



**Dr. D.Y. Patil School of MCA**

*Charoli (BK), PUNE- 412105*



**SAVITRIBAI PHULE PUNE UNIVERSITY  
MASTER OF COMPUTER APPLICATION**

**Project Report on**

**ADMISSION PREDICTION APPLICATION**

**BY**

**NIKITA SINHA**

**Seat no:13113**

**Under The Guidance Of**

**PROF. MR. MOHAMMAD ALI**

**Class: MCA-II(Sem-IV)**

**Year: 2024-2025**



# Dr D Y PATIL SCHOOL OF MCA

(Approved by AICTE, New Delhi Recognized by Govt. of Maharashtra, Affiliated to Savitribai Phule Pune University)

AISHE Code: C-45873 DTE Code: MC6201 SPPU PUN Code: IMMP019330



Date: ----/ ----/2025

## Certificate

This is to certify that Ms. **NIKITA SINHA**, Seat No.**13113**, a student of **Dr D Y Patil School of MCA**, has successfully completed the project entitled “**Admission Prediction Application**” in partial fulfillment of the requirements for the **Master of Computer Applications (MCA)** degree during the academic year **2024–2025**.

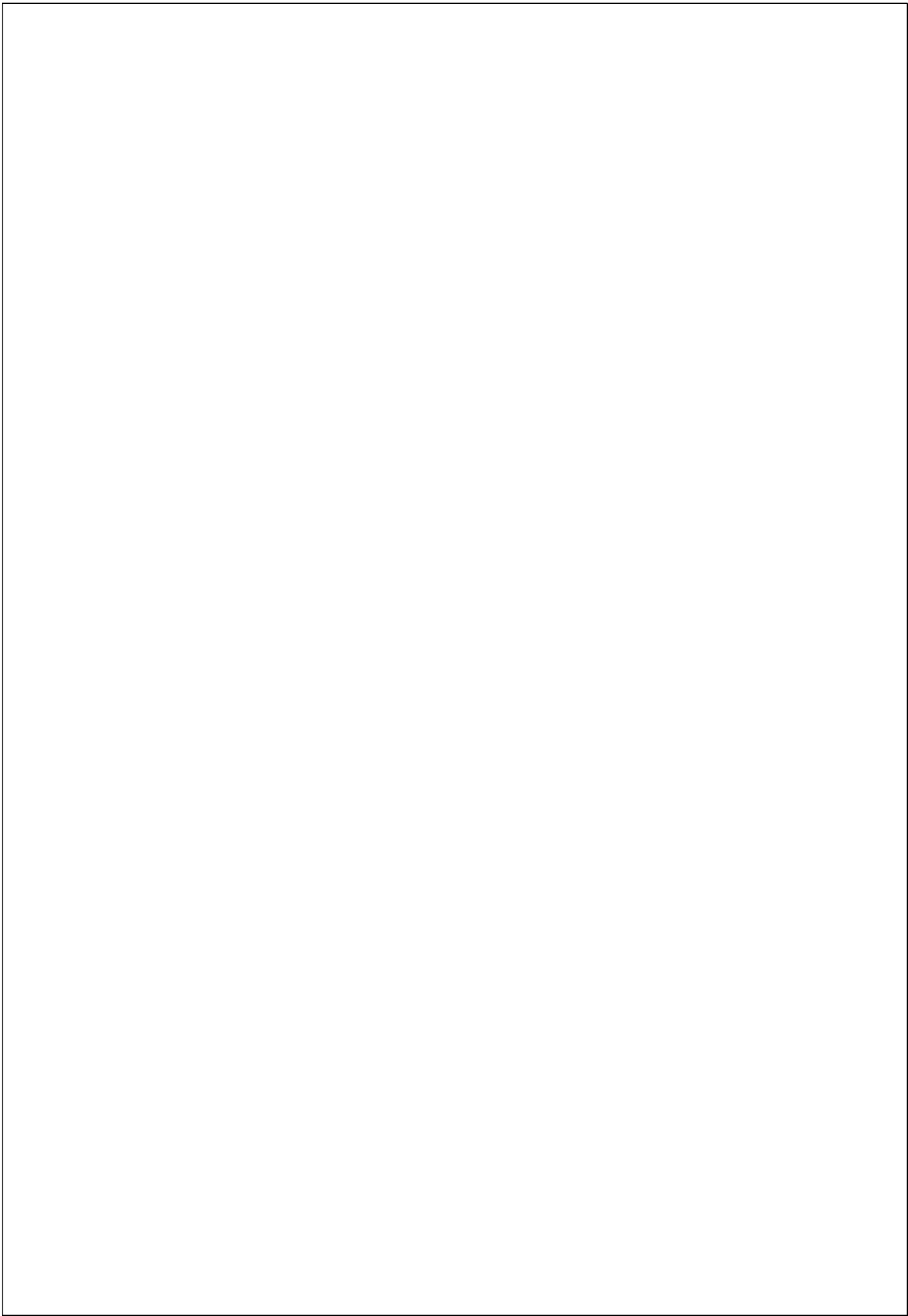
**Prof. Mr. Mohammad Ali**  
Project Guide

**Prof. Sapna Chavan**  
HOD

**Dr. E. B. Khedkar**  
Director

**Examiner 1: -----**

**Examiner 2: -----**



## AKCNOWLEDGMENT

I acknowledge to all those who have been so helpful in my academic project work. Nevertheless, I have made an effort through this report to express my deepest gratitude to all those who have given their precious time, skill, knowledge, valuable advice and guidance and facilities.

At the very outset I take opportunity to express my deepest gratitude and thanks to Director of our College **Dr. E. B. Khedkar** and **Prof. Sapna Chavan** HOD and other teaching and non-teaching staff for their useful guidance during the completion of this project Report.

I am highly obliged to **Prof. Mr. Mohammad Ali** for his valuable guidance and encouragement given to complete the project. Her/His guidance has certainly helped me to simplify the difficulties and finalize the system effectively. Last but not the least I thank to my almighty God, Parents and friends for their constant support to me in all aspects during the course of work.

Thank You,

Nikita Sinha

MCA–II(Sem-IV)

Dr DY Patil School of MCA

## INDEX

Serial No.	Particulars	Page Number
1	Introduction	1
1.1	Company Profile	1
1.2	Existing System and Need for system	2-3
1.3	Scope of System	4
1.4	Operating Environment - Hardware and Software	5
1.5	Brief Description of Technology Used	5
2	Proposed System	6
2.1	Study of Similar Systems	6-7
2.2	Feasibility Study	6-7
2.3	Objectives of Proposed System	7-8
3	Analysis and Design	9
3.1	System Requirements	9-12
3.2	Entity Relationship Diagram (ERD)	13
3.3	Use Case Diagrams	14-15
3.4	Table Structure	16-18
3.5	Activity Diagram	19-20
3.6	Class Diagram	21
3.7	Deployment Diagram	22
3.8	Sequence Diagram	23
3.9	Module Hierarchy Diagram	24
3.10	Input and Output UI Screens	25-27

4	Coding	28
4.1	Algorithms	28-31
5	Testing	32
5.1	Test Strategy	32-37
5.2	Defect report	38-39
6	User Manual	40
7	Limitations of Proposed System	41
8	Proposed Enhancements	42
9	Conclusion	43
10	Bibliography	44

# INTRODUCTION

## 1.1 Company Profile

**Name: Radixile Technologies Pvt Ltd.**

Radixile Technologies is a Pune (India) based software company expertise Enterprise Application and IT infrastructure implementation for an organization with a technology and as a strategic alliance. Our focus is on automation and digitization of an organization for cost reduction and improve productivity and accuracy.

We are also in making online presence with best possible way to reach desired client. It may include website Design & Development, Android & iPhone Mobile App Development. Our passionate and agile team committed to deliver excellent, innovative software product completed with best possible standards. Enterprise resource planning (ERP) is a process used by companies to manage and integrate the important parts of their businesses. ERP software applications are important to companies because they help them implement resource planning by integrating all of the processes needed to run their companies with a single system. An ERP software system can also integrate planning, purchasing inventory, sales, marketing, finance, human resources, and more. Petrox (Petrol Pump Software) is based on international guideline of petroleum, natural gas and petrochemical industries operations. It also describes data quality management and assurance practices to provide guidance for the users. Petrox is comprehensive petrol pump software for petrol pump billing management, petrol pump inventory management, petrol pump accounts management, petrol pump lube stock management and overall activities of petrol pumps.

**Company Vision:** We are a place that defines Integrity, Innovation and Serenity. We believe in building long term relationships with our clients and to do what, we keep them happy by providing services. We also believe in transparency with our valued clients.

## 1.2 Existing System and need for System

**Company Services Mission:** The current systems for predicting university admissions typically involve a combination of manual and automated processes. These systems often rely on a variety of factors, including standardized test scores (GRE and TOEFL), undergraduate GPA, letters of recommendation, personal statements, and relevant work experience. The admission committees at universities evaluate these criteria to make their decisions.

**University Admission Portals:** Most universities have their own online portals where applicants submit their documents and scores. These portals have basic features for sorting and filtering applications.

**Consultancy Firms:** Many students use educational consultancies that provide personalized guidance on the application process, helping to optimize their chances of admission based on historical data and expert knowledge.

**Drawbacks Of Existing Systems:**

**Lack of Standardization:** Different universities have varying criteria and weights for GRE, TOEFL scores, and other application components. This lack of standardization makes it difficult to develop a universally accurate prediction model.

**Manual Bias:** The human element in the admission process can introduce biases. Admissions officers might unconsciously favor certain profiles over others, leading to inconsistent decision-making.

**Time-Consuming:** The process of manually reviewing each application is time-consuming and resource-intensive. This can delay decisions and reduce the overall efficiency of the admissions process.



## **Need of System**

The primary purpose of developing a prediction system for university admissions based on GRE and TOEFL scores is to enhance the efficiency, transparency, and accuracy of the admissions process for both applicants and university administrators. Below are the key reasons highlighting the need for such a system:

**Data-Driven Decision Making:** By leveraging machine learning algorithms, the proposed system can analyze vast amounts of historical admissions data to identify patterns and trends. This data-driven approach helps in making more informed and objective decisions, reducing the reliance on subjective judgment.

**Increased Transparency:** A predictive system can provide clear insights into how different components of an application (such as GRE and TOEFL scores) influence the likelihood of admission. This transparency helps applicants understand their chances better and enables them to improve their profiles accordingly.

**Efficiency and Time-Saving:** Manual review of thousands of applications is a labor-intensive process. An automated prediction system can quickly filter and rank applications based on their likelihood of success, allowing admissions committees to focus their time and efforts on the most promising candidates.

**Fairness and Consistency:** By standardizing the evaluation criteria and minimizing human biases, the system ensures a fairer and more consistent assessment of all applicants. This reduces the potential for favoritism and ensures that all candidates are judged based on the same standards.

**Personalized Feedback for Applicants:** The system can provide personalized feedback to applicants, highlighting their strengths and areas for improvement. This feedback can be invaluable for applicants in refining their applications or enhancing their profiles for future admissions cycles.

### 1.3 Scope of System

The scope of this project encompasses the development, implementation, and deployment of a predictive system designed to assess the likelihood of university admissions based on key applicant attributes. The system leverages machine learning techniques to provide accurate and actionable predictions, offering significant benefits to both applicants and admissions committees.

The specific areas covered within the scope of the system include: Data Collection and Integration, Data Preprocessing and Analysis, Feature Selection and Engineering, Model Development, Model Optimization and Validation, Prediction and User Interaction.

The scope of the Admission Prediction system involves developing a data-driven model to forecast the likelihood of a student being admitted into a university or academic program. By analyzing historical data such as GRE scores, GPA, work or research experience, and recommendation strength, the system can identify patterns that influence admission outcomes. This model assists academic institutions in making more consistent and unbiased decisions during the admission process. It streamlines the evaluation of large volumes of applications, reducing manual effort and processing time.

For students, the system provides valuable insights into their admission chances, helping them make informed choices about where to apply. It can also be used to guide students on areas of improvement, increasing their chances of acceptance in future applications. Universities can use the results to prioritize outreach and scholarships for high-potential applicants. Additionally, the model supports fairness and transparency by minimizing human subjectivity.

The prediction tool is scalable and adaptable to various institutions and programs. It can be continually improved with new data to enhance prediction accuracy. Overall, the Admission Prediction system bridges the gap between applicants and academics.

## **1.4 Operating Environment - Hardware and Software**

### **HARDWARE SPECIFICATIONS**

Client Side: RAM Minimum 8GB and above

HARD DISK Minimum 500 GB

PROCESSOR Intel I3

Server Side: RAM 16 GB and above

HARDDISK 512 GB

PROCESSOR Intel I3

### **SOFTWARE SPECIFICATIONS**

Client-Side: Operating System Windows

Web Browser Google Chrome/Internet Explorer version 8.0 or higher

Server-Side: Operating System Windows 10

Database Used: MySQL

Programming Language: HTML, CSS, Python IDE: Visual Studio Code

Libraries used Pandas, Numpy, Seaborn, matplotlib, Sklearn, LinearRegression, train\_test\_split, RandomForestRegressor, DecisionTreeRegressor

## **1.5 Brief Description of Technology Used**

The project solution proposed to take the required input of user from the created interface and process all the provided data to meet the requirements of the machine learning model and finally display the output saying so and so amount is the predicted cost.

The Technology used in the project is Data Science (Machine Learning Algorithms) to predict the fare flight based on the historical data.

Python 3.9 is used as the programming language and frame works like Numpy, pandas, Sklearn and other modules for building the model. For visualizations seaborn and parts of matplotlib are being used.

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

## **PROPOSED SYSTEM**

### **2.1 STUDY OF SIMILAR SYSTEMS**

Several existing systems have been developed using machine learning to predict student admission outcomes based on academic and demographic features.

These systems provide valuable insights into which models work best under conditions and that Many university admission predictors use historical datasets with features like GRE scores, TOEFL scores, CGPA, and research experience.

These predictors often utilize supervised learning algorithms such as Linear Regression,

Logistic Regression, and Support Vector Machines (SVM).

Linear Regression models are easy to interpret and are commonly used for estimating the probability of admission. Decision Trees and Random Forests provide better accuracy, especially when handling non-linear relationships and noisy data.

Neural Networks are also applied for complex patterns but require more training data and computational resources K-Nearest Neighbors (KNN) has been explored in some systems for classification-based prediction tasks.

Naive Bayes classifiers have been used in simpler systems for probabilistic admission prediction SVMs offer robust classification performance in high-dimensional spaces, which is useful for diverse admission features.

Systems like GradSchoolPredictor and Yocket implement basic prediction models with user-friendly interfaces. Some systems allow users to compare their profiles against admitted student profiles at different universities.

These predictors also help users estimate safety, match, and dream universities based on scores. Most systems incorporate visualization features to help users understand the model's predictions.

A few advanced systems employ ensemble models to improve prediction reliability and generalization; the use of historical admission data ensures that predictions are based on real-world patterns and decisions. Some tools integrate feedback loops, where users can verify prediction accuracy post-admission and refine models.

Cloud-based deployment allows for real-time access and prediction at scale in many systems. Commonly faced challenges include data sparsity, missing values, and feature imbalance.

Best practices include data normalization, cross-validation, and outlier handling to improve model accuracy.

Ethical considerations like bias mitigation and fairness in prediction are also part of modern admission tools.

User data privacy and protection remain key priorities for trustworthy prediction systems.

These systems serve as an important reference and benchmarking foundation for developing new admission predictors.

Analyzing them helps identify strengths and gaps that your system can address through innovation and improvement.

## **2.2 FEASIBILITY STUDY**

The goal of this project is to develop a predictive model for university admissions based on key applicant attributes, such as GRE and TOEFL scores, undergraduate GPA, and other relevant factors. The system leverages machine learning techniques to provide accurate predictions of admission chances, enhancing both the efficiency and fairness of the admissions process.

### **Methodology**

The project utilizes a dataset from Kaggle containing 400 records with attributes such as GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA and Chance of Admit. The methodology involves several key steps:

**Data Cleaning and Preprocessing:** Handling missing values, converting datatypes and standardizing attributes to ensure a clean and usable dataset.

**Exploratory Data Analysis (EDA):** Analyzing data distribution and relationships between attributes to gain insights into the factors influencing admissions.

**Feature Selection and Engineering:** Identifying and selecting the most relevant features for the predictive model.

**Model Training and Evaluation:** Using the Linear Regression, Ridge Regression, Decision Tree Regressor and Random Forest Regressor Model to train the model and evaluate its performance using metrics such as R-square and adjusted R-square.

### **2.3 OBJECTIVES OF PROPOSED SYSTEM**

The proposed system aims to enhance the university admission prediction process by leveraging machine learning techniques to provide accurate, efficient, and fair assessments of applicants based on their GRE and TOEFL scores, along with other relevant attributes.

The primary objectives of the proposed system are:

**Accurate Prediction of Admission Chances:** To develop a machine learning model that accurately predicts the likelihood of admission to universities based on GRE and TOEFL scores, as well as other factors such as CGPA, university rating, SOP and LOR and to achieve a high level of prediction accuracy, with an aim to exceed 80% accuracy as indicated by R-square and adjusted R-square values.

**Streamlining the Admissions Process:** To create an automated system that reduces the time and effort required for university admissions committees to evaluate applications and to enable quick filtering and ranking of applicants, allowing committees to focus on the most promising candidates.

**Ensuring Fairness and Consistency:** To minimize human biases in the admissions process by using standardized, data-driven evaluation criteria and to ensure that all applicants are assessed based on the same objective standards, thereby promoting fairness and consistency.

**Enhancing Decision-Making for Universities:** To assist universities in making more informed decisions by providing data-driven insights into applicant profiles and to enable strategic planning by predicting enrolment trends and managing resources accordingly.

## ANALYSIS & DESIGN

### 3.1 SYSTEM REQUIREMENTS

#### Functional Requirements

The Admission Prediction System is designed to assist universities and students by leveraging historical data to predict the chances of admission based on various academic and personal indicators. The system must begin with robust data collection and loading functionality.

It should support importing datasets from trusted sources such as Kaggle or institutional databases. The system must ensure that key attributes—GRE Score, TOEFL Score, University Rating, SOP strength, LOR rating, CGPA, Research Experience (binary), and Chance of Admission—are present and correctly formatted. This step is critical for maintaining the integrity of all downstream processes. The system should validate that no essential fields are missing before proceeding.

Once data is loaded, data cleaning and preprocessing should be initiated. The system should identify and remove or correct inconsistencies in the dataset, such as the presence of special characters or unexpected null values. Categorical attributes should be converted into suitable numerical formats using techniques like label encoding or one-hot encoding where necessary. The system must handle missing values effectively—numerical attributes may be imputed with mean or median values, while categorical ones may be filled with mode or inferred values. Standardizing or normalizing numerical features is also necessary to ensure that all input features contribute equally to model training, particularly for distance-based algorithms like Support Vector Machines.

Following preprocessing, the system should perform Exploratory Data Analysis (EDA) to uncover patterns and distributions in the dataset. It should compute statistical summaries, including central tendencies and dispersion metrics for each feature. The system should generate visualizations such as histograms for distribution analysis, scatter plots to identify relationships between features, and box plots to detect outliers. Correlation heatmaps should also be included to visualize multicollinearity and feature importance relationships. These visualizations and summaries will not only guide further modeling but also offer users an intuitive understanding of how different

attributes influence the prediction outcome.

The next core requirement is feature selection and engineering. The system must identify the most relevant predictors for the target variable (Chance of Admission). This can be achieved through correlation analysis, recursive feature elimination, or tree-based feature importance scoring. Redundant, highly correlated, or noisy features must be removed to prevent overfitting and improve model performance. If necessary, the system should be capable of deriving new attributes from existing ones, such as interaction terms or normalized scores. This component ensures that the most predictive, cleanest, and compact dataset is used during training.

During model training and evaluation, the system must split the data into training and testing sets using stratified sampling to maintain proportional representation of the target variable. The system should support multiple algorithms, including Linear Regression for baseline benchmarking, Ridge and Lasso Regression for regularized models, Support Vector Regression (SVR) for margin-based prediction, and tree-based models like Decision Trees and Random Forests. Each model must be trained, validated using techniques like k-fold cross-validation and evaluated using metrics such as R-squared ( $R^2$ ), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). The system should generate a performance comparison report to guide model selection.

Once the best model is identified, the system must support real-time prediction for new inputs. The user should be able to enter values for GRE, TOEFL, CGPA, SOP, LOR, and research status, after which the system should return a probability or percentage score indicating the chance of admission. This prediction interface should be user-friendly, responsive, and secure. Additional interpretability options such as SHAP values or feature importance graphs can be provided to show which inputs influenced the prediction most, increasing user trust.

Each model must be subjected to rigorous training and validation procedures. The system should support k-fold cross-validation, preferably with  $k=5$  or  $10$ , to minimize overfitting ensure robust performance metrics across different data partitions. During training, hyperparameter tuning should be incorporated using GridSearchCV or RandomizedSearchCV to automatically identify optimal configurations for model-specific parameters like alpha in Ridge/Lasso and epsilon in SVR, or max\_depth in decision trees. To evaluate the performance of each model.

the system should compute and report key regression metrics, including R-squared ( $R^2$ )



to measure the proportion of variance explained by the model, Adjusted  $R^2$  to account for the number of predictors used, Mean Absolute Error (MAE) for average prediction errors and Root Mean Squared Error (RMSE) to penalize larger errors more heavily. These metrics should be summarized in a performance comparison dashboard or table, allowing users or developers to easily select the most accurate and reliable model for deployment.

### **Non-Functional Requirements**

In terms of performance, the system must deliver results quickly and efficiently. Predictions should be computed within 2–3 seconds per query, even under concurrent loads. For data preprocessing and training, the system should complete operations on datasets containing up to 10,000 records in under 10 seconds using standard hardware. These targets are necessary to ensure that the system is not only accurate but also practical in real-world deployment scenarios.

Scalability is a key non-functional requirement. As the user base grows—particularly during peak admission cycles—the system must maintain responsiveness. It should be architected with scalable components, potentially using microservices, and should allow deployment in cloud environments like AWS, GCP, or Azure. As user demand increases, horizontal scaling should automatically provision additional resources. Technologies like container orchestration (e.g., Kubernetes) can be used to manage dynamic workloads.

Reliability and availability are also crucial. The system must be accessible at all times, particularly during high-stakes admission seasons. It should have an uptime of at least 99.9% and support failover mechanisms to handle system crashes. Scheduled maintenance should be minimal and, if necessary, communicated in advance to users. System logs must be monitored to proactively detect and resolve errors.

Security and data privacy are paramount, especially since the system may deal with sensitive applicant information. All data transfers should use secure HTTPS protocols, and the system should implement authentication mechanisms such as login credentials or OAuth for restricted access. Personal data must be stored using encryption-at-rest, and data anonymization techniques should be used wherever appropriate. Furthermore, data storage and usage should comply with global standards such as GDPR, particularly if user data is retained.

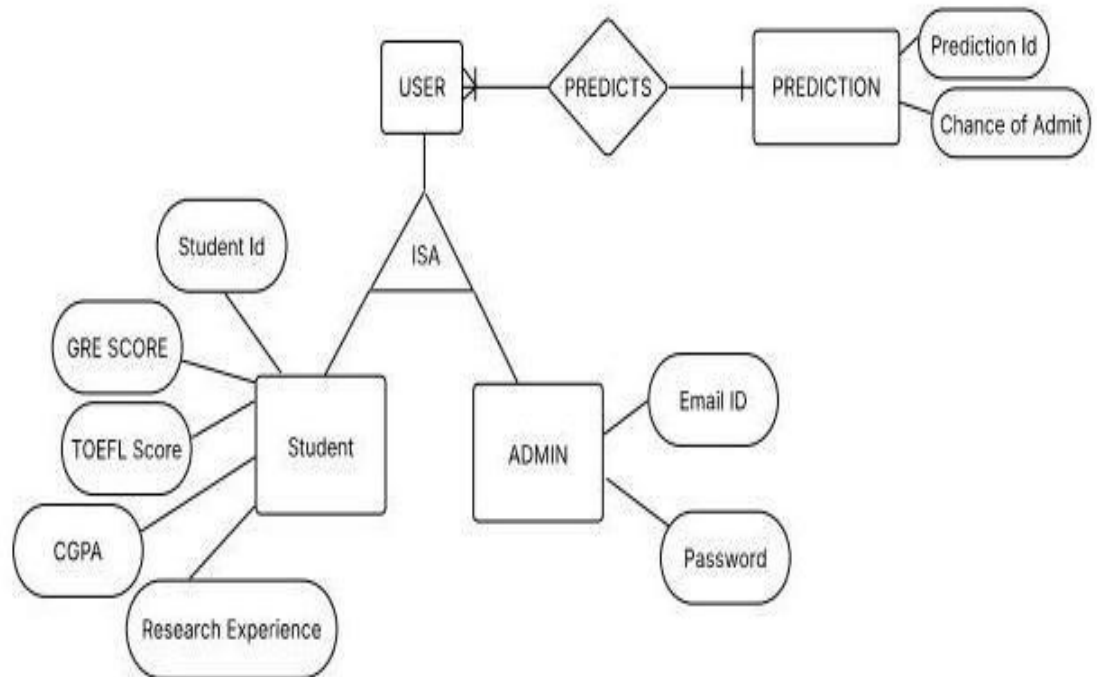
From a **usability** perspective, the system must offer a smooth, intuitive user experience. Both technical and non-technical users should be able to interact with the system without steep learning curves. The interface should provide descriptive messages for both success and error cases, and all data input fields must include clear labels and validation checks

**Maintainability** is essential for long-term success. The codebase should be modular and documented so that updates and improvements can be made easily without affecting unrelated modules. Logging should be built-in to help debug and track errors or unusual behavior. Version control systems (e.g., Git) and CI/CD pipelines can be used to automate testing and deployment.

**Interoperability** ensures that the system can work with other educational tools, portals, or analytics dashboards. It should offer RESTful APIs to facilitate integration with third-party applications, and support data exchange using JSON, XML, and CSV formats. The system should also support plug-and-play model upgrades, where new models can be introduced without major architectural changes.

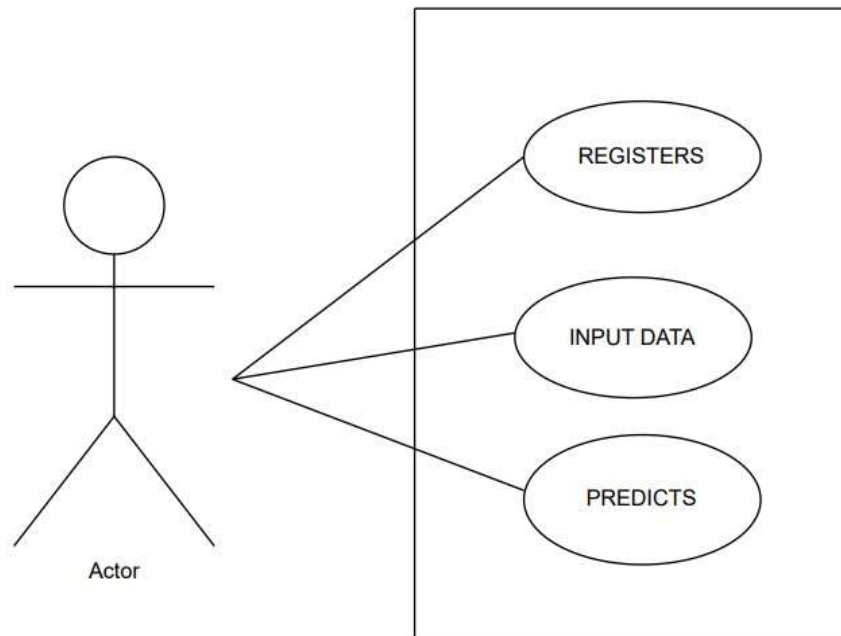
The system should ensure scalability, allowing it to handle a growing number of users, datasets without performance degradation. Usability is key interfaces must be intuitive and user-friendly, even for non-technical users such as applicants or academic staff. Security must be enforced through data encryption, secure authentication, and protection against input-based attacks. The system should be maintainable, with modular code and clear documentation to support future updates or model retraining. Lastly, portability should be considered, enabling deployment across different platforms or cloud environments with minimal configuration changes.

### 3.2 ENTITY RELATIONSHIP DIAGRAM

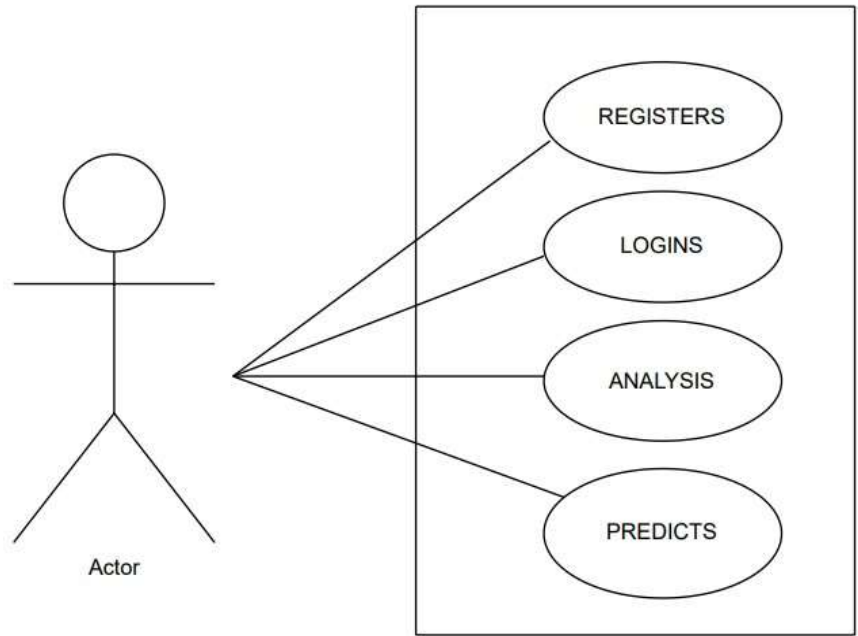


### 3.3 USE CASE DIAGRAM

#### STUDENT



**ADMIN**



### 3.4 TABLE STRUCTURE

**Table: STUDENT**

Column Name	Data Type	Constraints	Description
Student_id	INT	PRIMARYKEY, AUTO_INCREMENT	Unique Identifier for Each Student
GRE Score	INT	CHECK (GRE Score BETWEEN 260 AND 340)	GRE (Graduate Record Examination) Score, Ranging From 260 To 340.
TOEFL Score	INT	CHECK (TOEFL Score BETWEEN ( 0 AND 120)	TOEFL (Test of English as a Foreign Language) score, ranging from 0 to 120
SOP	FLOAT	CHECK(SOP BETWEEN(1.0AND 5.0)	Strength of the Statement of Purpose, rated on a scale of 1 to 5
LOR	FLOAT	CHECK(LOR BETWEEN(1.0AND 5.0)	Strength of the Letter of Recommendation, rated on a scale of 1 to 5.
CGPA	FLOAT	CHECK(CGPA BETWEEN(0 AND 10.0)	Cumulative Grade Point Average, typically ranging from 0 to 10
Research Experience	BOOLEAN	CHECK(Research Experience IN (0, 1))	Binary value indicating research experience: 0 = No, 1 = Yes

**Table : ADMIN**

Column Name	Data Type	Constraints	Description
Email_id	VARCHAR(100)	PRIMARY KEY, UNIQUE, NOT NULL	Unique email ID for login or identification
Password	VARCHAR(100)	NOT NULL	Encrypted user password for authentication

**Table: PREDICTION**

Column Name	Data Type	Constraints	Description
prediction_id	INT	PRIMARYKEY, AUTO_INCREMENT	Unique identifier for each prediction
student_id	INT	FOREIGNKEY REFERENCES ApplicantData (applicant_id)	Link to the student data table
chance_of_admit	DECIMAL (3, 2)	NOT NULL	Predicted chance of admission (0.00-1.00)

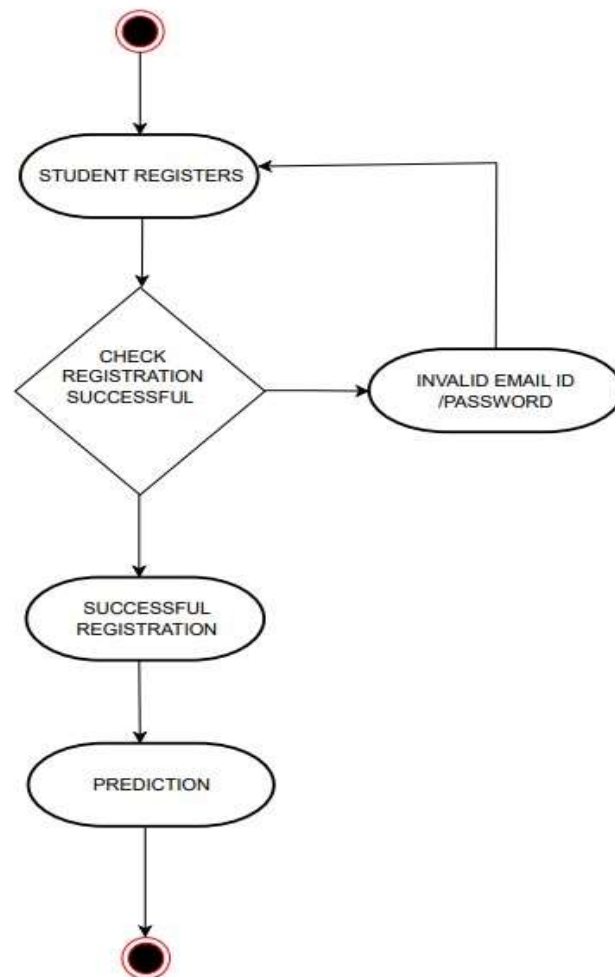
**Table : USER**

Column Name	Data Type	Constraints	Description
User_id	INT	PRIMARYKEY, AUTO_INCREMENT	Unique Identifier for Each Student
GRE Score	INT	CHECK (GRE Score BETWEEN 260 AND 340)	GRE (Graduate Record Examination) Score, Ranging From 260 To 340.
TOEFL Score	INT	CHECK (TOEFL Score BETWEEN( 0 AND 120))	TOEFL (Test of English as a Foreign Language) score, ranging from 0 to 120
SOP	FLOAT	CHECK(SOP BETWEEN(1.0AND 5.0))	Strength of the Statement of Purpose, rated on a scale of 1 to 5
LOR	FLOAT	CHECK(LOR BETWEEN(1.0AND 5.0))	Strength of the Letter of Recommendation, rated on a scale of 1 to 5.
CGPA	FLOAT	CHECK(CGPA BETWEEN(0 AND 10.0))	Cumulative Grade Point Average, typically ranging from 0 to 10
Research Experience	BOOLEAN	CHECK(Research Experience IN (0, 1))	Binary value indicating research experience: 0 = No, 1 = Yes

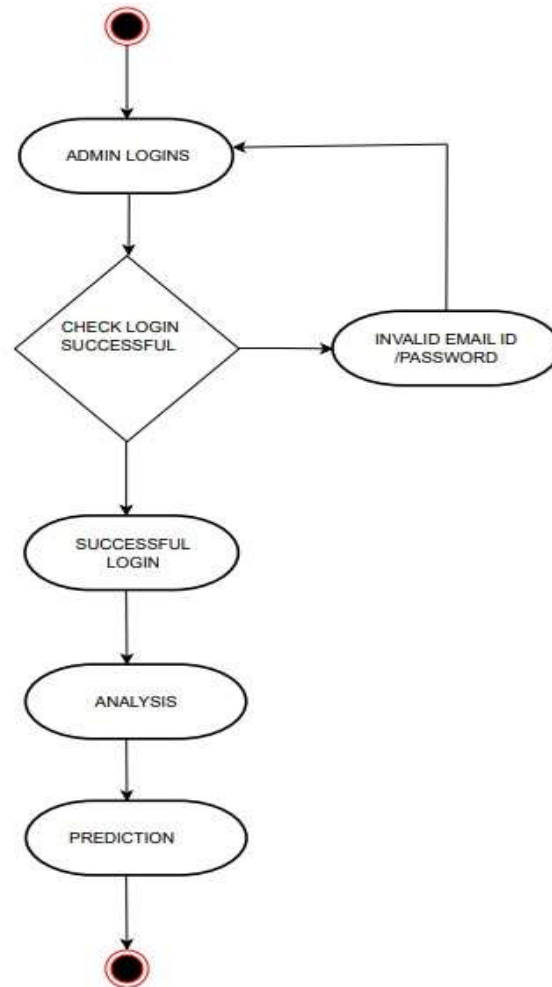


### 3.5 ACTIVITY DIAGRAM

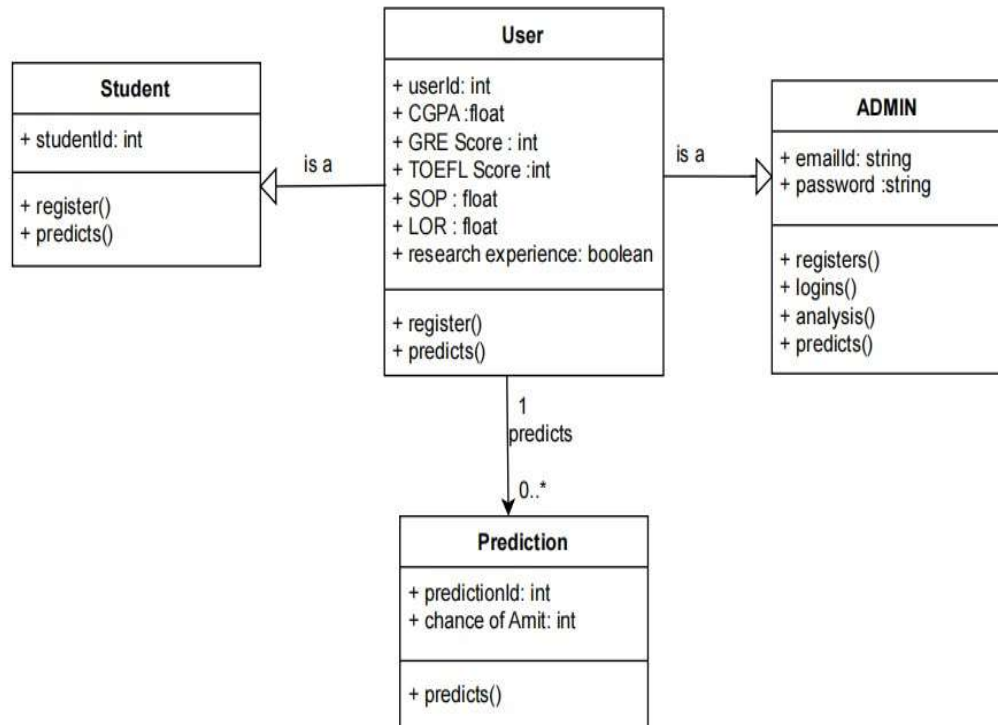
#### STUDENT



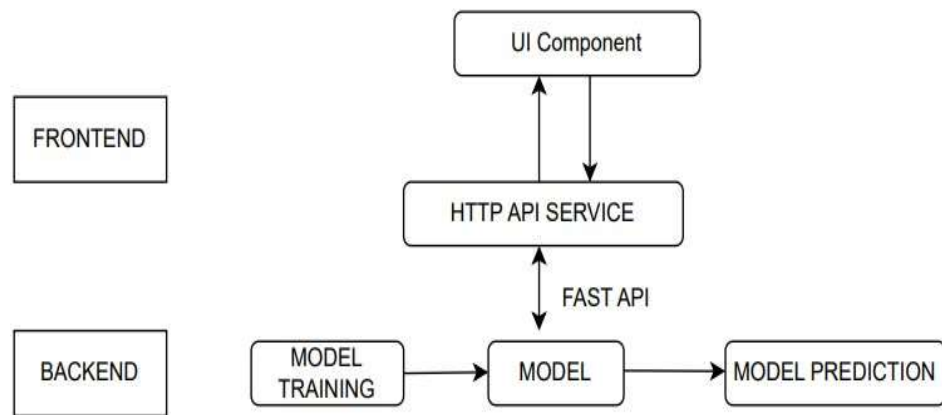
## ADMIN



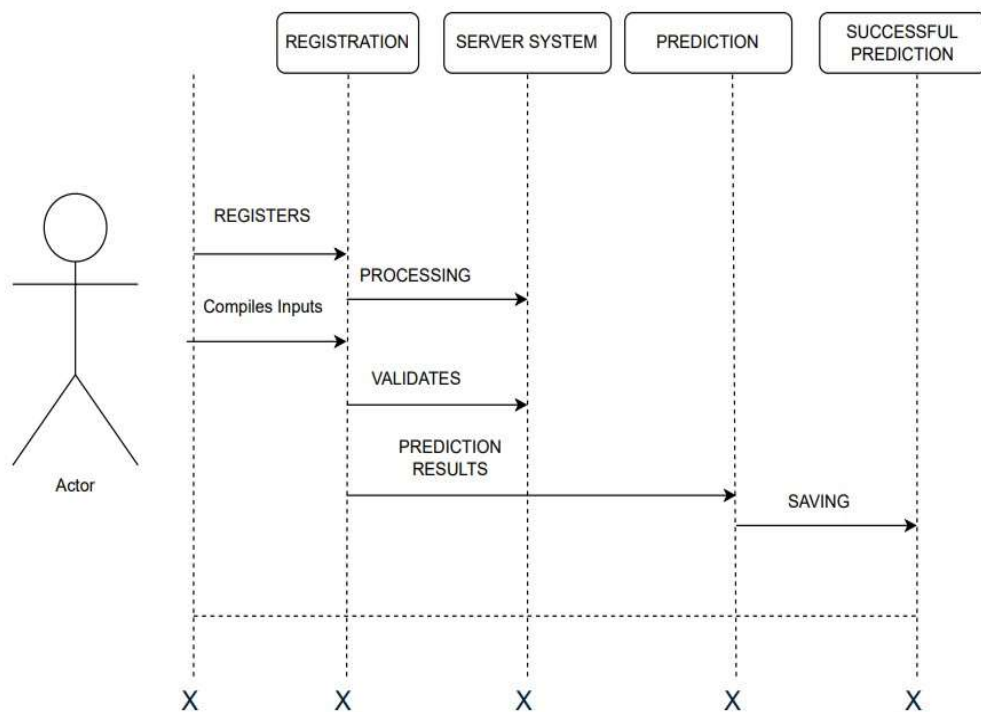
### 3.6 CLASS DIAGRAM



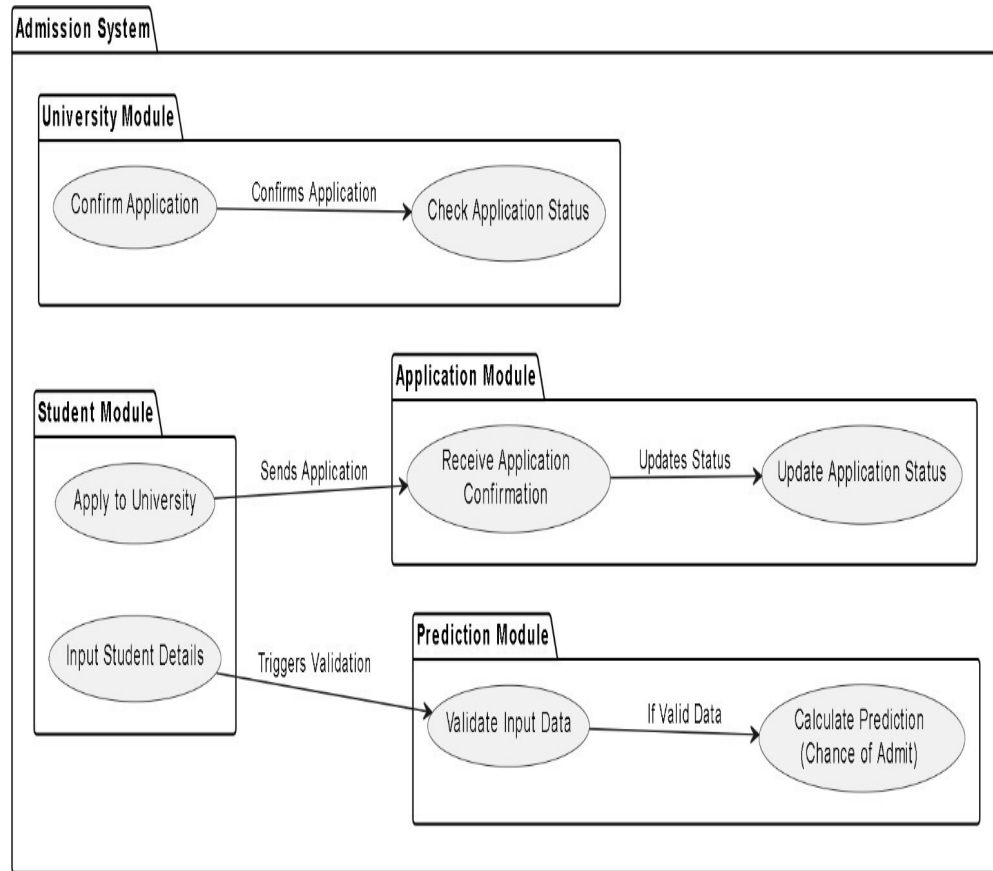
### 3.7 DEPLOYMENT DIAGRAM



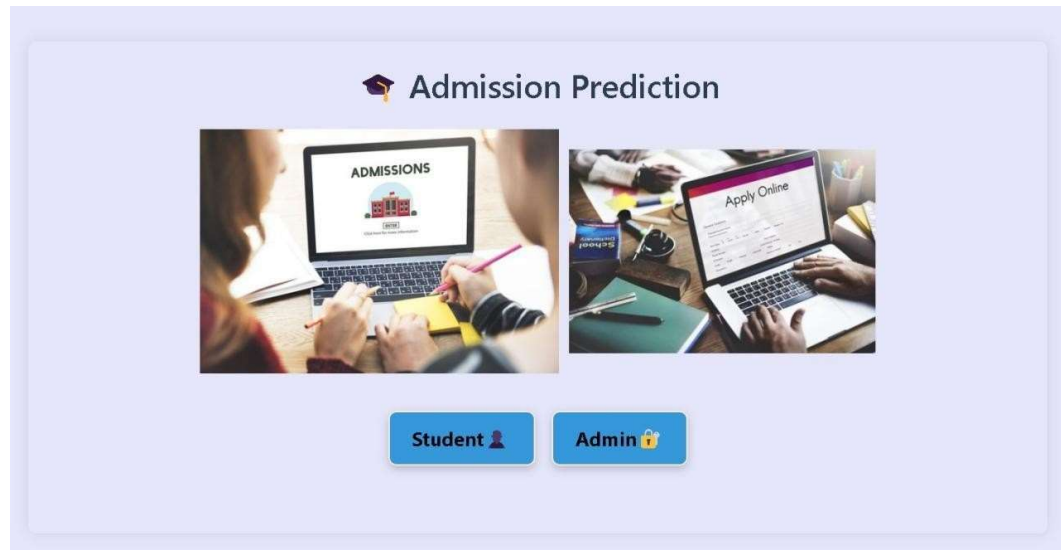
### 3.8 SEQUENCE DIAGRAM



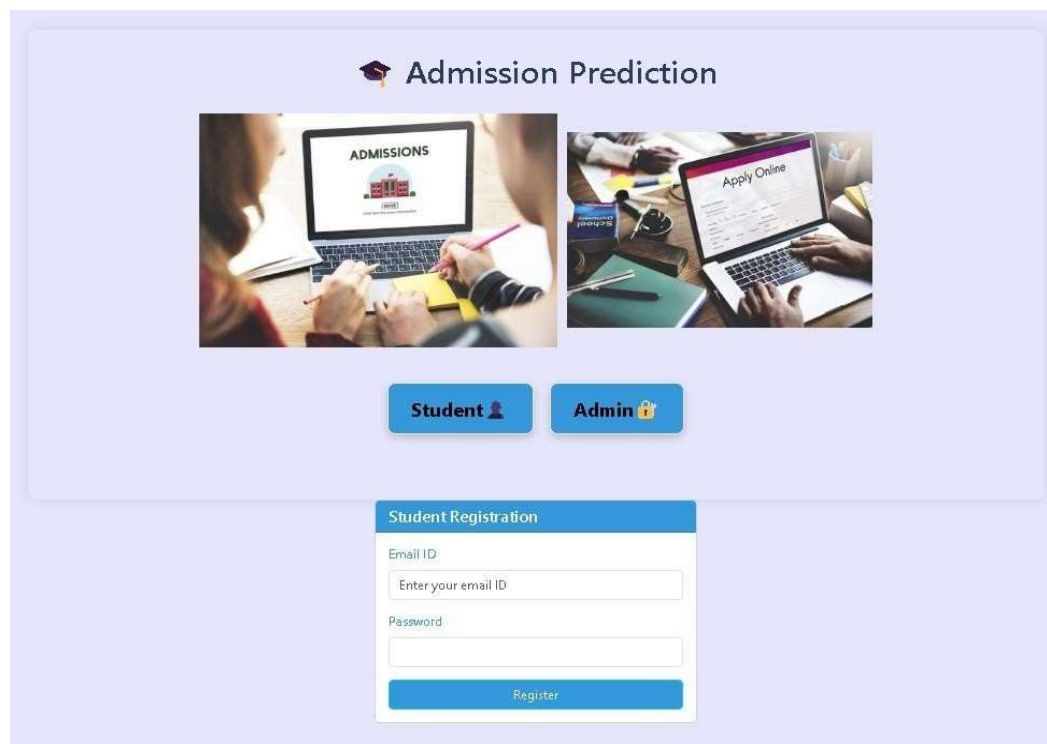
### 3.9 MODULE HIERARCHY DIAGRAM




### 3.10 INPUT & OUTPUT SCREENS






**Fig: User Interface of Admission Prediction Application**




**Fig: Student Registration User Interface**



Student 

Admin 

Admin Login

Admin Email

Admin Password

Login

**Fig: Admin Login Form UI**

### Admission Model Evaluation Dashboard

Display Model Evaluation Summary

Model Name	R <sup>2</sup> Value	Adjusted R <sup>2</sup>	MAE	MSE	RMSE
Linear Regression	0.8163	0.8053	0.04	0	0.06
Decision Tree Regressor	0.5716	0.546	0.06	0.01	0.09
Random Forest Regressor	0.7982	0.7861	0.04	0	0.06

**Fig: Admission Model Evaluation Dashboard**



Model Metrics

Polynomial Degree:

2

Get Model Metrics

Model Performance Metrics:

<b>R<sup>2</sup> Score:</b> 0.8132	<b>Adjusted R<sup>2</sup>:</b> 0.7368	<b>MAE:</b> 0.0427
<b>MSE:</b> 0.0036	<b>RMSE:</b> 0.0604	<b>Polynomial Degree:</b> 2

**Fig:Model Performance Metrics**

Predict Admission Chance

GRE Score:

TOEFL Score:

320

110

University Rating (1-5):

SOP (1-5):

LOR (1-5):

4

4

4.5

CGPA (0-10):

Research Experience:

9.2

Yes

Polynomial Degree for Prediction:

2

Predict Admission Chance

Prediction Result

Chance of Admission: 84.85%

Model Performance Metrics:

<b>R<sup>2</sup> Score:</b> 0.8132	<b>Adjusted R<sup>2</sup>:</b> 0.7368	<b>MAE:</b> 0.0427
<b>MSE:</b> 0.0036	<b>RMSE:</b> 0.0604	<b>Polynomial Degree:</b> 2

**Fig: Chance of Admission Prediction**

## CODING

### 4.1 ALGORITHMS

#### Project Overview

The Admission Prediction System leverages machine learning to estimate the likelihood of a candidate's acceptance into a university based on key attributes such as GRE and TOEFL scores, undergraduate GPA, and other factors. The system enhances decision-making for applicants and institutions by providing data-driven insights.

Phases: The project is divided into distinct phases, including data collection, exploratory analysis, model building, advanced machine learning techniques, deployment, and monitoring.

Tools: The project employs a variety of tools, including Python libraries like pandas, seaborn, scikit-learn along with machine learning algorithms such as Linear Regression, Decision Tree Regressor, and Random Forest Regressor.

#### Data Collection & Preparation

Sourcing: The data is sourced from a publicly available dataset, "admission\_predict\_system.csv," with attributes such as GRE Score, TOEFL Score, SOP, LOR, CGPA, and Chance of Admit.

Cleaning: Steps include handling missing values, dropping irrelevant columns (e.g., Serial No.), and ensuring consistent datatypes.

Feature Selection: Independent variables like GRE Score and CGPA are selected as predictors, while the dependent variable is Chance of Admit.

Scaling: Standardization of numerical features is performed using StandardScaler to ensure compatibility across machine learning models.

#### Exploratory Data Analysis (EDA)

Statistics: Descriptive statistics such as mean, median, and standard deviation provide a quantitative understanding of the dataset.

Visualizations: Histograms and KDE plots help identify the distribution of variables. A correlation heatmap reveals relationships among features.

Correlation: Significant positive correlations are observed between features like CGPA and GRE Score with the target variable Chance of Admit.

Outliers: Visualizations like box plots are used to detect outliers.

## Model Building

Three primary models are implemented:

Linear Regression: Captures linear relationships between features and the target.

Decision Tree Regressor: Captures non-linear relationships but is prone to overfitting.

Random Forest Regressor: An ensemble method that balances bias and variance for improved accuracy.

Training and Validation: The dataset is split into training (75%) and test sets (25%) to evaluate the models. Performance metrics such as R-square, Adjusted R-square, Mean Absolute Error, and Root Mean Square Error are computed.

## Advanced Machine Learning

Ensembles: Random Forest is used as an ensemble method, combining multiple decision trees to improve prediction accuracy.

Hyperparameter Tuning: GridSearchCV is planned for fine-tuning models, including parameters like tree depth and the number of estimators for Random Forest.

Deep Learning: This phase can include experimentation with deep neural networks for further improvements.

Model Training and Testing: The data is split into training and testing sets using stratified sampling to ensure the target variable's proportional representation. Models are trained on the training set and evaluated on the test set using performance metrics such as R-squared ( $R^2$ ), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE).

Advanced Machine Learning: To further improve model performance, ensemble learning methods like Random Forests are explored. Hyperparameter tuning using GridSearchCV is applied to optimize model settings, including tree depth, minimum samples per split, and number of estimators. K-fold cross-validation ensures the model generalizes well.

Potential for Deep Learning: For future phases, deep neural networks (DNNs) may be explored. Frameworks like TensorFlow or PyTorch could be employed to build and train more complex models, potentially increasing accuracy. Neural networks can capture complex patterns, especially in larger datasets.

Model Evaluation and Selection: Each trained model is evaluated on test data using  $R^2$ , MAE, and RMSE. Comparisons are visualized through bar plots. Based on the results, the most suitable model is selected for deployment. Random Forest often performs best due to its ability to manage bias and variance.

**Deployment and Real-time Prediction:** Once the model is finalized, it is saved using joblib or pickle. The FastAPI backend loads this model and exposes prediction endpoints. The frontend, built with HTML/CSS/JavaScript, accepts input features and displays the prediction results returned by the API. TOEFL score, SOP, LOR, CGPA, and research status. The values are sent via HTTP requests to the FastAPI backend, which returns the probability of admission. Results are dynamically displayed, providing an engaging and informative user experience.

**Interpretability:** To increase user trust, model interpretability tools like SHAP values or feature importance graphs are integrated. These tools explain which features most influenced a specific prediction, which is especially important for stakeholders unfamiliar with machine learning.

**Security and Ethics:** Data privacy and ethical considerations are crucial. Inputs and outputs are validated to prevent injection attacks. Sensitive data is encrypted in transit and at rest. The system avoids biases by regularly testing for fairness across demographic groups.

**Monitoring and Maintenance:** After deployment, system performance is monitored using tools such as Prometheus or custom logging scripts. Retraining schedules are defined to update the model with new data periodically. Performance drift is tracked to ensure accuracy remains consistent over time.

**User Roles and Access:** The system defines two primary user roles—Students and Admins. Admins can manage datasets, view aggregated analytics, and initiate model retraining. Students can only input personal data and view their prediction. Role-based access control (RBAC) ensures only authorized actions are performed.

**User Experience Enhancements:** To enhance usability, form validation, dropdowns, sliders, and helpful tooltips are integrated. Responsive design ensures compatibility across devices. A chatbot-like assistant may be added in the future to help users understand their admission chances and provide suggestions.

**Non-Functional Requirements:** Scalability is ensured through containerization using Docker, allowing the system to handle increasing workloads. High availability is achieved through server redundancy. Efficient resource usage minimizes cost without compromising performance. The system is deployable on cloud platforms like AWS, GCP, or Azure. Logging and exception handling are embedded to support debugging and maintenance. Usability testing ensures that interfaces are intuitive and user-centric.

Linear Regression is a fundamental machine learning algorithm used for predicting continuous outcomes by establishing a linear relationship between the input features and the target variable. Decision Trees, on the other hand, represent decisions and their possible consequences using a tree-like structure, making them highly interpretable and suitable for both classification and regression tasks. Building on this, Random Forest is an ensemble method that combines multiple decision trees to improve predictive performance, enhance robustness, and reduce the risk of overfitting by averaging the results of many trees.

#### Project Execution

In this project, the backend is developed using FastAPI, enabling the efficient creation of RESTful API endpoints to handle data processing, logic execution, and server-side communication. These API endpoints serve as the backbone for interacting with the application's functionalities, such as data retrieval, prediction, or user input handling. The frontend is designed using HTML, CSS, and JavaScript, providing a responsive and interactive user interface. It sends HTTP requests to the FastAPI backend and dynamically displays the results to the users. This separation of concerns ensures a clean architecture and smooth integration between the client and server.

#### Ethics & Best Practices

**Fairness:** The model is tested for bias to ensure fairness across demographic groups.

**Explainability:** The inclusion of interpretable models ensures stakeholders understand the rationale behind predictions.

This Admission Prediction System brings together various aspects of software engineering, and user experience design to deliver an efficient, reliable tool students and educational institutions alike. Through continuous enhancement, ethical governance, and robust design principles, the system aims to serve technology.

# TESTING

## 5.1 Test Strategy

A comprehensive test strategy ensures that the university admissions prediction system operates as intended, delivering accurate, reliable, and user-friendly outputs. The testing process involves multiple stages, addressing both functional and non-functional requirements, and leveraging a combination of manual and automated testing approaches.

### Testing Objectives

Validate the accuracy and robustness of predictive models.

Ensure data integrity and correctness in preprocessing and feature engineering steps.

Verify the system's ability to handle edge cases and diverse user inputs.

Test the performance, scalability, and usability of the system.

### Unit Test Plan

Focus: Validate individual components of the system, such as preprocessing scripts, model training pipelines, and utility functions.

Tools: Python testing frameworks like unit test or pytest.

Ensuring that missing values are handled correctly in the preprocessing step and that the model outputs valid probability scores.

### Acceptance Test Plan

**Focus:** Validate that the system meets the expectations of end-users.

**Process:** Conduct sessions with applicants, admissions officers, and consultants to collect feedback on usability and accuracy. Ensuring that predictions align with expert evaluations.

### Integration Testing

Focus: Test interactions between system components, such as the data pipeline, model inference, and user interface.

Tools: Postman (for APIs), Selenium (for UI components).

Verify that the trained model integrates seamlessly with the web application to provide predictions in real-time.

### **Regression Testing**

Focus: Ensure that updates or changes to the system do not introduce new defects.

Approach: Maintain a suite of test cases to re-run after each system update.

After hyperparameter tuning, check that the adjusted model does not negatively affect existing prediction accuracy.

### **Functional Testing**

Focus: Validate system features against functional requirements.

Techniques: Black-box testing.

Example: Ensuring that inputting applicant data (e.g., GRE, TOEFL, CGPA) generates a prediction of the chance of admission.

It verifies whether each feature of the admission prediction system performs as expected.

Test cases are derived from the specification without knowledge of internal code.

Validation includes checking the accuracy of predictions based on historical data.

Also ensures correct user roles (student/admin) have appropriate access to functionalities.

Includes testing user registration, login, and role-based access.

Helps identify missing or incorrect functionalities early in development.

Functional tests also verify data validation rules and error handling.

Ensures predictions are not generated without complete input data.

Supports testing boundary conditions (e.g., min/max scores for GRE, TOEFL).

Also involves checking data persistence and retrieval from the backend.

Provides confidence that the system behaves according to defined use cases.

### **Performance Testing**

Focus: Assess the system's responsiveness and stability under various conditions.

Techniques: Load testing and stress testing.

Example: Simulating multiple users accessing the prediction feature simultaneously.

It evaluates how quickly and reliably predictions are returned under peak load.

Stress testing determines the system's breaking point and behavior during failures.

Performance bottlenecks, memory leaks, or slow database responses are identified.

Ensures the system scales well as the number of users or data volume increases.

Includes measuring response time, throughput, and resource utilization.

Load testing helps ensure that server and database resources are optimized.

Test results guide decisions on system tuning and infrastructure upgrades.

## Test Cases

### Preprocessing Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Pass/Fail Criteria
TC_01_Preprocess	Verify missing value handling	Dataset with missing GRE scores	Missing values replaced with mean/median	Missing values handled correctly
TC_02_Preprocess	Check data standardization	Dataset with unscaled CGPA values.	CGPA values scaled to 0-1 range	Output matches scaling function
TC_03_Preprocess	Test datatype conversion	String CGPA in dataset.	CGPA Converted to float	Datatypes match
TC_04_Preprocess	Test outlier detection/removal	Dataset with CGPA = -1 or GRE = 8000	Outliers flagged or removed	Outliers handled as per preprocessing strategy
TC_05_Preprocess	Validate categorical encoding	Dataset with categorical fields (e.g., SOP)	Categorical fields encoded numerically	Encoded output as expected
TC_06_Preprocess	Check duplicate removal	Dataset with duplicate records	Duplicate records removed	No Duplicate entries remain
TC_07_Preprocess	Validate pipeline with empty dataset	Empty dataset	Appropriate error or empty output	System handles gracefully



### Model Training Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Pass/Fail Criteria
TC_01_ModelTrain	Verify model training process.	Training dataset (cleaned).	Model trained successfully without errors	Training completes without exceptions
TC_02_ModelTrain	Validate feature importance ranking	Dataset with all attributes	Model provides ranking of features	Feature ranking aligns with domain knowledge
TC_03_ModelTrain	Test performance Metrics during training	Training dataset	R-squared>0.8, MSE within acceptable limits	Metrics exceed threshold
TC_04_ModelTrain	Validate model saving/loading	Trained model	Model saved and reloaded successfully	Reloaded model gives same predictions
TC_05_ModelTrain	Test hyperparameter tuning	Training dataset	Best parameters found and reported	Report includes tuned parameters
TC_06_ModelTrain	Test reproducibility with random seed	Training dataset with fixed seed	Same model metrics on repeated runs	Same model metrics on repeated runs
TC_07_ModelTrain	Validate error handling for corrupted data	Corrupted or malformed dataset	Meaningful error message	Clear Error

### Prediction Functionality Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Pass/Fail Criteria
TC_01_Predict	Verify prediction output format	GRE: 320, TOEFL: 110, CGPA: 8.5	Predicted chance of admit (0-1)	Output is a valid probability score
TC_02_Predict	Test prediction with edge values	GRE: 0, TOEFL: 0, CGPA: 0	Low chance of admit	Prediction aligns with input logic
TC_03_Predict	Test prediction For high-performing candidate	GRE: 340, TOEFL: 120, CGPA: 10	High chance of admit (~1)	Prediction aligns with input logic
TC_04_Predict	Test prediction with invalid inputs	GRE: "abc", TOEFL: -10, CGPA: "N/A"	Error or prompt to correct inputs	Error message shown without crash
TC_05_Predict	Validate batch prediction	Batch of 50 applicant records	Batch predictions returned	Predictions generated for all records
TC_06_Predict	Test multi-model prediction comparison	Applicant data + multiple models (if available)	Predictions from different models	Consistency check between models (if required)
TC_07_Predict	Validate prediction caching (if applicable)	Repeat same applicant data	Result returned from cache	Faster response time for repeated requests

### Edge Case Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Pass/Fail Criteria
TC_01_Edge	Test with missing or null fields	GRE: 320, TOEFL: null, CGPA: 9.0.	Prediction still generated (handling missing fields)	System does not crash, and output is valid
TC_02_Edge	Test with extreme high values	GRE: 400, TOEFL: 200, CGPA: 15.0	Error or capped prediction output	Output within logical bounds
TC_03_Edge	Test with very large batch size	10,000 applicant records	Predictions returned or memory error handled	System remains stable
TC_04_Edge	Test API request timeout	Prediction API with delayed response	Timeout error or fallback response	System times out gracefully
TC_05_Edge	Test prediction without authentication (if API)	No auth token	Authentication error	Correct error response

### Usability Test Cases

Test Case ID	TestCase Description	Input	Expected Output	Pass/Fail Criteria
TC_01_Usability	Validate ease of navigation	User navigates through the system	Smooth navigation without confusion	Feedback from users is positive
TC_02_Usability	Test clarity of prediction results	Prediction request submitted	Prediction displayed clearly with explanation	Users understand the result

## 5.2 DEFECT REPORT

Defect Id	Defect Description	Module	Priority	Severity	Steps to reproduce	Expected Behavior	Actual Behavior	Status	Assigned
D01	Incorrect handling of missing GRE Scores	Data preprocessing	High	High	Input dataset with missing GRE Scores. Run preprocessing script	Missing GRE scores replaced with mean or median values	Missing GRE scores remains null, causing training failure	Open	Niks
D02	Incorrect Scaling of CGPA values	Data preprocessing	High	High	Input dataset with CGPA A ranging from 6 to 10. Run preprocessing	CGPA scaled to 0-1 range	CGPA scaled incorrectly resulting in values >1	Open	Niks
D03	Model performance below acceptable threshold	Model Training	High	High	Train model on dataset. Evaluate metrics (R square, MSE)	R square > 0.8. MSE less as accepted	R square = 0.65. MSE higher	Open	Niks

D04	API returns error 500 when TOEFL score is missing	Prediction API	High	High	Submit prediction request with GRE and CGPA, leave TOEFL empty.	Prediction generated with missing field handled gracefully	API crashes with error 500.	Open	Niks
D05	Logs not generated for prediction requests	Logging	High	High	Submit prediction request. Check logs.	Log entry with timestamp, input, and output.	No logs generated	Open	Niks
D06	Model fails to load after redeploy	Backend Integration	Medium	Medium	Redeploy app and call prediction	Model loads and serves prediction	API error: "Model not found"	Open	Niks
D07	Incorrect error handling for corrupted CSV upload	Data Import	High	Medium	Upload malformed CSV file with broken column	Error message with clear guidance	System crashes; no meaningful error shown.	Open	Niks
D08	Caching not working for repeated requests	Prediction API	Low	Low	Submit same applicant data twice. Measure response time	Faster response for second request (cached).	Same response time; no caching observed.	Open	Niks

# USER MANUAL

## Admission Prediction Application

The Admission Prediction Application uses machine learning to estimate the probability of a candidate being admitted to a university based on academic and profile-related features.

## System Requirements

Web Browser: Chrome, Firefox, Edge (latest versions recommended)

Internet Connection: Required

## Features

Predicts admission chances based on user input.

Displays probability of admission along with recommendations.

## Input Parameters

Users must provide the following details:

GRE Score (e.g., 290–340)

TOEFL Score (e.g., 90–120)

University Rating (1–5)

Statement of Purpose (SOP) Score (1–5)

Letter of Recommendation (LOR) Score (1–5)

CGPA (e.g., 6.0–10.0)

Research Experience (Yes/No)

## How to Use

Open the application in your browser or run the script (if local).

Enter the required input values in the corresponding fields.

Click on the submit button to predict.

The application will display your Admission Probability as a percentage.

## Output Explanation

Admission Chance: A percentage score indicating your likelihood of admission.

## Notes & Limitations

This is a predictive tool and not a guarantee of admission.

The prediction is based on a trained model using historical data.

Always consult official university guidelines for actual admission criteria.

## **LIMITATIONS OF PROPOSED SYSTEM**

### **Dependence on Historical Data**

The system relies heavily on the quality and representativeness of the training dataset. If the dataset contains biases or does not reflect current admission trends, predictions may be inaccurate. Overrepresentation of candidates from specific regions or educational systems may skew the model's predictions.

### **Limited Scope of Features**

While the system considers important factors like GRE, TOEFL, CGPA, and others, it does not account for subjective or qualitative factors such as: Extracurricular achievements, personal statements, or recommendation letters. Institutional preferences, such as a focus on diversity or specific disciplines.

### **Potential for Overfitting**

If the model is too complex or tuned excessively to the training data, it may perform well on historical data but poorly on new or unseen cases, limiting its generalizability.

### **Lack of Contextual Interpretation**

The system provides probabilities for admission but lacks the ability to explain the broader context or rationale behind predictions. This can make it challenging for users to: Understand why they received a particular prediction.

Identify actionable steps to improve their chances of admission beyond numerical improvements.

### **Sensitivity to Input Data Quality**

Errors or inconsistencies in user-provided input data (e.g., incorrect GRE or TOEFL scores) can significantly impact prediction accuracy. Additionally:

Missing or incomplete data may lead to unreliable predictions.

## PROPOSED ENHANCEMENTS

The prediction system for university admissions using GRE and TOEFL scores can be expanded and enhanced in numerous ways to increase its utility, accuracy, and user-friendliness. The following points outline potential future developments and enhancements:

### **Advanced Machine Learning Techniques:**

**Deep Learning Models:** Implement deep learning techniques such as neural networks to capture complex relationships in the data.

**Ensemble Methods:** Use ensemble methods like Random Forests, Gradient Boosting Machines (GBM), or XGBoost to enhance prediction accuracy.

**Personalized Recommendations:** **University and Program Recommendations:** Provide personalized recommendations for universities and programs based on the applicant's profile and historical admission data. **Improvement Suggestions:** Offer actionable advice to applicants on how to improve the chances of admission, such as retaking exams or enhancing specific skills.

**Integration with Other Systems:** **Educational Platforms:** Integrate with educational platforms like Coursera, edX, and LinkedIn Learning to consider completed courses and certifications. **Application Portals:** Link with university application portals to streamline the application process and provide seamless submission of predictions and recommendations.

**Scalability and Performance Enhancements:** **Cloud Scalability:** Utilize cloud infrastructure to handle increased user load and large-scale data processes. **Real-Time Predictions:** Implement real-time prediction capabilities to provide immediate feedback to applicants.

**Security and Compliance:** **Data Privacy:** Enhance data privacy measures to comply with regulations like GDPR and CCPA. **Secure Authentication:** Implement advanced authentication mechanisms like biometric authentication or multi-factor authentication.



## CONCLUSION

The proposed university admissions prediction system represents a significant step forward in leveraging machine learning to enhance the efficiency, accuracy, and fairness of the admissions process. By utilizing key applicant attributes such as GRE and TOEFL scores, CGPA, and other factors, the system provides data-driven predictions that can support applicants, admissions officers, and other stakeholders in making informed decisions.

Through careful implementation, including data preprocessing, exploratory data analysis, feature selection, and rigorous model evaluation, the system demonstrates the potential to streamline admissions workflows and offer valuable insights into factors influencing admission outcomes.

However, the system's limitations, such as dependence on historical data, exclusion of qualitative factors, and sensitivity to input quality, highlight the need for ongoing refinement and the integration of complementary tools to address these gaps.

In conclusion, while the system is not a substitute for comprehensive human evaluation, it serves as a powerful supplement that can improve transparency and efficiency. With further enhancements and ethical safeguards, this predictive model can become a valuable tool in modernizing university admissions, benefiting both institutions and applicants alike.

## **BIBLIOGRAPHY**

### **BIBLIOGRAPHY ANEXURES**

Rutuja Konde, Rutuja Somvanshi, Pratiksha Khair, Prachi Zende, Kamlesh Patil, “Admission Prediction System”, International Research Journal of Engineering and Technology (IRJET), Volume: 09 Issue: 05, May 2022.

Mrs. Sowjanya, Saud Ikram, Rayyan Khan, Shaik Haseeb Jawad, Saadul arifeen, “Prediction of Airfare Prices Using Machine Learning,” International Journal of Mechanical Engineering, Vol.7 No.6, ISSN: 0974-5823, June, 2022.

Pavithra Maria K, Anitha K L, “Admission Prediction for Users by Machine Learning Techniques”, International Advanced Research Journal in Science, Engineering and Technology, Vol.8, Issue 3, DOI: 10.17148/IARJSET.2021.8321, March 2021.

Prof. Ms. Archana Dirgule, Shubham Agarwal, Ram Agrawal, Neha Singh, Kiran Adsul, “Admission Prediction using Random Forest Algorithm,” International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), Volume 2, Issue 3, ISSN (Online) 2581-9429, May 2022.

Neel Bhosale, Pranav Gole, Hrutuja Handore, Priti Lakde, Gajanan Arsalwad, “Admission Prediction Using Machine Learning,” International Journal for Research in Applied Science & Engineering Technology (IJRASET), ISSN: 2321-9653, Volume 10, Issue V, May