easy to measure.

Limited to Predicting Academic Performance

- The system focuses primarily on predicting academic performance based on past data. It does not account for other factors that may affect student performance, such as emotional well-being, personal issues, or external influences like family support.

Complexity of Machine Learning Models

- While the system uses machine learning models like Linear Regression, more complex models could provide better accuracy. However, these models require more data, computational resources, and may increase the complexity of the system.

Not a Replacement for Teacher Expertise

- The system provides predictions and insights, but it should not replace the judgment and expertise of teachers. Teachers' personal insights and observations are still crucial in providing the right support to students.

Limited Scalability for Large Institutions

- While the system is scalable, large institutions with extensive student populations may face challenges in handling large datasets in real-time. Additional optimizations and infrastructure may be required for such scenarios.

4.3 Future Enhancements

The **Student Performance Indicator** system has the potential for several future enhancements to improve its functionality, accuracy, and user experience. Some of these enhancements include:

Integration with Learning Management Systems (LMS)

- Future versions of the system could be integrated with popular Learning Management Systems (LMS) like Moodle or Google Classroom. This would allow for automatic data syncing, including exam scores, assignments, and real-time performance metrics, reducing the need for manual data input.

Use of Advanced Machine Learning Models

- While the current system uses Linear Regression, more complex machine learning models such as Random Forests, Support Vector Machines (SVM), or Neural Networks could be implemented to improve the accuracy of performance predictions by capturing more complex patterns in the data.

Real-time Performance Tracking

- The system could evolve to provide real-time performance tracking and updates. By collecting continuous data from students, such as participation, assignments, or quizzes, it can provide ongoing performance predictions and suggest adjustments to learning strategies.

Multi-Language Support

- To increase accessibility, the system could be enhanced with multi-language support. This would allow students and teachers from diverse linguistic backgrounds to use the system in their preferred language, improving adoption across global educational institutions.

Personalized Learning Pathways

- The system could suggest personalized learning pathways based on students' predicted performance. For instance, if a student is predicted to struggle with a subject, the system could recommend specific resources, tutorials, or study plans to improve their understanding and performance.

Emotional and Behavioral Data Integration

- Incorporating emotional and behavioral data from surveys, sentiment analysis, or self-reports could enhance the model's prediction accuracy by considering factors beyond academic performance, such as a student's mental health or engagement level.

BIBLIOGRAPHY

- A foundational reference for machine learning algorithms and techniques used in the project.

Flask Documentation. (n.d.). *Flask Web Framework Documentation*. Retrieved from https://flask.palletsprojects.com/en/latest/

- The official Flask documentation, which guided the development of the web application used to deploy the student performance prediction system.

Docker Inc. (n.d.). *Docker: Develop, Share, and Run Any App, Anywhere*. Retrieved from https://www.docker.com/

- Information about Docker containerization techniques that helped with deploying the project efficiently.

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. New York: Springer.

- Provides an in-depth understanding of predictive modeling techniques used in the project for predicting student performance.

Hugging Face. (n.d.). *Hugging Face Spaces Documentation*. Retrieved from https://huggingface.co/docs

- Documentation on Hugging Face Spaces, which was used for deploying the machine learning model.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics.

- A reference on statistical methods and machine learning models relevant to performance prediction.

Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H.-T. (2012). *Learning from Data*. AMLBook.

- Covers foundational machine learning concepts that were useful in building and training the predictive models.

MongoDB Documentation. (n.d.). *MongoDB: NoSQL Database for Modern Applications*. Retrieved from https://www.mongodb.com/docs/

- Reference for managing and storing user data in a scalable, NoSQL database.

King, P., & Cox, C. (2011). *The Data Flow Diagram: A Practical Guide*. Wiley.

  - Guide for understanding and constructing Data Flow Diagrams, used in the system analysis and design phase of the project.

Müller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media.

- Provided practical insights into implementing machine learning models using Python, which were crucial for the project's development.

ChatGPT by OpenAI. (2024). *Personal Assistance and Technical Guidance*. Retrieved from https://openai.com/chatgpt

- Used for generating ideas, clarifying doubts, and receiving guidance throughout the project development process.

Google Search. (2024). *Information and Troubleshooting*. Retrieved from https://www.google.com/

- Used extensively for researching various technical aspects, troubleshooting issues, and learning new concepts related to machine learning, Flask, and Docker.

YouTube Tutorials. (2024). *Video Tutorials on Flask, Machine Learning, and Docker*. Retrieved from https://www.youtube.com/

- Various YouTube tutorials were consulted for step-by-step guidance on implementing Flask applications, deploying using Docker, and understanding machine learning concepts.

Stack Overflow. (n.d.). *Developer Community for Problem-Solving and Code Debugging*. Retrieved from https://stackoverflow.com/

ANNEXURE

App.py

```
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
import logging
from pymongo import MongoClient
from sklearn.preprocessing import StandardScaler
from src.pipeline.predict_pipeline import CustomData, PredictPipeline

application = Flask(__name__)
app = application

# MongoDB connection
client = MongoClient("mongodb://localhost:27017/")  # Replace with your
MongoDB URI
db = client['flask_logs']  # Database name
collection = db['predictions']  # Collection name


def log_to_db(data):
    """
    Inserts logging data into MongoDB.
    :param data: Dictionary containing log details.
    """
    try:
        collection.insert_one(data)
    except Exception as e:
        logging.error(f"Failed to log data to MongoDB: {str(e)}")


@app.route('/', methods=['GET', 'POST'])
def predict_datapoint():
    if request.method == 'GET':
        return render_template('home.html')
    else:
        try:
            # Gather input data
            data = CustomData(
                gender=request.form.get('gender'),
                race_ethnicity=request.form.get('ethnicity'),
                parental_level_of_education=request.form.get('parental_level_o
f_education'),
                lunch=request.form.get('lunch'),
                test_preparation_course=request.form.get('test_preparation_cou
rse'),
                reading_score=float(request.form.get('writing_score')),
                writing_score=float(request.form.get('reading_score'))
            )

            pred_df = data.get_data_as_data_frame()
            logging.info("Input DataFrame created.")

            # Prediction pipeline
            predict_pipeline = PredictPipeline()
            results = predict_pipeline.predict(pred_df)

            # Log successful prediction
            log_to_db({
                "status": "success",
                "input_data": pred_df.to_dict(orient='records'),
                "predictions": results.tolist(),
                "error": None
            })

            return render_template('home.html', results=results[0])

        except Exception as e:
            # Log error to MongoDB
            log_to_db({
                "status": "error",
                "input_data": None,
                "predictions": None,
                "error": str(e)
            })
            logging.error(f"Prediction error: {str(e)}")
            return render_template('home.html', results="An error occurred
during prediction.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.DEBUG)
    app.run(debug=True, port=7860, host="0.0.0.0")
```

application.py

```
from flask import Flask,request,render_template
import numpy as np
import pandas as pd

from sklearn.preprocessing import StandardScaler
from src.pipeline.predict_pipeline import CustomData,PredictPipeline

application=Flask(__name__)

app=application

## Route for a home page

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predictdata',methods=['GET','POST'])
def predict_datapoint():
    if request.method=='GET':
        return render_template('home.html')
    else:
        data=CustomData(
            gender=request.form.get('gender'),
            race_ethnicity=request.form.get('ethnicity'),
            parental_level_of_education=request.form.get('parental_level_of_ed
ucation'),
            lunch=request.form.get('lunch'),
            test_preparation_course=request.form.get('test_preparation_course'
),
            reading_score=float(request.form.get('writing_score')),
            writing_score=float(request.form.get('reading_score'))

        )
        pred_df=data.get_data_as_data_frame()
        print(pred_df)
        print("Before Prediction")

        predict_pipeline=PredictPipeline()
        print("Mid Prediction")
        results=predict_pipeline.predict(pred_df)
        print("after Prediction")
        return render_template('home.html',results=results[0])
```

```python
if __name__=="__main__":
    app.run(debug=True,port=7860,host="0.0.0.0")
```

Modules required

```
pandas
numpy
seaborn
matplotlib
scikit-learn==1.5.1
catboost
xgboost
Flask
dill
pymongo
-e .
```

Dockerfile

```dockerfile
FROM python:3.9-slim-buster

# Set working directory
WORKDIR /app

# Copy your application code

COPY . .
# Copy requirements.txt and install dependencies

COPY requirements.txt ./
RUN pip3 install -r requirements.txt
RUN pip3 install gunicorn

RUN mkdir -p /app/logs && chmod 755 /app/logs
# Expose port that Flask app uses

EXPOSE 7860

# Set the entrypoint to run your Flask app

ENTRYPOINT ["python", "app.py"]
```

Setup.py

```
from setuptools import find_packages,setup
from typing import List

HYPEN_E_DOT='-e .'
def get_requirements(file_path:str)->List[str]:
    '''
    this function will return the list of requirements
    '''
    requirements=[]
    with open(file_path) as file_obj:
        requirements=file_obj.readlines()
        requirements=[req.replace("\n","") for req in requirements]

        if HYPEN_E_DOT in requirements:
            requirements.remove(HYPEN_E_DOT)

    return requirements

setup(
name='mlproject',
version='0.0.1',
author='ujwala',
author_email='ujwalakusma26@gmail.com',
packages=find_packages(),
install_requires=get_requirements('requirements.txt')

)
```

Exception.py

```
import sys
import logging

def error_message_detail(error,error_detail:sys):
    _,_,exc_tb=error_detail.exc_info()
    file_name=exc_tb.tb_frame.f_code.co_filename
    error_message="Error occured in python script name [{0}] line number [{1}]
error message[{2}]".format(
     file_name,exc_tb.tb_lineno,str(error))

    return error_message



class CustomException(Exception):
    def __init__(self,error_message,error_detail:sys):
        super().__init__(error_message)
        self.error_message=error_message_detail(error_message,error_detail=err
or_detail)

    def __str__(self):
        return self.error_message

if __name__=='__main__':
    try:
        a=1/0
    except Exception as e:
        logging.info('logging tarted')
        raise CustomException(e,sys)
```

logger.py

```
import logging
import os
from datetime import datetime

LOG_FILE=f"{datetime.now().strftime('%m_%d_%Y_%H_%M_%S')}.log"
logs_path=os.path.join(os.getcwd(),"logs",LOG_FILE)
os.makedirs(logs_path,exist_ok=True)

LOG_FILE_PATH=os.path.join(logs_path,LOG_FILE)

logging.basicConfig(
    filename=LOG_FILE_PATH,
    format="[ %(asctime)s ] %(lineno)d %(name)s - %(levelname)s -
%(message)s",
    level=logging.INFO,


)
if __name__=='__main__':
    logging.info('logging tarted')
```

utils.py

```
import os
import sys

import numpy as np
import pandas as pd
import dill
from sklearn.metrics import r2_score
from sklearn.model_selection import GridSearchCV

from src.exception import CustomException

def save_object(file_path, obj):
    try:
        dir_path = os.path.dirname(file_path)

        os.makedirs(dir_path, exist_ok=True)

        with open(file_path, "wb") as file_obj:
            dill.dump(obj, file_obj)

    except Exception as e:
        raise CustomException(e, sys)

def evaluate_models(X_train,y_train,X_test,y_test,models,param):
    try:
        report={}

        for i in range(len(list(models))):
            model=list(models.values())[i]
            para=param[list(models.keys())[i]]

            gs = GridSearchCV(model,para,cv=3)
            gs.fit(X_train,y_train)

            model.set_params(**gs.best_params_)
            model.fit(X_train,y_train)

            y_train_pred=model.predict(X_train)

            y_test_pred=model.predict(X_test)

            train_model_score=r2_score(y_train,y_train_pred)

            test_model_score=r2_score(y_test,y_test_pred)
```

```python
                report[list(models.keys())[i]]=test_model_score

            return report

    except Exception as e:
        raise CustomException(e, sys)

def load_object(file_path):
    try:
        with open(file_path, "rb") as file_obj:
            return dill.load(file_obj)

    except Exception as e:
        raise CustomException(e, sys)
```

home.html

```html
<html>
<head>
    <title>Student Exam Performance Indicator</title>
    <link rel="stylesheet" type="text/css" href="/static/style.css">
</head>

<body>
    <div class="all">
    <div class="login">
       <form action="{{ url_for('predict_datapoint')}}" method="post">
        <h1>Student Exam Performance Prediction</h1>

        <div class="mb-3">
            <label class="form-label">Gender</label>
            <select class="form-control" name="gender" placeholder="Enter you
Gender" required>
                <option class="placeholder" selected disabled value="">Select
your Gender</option>
                <option value="male">
                    Male
                </option>
                <option value="female">
                    Female
                </option>
            </select>
        </div>
        <div class="mb-3">
            <label class="form-label">Race or Ethnicity</label>
            <select class="form-control" name="ethnicity" placeholder="Enter
you ethnicity" required>
                <option class="placeholder" selected disabled value="">Select
Ethnicity</option>
                <option value="group A">
                    Group A
                </option>
                <option value="group B">
                    Group B
                </option>
                <option value="group C">
                    Group C
                </option>
                <option value="group D">
                    Group D
                </option>
                <option value="group E">
                    Group E
                </option>
            </select>
        </div>
        <div class="mb-3">
            <label class="form-label">Parental Level of Education</label>
            <select class="form-control" name="parental_level_of_education"
                placeholder="Enter you Parent Education" required>
                <option class="placeholder" selected disabled value="">Select
Parent Education</option>
                <option value="associate's degree">
                    associate's degree
                </option>
                <option value="bachelor's degree">
                    bachelor's degree
                </option>
                <option value="high school">
                    high school
                </option>
                <option value="master's degree">
                    master's degree
                </option>
                <option value="some college">
                    some college
                </option>
                <option value="some high school">
                    some high school
                </option>
            </select>
        </div>
        <div class="mb-3">
            <label class="form-label">Lunch Type</label>
            <select class="form-control" name="lunch" placeholder="Enter you
Lunch" required>
                <option class="placeholder" selected disabled value="">Select
Lunch Type</option>
                <option value="free/reduced">
                    free/reduced
                </option>
                <option value="standard">
                    standard
                </option>
            </select>
        </div>
        <div class="mb-3">
            <label class="form-label">Test preparation Course</label>
            <select class="form-control" name="test_preparation_course"
placeholder="Enter you Course"
                required>
                <option class="placeholder" selected disabled value="">Select
Test_course</option>
                <option value="none">
                    None
                </option>
                <option value="completed">
                    Completed
                </option>
            </select>
        </div>
        <div class="mb-3">
            <label class="form-label">Writing Score out of 100</label>
            <input class="form-control" type="number" name="reading_score"
                placeholder="Enter your Reading score" min='0' max='100' />
        </div>
        <div class="mb-3">
            <label class="form-label">Reading Score out of 100</label>
            <input class="form-control" type="number" name="writing_score"
                placeholder="Enter your Reading Score" min='0' max='100' />
        </div>
        <div class="mb-3">
            <input class="btn btn-primary" type="submit" value="Predict your
Maths Score" required />
        </div>
    </form>
    <h2>
        Your Maths Score  prediction is {{results}}
    </h2>
    </div>
   <body>
</html>
```

Style.css

```
/* General Styles */
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background-color: #001C30;
}

.all {
    background-color: #6bb1e0;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    border-radius: 8px;
    padding: 20px;
    width: 550px; /* Increased width */
    max-width: 90%; /* Ensures it fits on smaller screens */
    color: #393737;
}

h1 {
    text-align: center;
    color: #001C30;
    margin-bottom: 15px;
    font-size: 1.5rem;
}

/* Form Styles */
.form-label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
}

.form-control, select {
    width: 100%;
    padding: 8px; /* Reduced padding */
    margin-bottom: 12px; /* Reduced margin */
    border: 1px solid #ccc;
```

```css
    border-radius: 4px;
    box-sizing: border-box;
}

.btn-primary {
    background-color: #0c010c;
    color: #ffffff;
    padding: 8px 12px; /* Reduced padding */
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.btn-primary:hover {
    background-color: #045c91;
}

/* Prediction Result */
h2 {
    text-align: center;
    color: #001C30;
    margin-top: 20px;
}
```