**An Intelligent Application For Improving English Proficiency Using Machine Learning**

Prathmesh Phatake, Dhanaraj Pingle, Sneha Rahate, Abhijeet Shinde

1. Student at Dept. of Computer Engineering, BRACT’s Vishwakarma Institute of Information Technology, Pune, India, E-mail: prathmesh.22211184@viit.ac.in
2. Student at Dept. of Computer Engineering, BRACT’s Vishwakarma Institute of Information Technology, Pune, India, E-mail: dhanraj.22210367@viit.ac.in
3. Student at Dept. of Computer Engineering, BRACT’s Vishwakarma Institute of Information Technology, Pune, India, E-mail: sneha.22211041@viit.ac.in
4. Student at Dept. of Computer Engineering, BRACT’s Vishwakarma Institute of Technology, Pune, India, E-mail: abhijeet.22211302@viit.ac.in
5. Assistant Prof at Dept. of Computer Engineering, BRACT’s Vishwakarma Institute of Technology, Pune, India, E-mail: disha.wankhede@viit.ac.in

**Abstract**

This research paper proposes and develops an intelligent conversational system for evaluating and enhancing language proficiency. Using a machine learning model, it predicts the required practice days and daily minutes that would be necessary for reaching 90% target proficiency level. Used technologies include Python's Django framework for the backend, HTML/CSS for the frontend, and joblib for deploying pre-trained models. In addition, this system has also included speech-to-text and text-to-speech functionalities for fluent conversational interactions, thereby providing personalized feedback that enhances the users' experience in language learning.

**[I]Introduction**

With increasing demand for proficiency in languages in global communication, intelligent systems of language learning become supplements to traditional modes of language learning. The present project meets this requirement of designing a conversational system that not only encourages the user to get into real-time dialogue but will also provide them with personalized insights into their own proficiency level. Machine learning and natural language processing have been employed here for predicting both the time and intensity of practice required to achieve a desired level of proficiency that could be 90%.

**[II]Literature Survey:**

In the literature survey of an intelligent application for improving English proficiency using machine learning, various related works have been explored. One system utilized sensors, technologies, and interfaces for real-time information collection and display, albeit requiring expensive private infrastructure. Other such a platform

Language proficiency testing is an increasingly important part of global society. The need to demonstrate language skills—often through standardized testing—is now required in many situations for access to higher education, immigration, and employment opportunities. However, standardized tests are cumbersome to create and maintain. Lane et al. (2016) and the Standards for Educational and Psychological Testing (AERA et al., 2014) describe many of the procedures and requirements for planning, creating, revising, administering, analysing, and reporting on high-stakes tests and their development.[1]

**[III]Objective**

The overall objective of this research project is the development of a Web Application based on frontend design by HTML and CSS, MongoDB as the database, and for connectivity, Python Django. The specific aims of the project are as follows:

1. Developing a speech-based interactive language proficiency assessment system.
2. Calculation of the number of days and minutes daily to reach the target proficiency level of 90%.
3. Machine learning-based prediction with feedback at real time.
4. Interactive and adaptive dialogue: enhancing user engagement.
5. Seamless integration of error-free text-to-speech and speech-to-text conversion for smooth conversation.

**[IV]Methodology**

1. **HTML, CSS & JAVA-SCRIPT:**

HTML and CSS are two basic technologies of web development, helping carry out the basically important functions of creating a good-looks, interactive user interface.

**HTML Hypertext Markup Language**

* HTML Basically represents a skeleton of web pages, offering an organization of content.
* It employs tags which describe what any given element of any webpage is, and thus comprises headings, paragraphs, lists, images, forms, links.
* HTML would be the base where the styles of CSS would be added to improve the layout of the page.

**CSS (Cascading Style Sheets)**

* CSS style the HTML element, which defines its look, layout, and behavior on the web page.
* It allows developers to have control over items such as font, color, spacing, border, background, and position.
* Responsive design: CSS is used to create responsive designs that adapt to different screen sizes and devices. It will increase users' experience over various platforms.
* CSS makes it possible to use selectors and rules that can be very powerful in delivering uniformity with styles, thereby reiterating professionalism.
* CSS also supports animations and transitions, allowing developers to add interactive elements and visual effects to enhance user engagement.

**JAVA-SCRIPT**

* Interactivity and Dynamic Content: JavaScript enables developers to create interactivity such as form validation, animations, and updating dynamic content on a page without requiring full page refresh. This makes the user experience more engaging because the user will see instant feedback and response.
* DOM Manipulation: JavaScript can manipulate the Document Object Model, which helps developers alter contents, structures, and styles of web pages in a dynamic way. This makes complex interfaces which react to user interactions without breakage.
* Asynchronous Communication: JavaScript through the use of AJAX (Asynchronous JavaScript and XML) allows asynchronous client-server communication. This enables data to be fetched in the background without affecting the experience of the user and gives the application a lifelike feel by updating the information while not blocking the user's interaction with the page.
* Frameworks and Libraries: JavaScript has an incredibly rich landscape of frameworks and libraries that make it easy for someone to build complicated web applications be it with React, Angular, or Vue.js. These tools provide pre-built components that allow it to efficiently manage application state, which accelerates the development of far more complex applications.
  + Cross-Browser Compatibility: JavaScript ensures that web applications function consistently in any other browser and the device one is using. Developers can implement polyfills and use feature detection techniques to provide a uniform experience for all users. They are three-part, balance each other out so that the interface in web applications is at the same time pleasing to look at, accessible, and user-friendly; whereas HTML forms the structure and semantics, CSS brings aesthetic and stylistic improvements, while added enhancements of interactive and dynamic experiences for a user arise due to JavaScript.

By leveraging the capabilities of HTML, CSS, and JavaScript, developers can design interfaces that not only meet functional requirements but also captivate and delight users with their visual appeal, responsiveness, and engaging interactivity.

1. **Python Django connectivity:**

In the Intelligent Application for Improving Proficiency Project, Django constitutes a fundamental role in realizing communication with the database, server-side logic, and frontend interface. Here's how Django enables this communication:

* + Routing and URL Handling: Django can support Python functions as it associates routes. Routes defined within the application describe endpoints that will process incoming HTTP requests. Here, functionalities like user authentication assessment, session management, and return of feedback are taken up.
  + Template Rendering: Django's integrated template engine is used very aptly with the application, and it generates dynamic HTML content with data retrieved from the database. This allows a developer to create reusable templates for rendering almost all frontend components, like proficiency assessment results, user dashboards, and much more about feedback forms. Then, using context variables passed to such templates, Django perfectly helps in the generation of dynamically composed content based on the needs of the user.
  + Database Interaction: Due to the ORM (Object-Relational Mapping) in Django, this project has been enabled to connect easily with the database, meaning it streamlines interaction between the stored data. It streamlines CRUD operations from proficiency assessments, user profiles, and session records, making it easy to manage data.
  + Form Handling and Validation: Django Forms enhanced input handling and validation for the users: it offers Python classes to assist in building and validating forms. The framework ensures integrity and security in data management, shielding against CSRF attacks. Hence, error management becomes easier, and this is entirely integrated with templates within Django. Overall, development experience and solidity of the application get improved.

In other words, Django is a tool for free communication and cooperation between the frontend UI of the Intelligent Application to Improve Proficiency, its database, and server-side functionality. Simplicity, flexibility, and its vast ecosystem of extensions made Django an extremely suitable option for creating web applications with complex functionalities such as user authentication, database interaction, and dynamic content rendering on the backend.

1. **Machine learning model:**

**Decision Tree Regressor**

1. Data collection

* Source data. Gather all data on language proficiency-from session duration to an individual's initial proficiency level, and so forth. Such data would be available from user assessment, surveys, and historical records of performance.
* Target Variable Definition: Define the target variable with high precision. In this case, it refers to how many total days and minutes per day will be required to attain the target level of 90% proficiency by the users.

1. Data Preprocessing

* Data Cleaning:
* Remove duplicates and undue noise in the data.
* Make sure any missing values are imputed either by mean, median, mode or removes them from records.
* Feature Engineering:
* Convert categorical variables, if any, to numerical representations through techniques like one-hot encoding or label encoding.
* Normalize or standardize numerical features, if needed, in order to let features be on comparable scale.
* Apply Train-Test Split. Partition this data into train-set and test-set in a typical 80:20 split. This ensures unbiased evaluation of the model.

1. Model Selection and Configuration

* Selection of Model Now, choose a Decision Tree Regressor from all the algorithms available for use for regression tasks. As can be noted, decision trees are appropriate for that category since they handle non-linear models and interaction between features.

Hyperparameter Tuning:

* Define hyperparameters like the maximum tree depth, minimum samples required to split a node, and minimum samples required at the leaf node.
* Apply methodologies, such as Grid Search or Random Search with cross-validation, to select the best set of hyperparameters that achieves the highest performance.

1. Model Training

* Fitting the Model:

Train the Decision Tree Regressor with the training dataset. In this way, it learns to associate input features (initial proficiency, session duration, improvement rates) with the target variable (total days and minutes required).

* Creating the Decision Tree:

The algorithm, of course, constructs a tree-like structure. Decisions made with respect to the values of input features go into every node, and every leaf node corresponds to a predicted outcome: total days and minutes required.

1. Model Evaluation

* Prediction: Use the trained model to make predictions on a test dataset. For each test instance, it makes the prediction of total days and minutes per day to reach the target proficiency level.
* Performance Metrics:

Apply the model with proper measures appropriate metrics

Mean Absolute Error (MAE): The measure which shows an average absolute difference between predictive and actual values.

MSE: Average of the squares of errors; larger errors are given more weight.

R-squared: Refers to the fraction of variance in the target variable explained by the model, indicating its performance in terms of explanatoriness.

* Visual Interpretation:
* Plot the decision tree to understand the structure and rules of decisions.
* Produce residual plots to graphically demonstrate the distribution of residuals, as this will help diagnose overall patterns or biases in the predictions.

6. Interpretation and Application

* + Feature Importance: how much a certain feature contributed to the decision tree's decision. This could be really useful in understanding what really influences the prediction about increasing their proficiency.
  + User's Feedback Loop: Use the outputs of this model to provide them with specific feedback that nudges them into their learning loop.

7. Deployment and Aftercare

* Tuning the Trained Decision Tree Regressor Model into a Smart Application for Increasing Capability, in addition to Providing Instant Predictions of Desired Service Based on the Input Provided by Users.
* Monitor and Update: Monitor the performance of the model constantly with new data. Then, update and retrain to continue to have accurate and relevant predictions.

**Linear regression**

1. Dataset Collection: Collect the dataset by involving relevant feature attributes concerning improvement in language proficiency. Some possible variables in the dataset include the initial levels of proficiency, sessional duration rates of improvement, and total days required for obtaining the target level of proficiency.
2. Data Preprocessing

* Load the Dataset: Use libraries like Pandas to load the dataset into a DataFrame for analysis and manipulation. In this case, the dataset is loaded from an Excel file.
* Feature Selection:
  + specifies the features, which are independent variables and include:
    - Initial Proficiency (%): This will be at the start of the session.
    - Session Duration (minutes): The length of the practice session is.
    - Improvement Rate (%): The rate that is expected after every session.
    - Minutes Per Day: The minutes spent by the user each day.
  + Define the target variable (dependent variable), in this case, Total Days Required for achieving the required level of proficiency.

1. Splitting the Dataset

* Train Test Split: This is where the dataset is split into a training and a testing set using the train\_test\_split function in sklearn.model\_selection. The standard split is typically 80% of your data for training and 20% for testing to avoid overfitting as testing is done on unseen data.

1. Model Initialization and Training

* Model Selection - create the Linear Regression model from the sklearn.linear\_model module; Linear regression is a simple yet powerful model for modeling the linear relationship between variables.
* Model Training: Fit the model to the training data using the fit method. The model learns the relationship between input features (X\_train) and the target variable (y\_train), estimates the coefficients that define this relationship.

1. Model Evaluation

* Make Predictions: Apply the trained model to make predictions on the test dataset (X\_test) through the predict function.
* Calculate Evaluation Metrics: Compute the performance of the model on the following metrics:
  + Mean Absolute Error (MAE): Average absolute difference between true and predicted values; very easy to interpret in terms of error magnitude.
  + Mean Squared Error (MSE): Average of squared errors, which gives more weight to the bigger errors.
  + Root Mean Squared Error (RMSE): The square root of the MSE, giving error measures in units consistent with the target variable.
  + R-squared (R²): Provides a measure of the amount of variance for the target variable explained by the model, and hence, its goodness-of-fit for the phenomenon at hand.
* Print Evaluation Metrics: Compute and display the evaluated metrics to determine at-a-glance what the model accomplishes.

1. Model Saving

* Save the Trained Model: Save a trained Linear Regression model using joblib to a file for later use, and then you can load it from the file later to make predictions instead of training it all over again.

1. Deployment and Future Considerations

* Model Deployment: Deploy the saved model into the Intelligent Application for Proficiency Enhancement. The user will be able to load the data for using in the function of obtaining the predictions of the total days required for proficiency enhancement.
* Continuous Monitoring: Monitor continuously the model with newer data input so that it does not become outdated and becomes correct. Retrain the model by upgrading the changes in user behavior or proficiency standards.

**Gradient Boosting Regression**

1. Dataset Collection:

* Collect a dataset relevant to language proficiency improvement, such as initial proficiency levels, session duration, improvement rates, and the total number of days required to reach a target proficiency level. These attributes can help predict how much time per day a learner needs to spend.

1. Data Preprocessing:

* Load the Dataset: Use the pandas library to load the dataset from an Excel file. The dataset includes features like "Initial Proficiency (%)" and "Total Days Required."
* Feature Selection:
  1. The independent variables (features) are:
     1. Initial Proficiency (%): The proficiency level at the start.
     2. Total Days Required: The target days to reach the desired proficiency.
  2. The target variable (dependent variable) is:
     1. Minutes Per Day: Time required each day to achieve the proficiency target.
* Scaling: Use StandardScaler from sklearn.preprocessing to scale the features for better performance of the Gradient Boosting model.

1. Splitting the Dataset:

* Split the data into training and testing sets using train\_test\_split from sklearn.model\_selection. Typically, an 80/20 split is used to ensure sufficient training while testing on unseen data to prevent overfitting.

1. Model Initialization and Training:

* Model Selection: Initialize a Gradient Boosting Regressor from sklearn.ensemble. Gradient boosting is powerful for handling non-linear relationships between variables.
* Hyperparameter Tuning: Use GridSearchCV to find the best model parameters, such as n\_estimators, learning\_rate, and max\_depth, by testing different combinations.
* Model Training: Fit the Gradient Boosting model to the training data using the best parameters identified from grid search.

1. Model Evaluation:

* Make Predictions: Use the trained model to predict Minutes Per Day on the test data.
* Calculate Evaluation Metrics: Assess the model's performance using:
  + Mean Squared Error (MSE): The average squared difference between actual and predicted values.
  + R-squared (R²): Measures how well the independent variables explain the variance in the target variable.
* Print Evaluation Metrics: Display these metrics to understand the model's effectiveness.

1. Model Saving:

* Save the trained Gradient Boosting model using joblib for future use, allowing predictions without retraining.

1. Deployment and Future Considerations:

* Model Deployment: Deploy the model in the Intelligent Application for Proficiency Enhancement, allowing users to input their initial proficiency and desired total days to receive a prediction on how much time they need to invest each day.
* Continuous Monitoring: Monitor the model’s performance with new data, retrain it periodically to adjust to changes in user behavior or proficiency standards, and ensure the model stays up-to-date.

**[VI]Development Process:**

1. Architecture:

* Frontend: The frontend is developed in HTML and CSS. For the interactive, client-side scripting of JavaScript is applied for interactive elements.
* Backend: Django, the Python web framework, will handle routing, database, and authentication. Deploying pre-trained models using Joblib.
* Database: MongoDB is planned to be used for the database to manage user profiles, proficiency scores, and session logs. Interface with the backend is planned to be done using PyMongo.

1. Database Design (MongoDB planned):

* Collections: Collections will have users, session details and their proficiency levels.
* Document Structure:  In MongoDB, while storing user progress user session length, proficiency level at the beginning, time for improvement, the flexible schema can be utilized to include fields such as these.
* Relationships: Since MongoDB is not relational, cross-links between collections like user profiles and session details need to be implemented when required using either references or embedded documents.

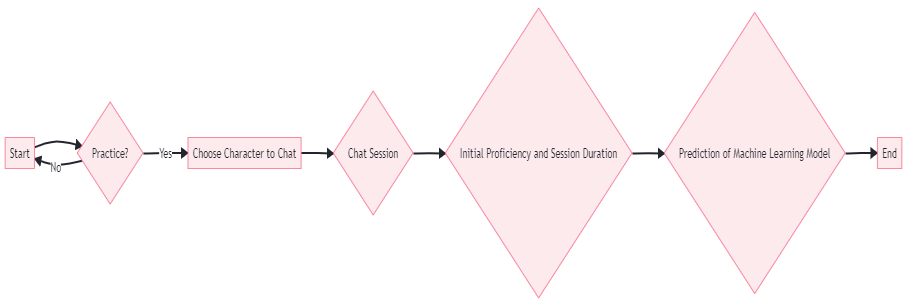
1. Frontend Development:

* HTML/CSS: designs the UI for different pages and makes them look good, interactive, and easy to use.
* Templates: Rather than rendering templates in the manner of Django, render dynamic HTML content using the outputs that were generated by pre-training a model and the interactions made by a user.
* Responsive design: Supports any device or screen for maximum performance.

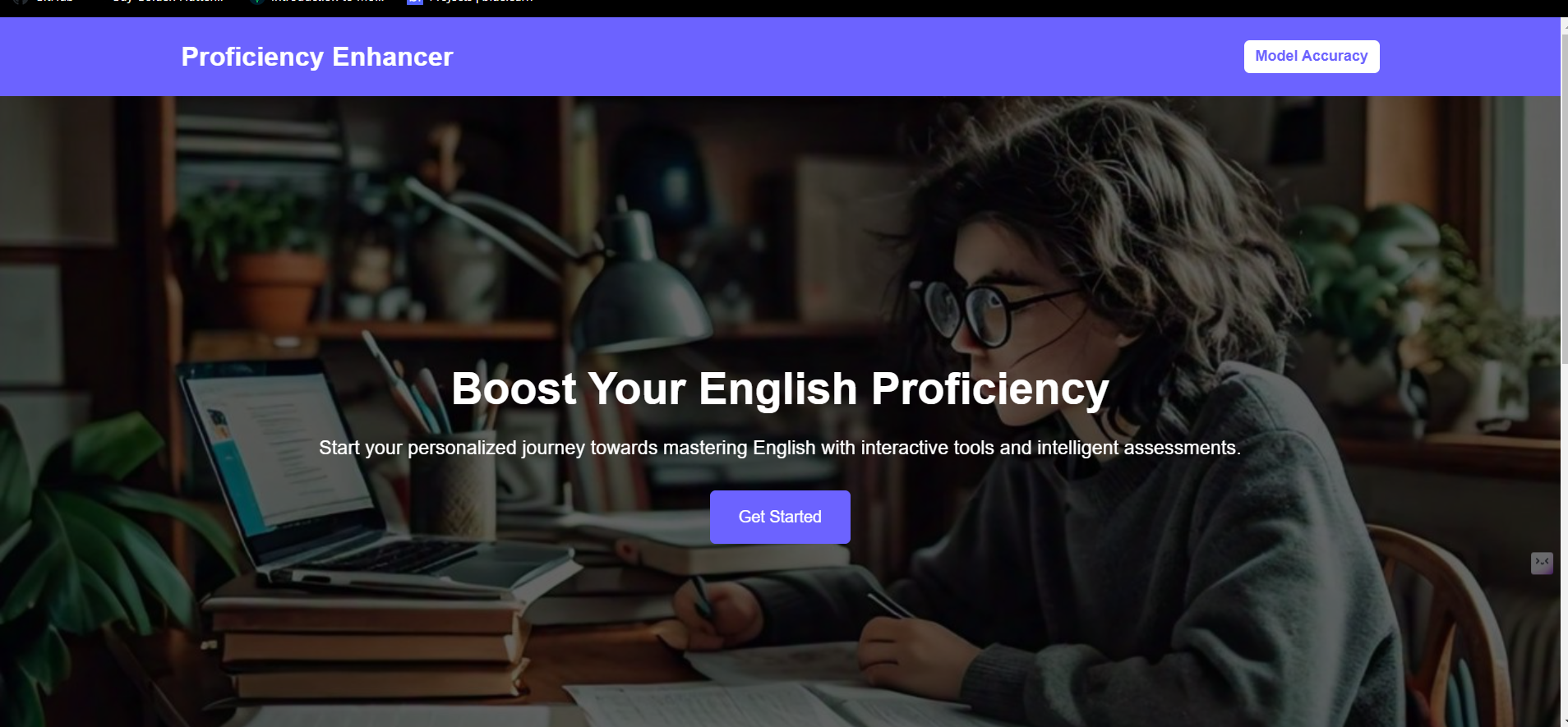
1. Backend Application:

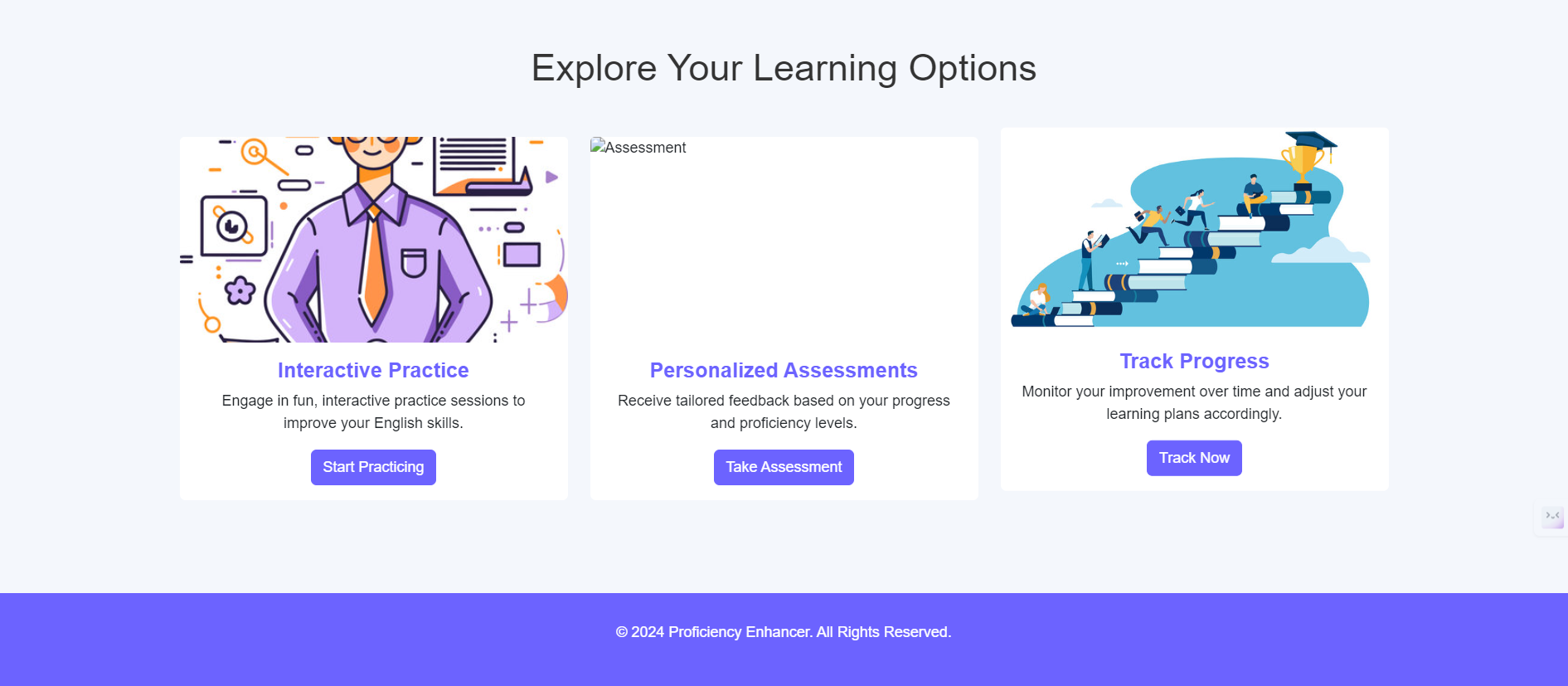
* Routing: The back-end application has specific routes defined in Django, which map HTTP requests to functions that can perform particular tasks or execute specific functionalities-processing the input of the user, making predictions, and storing results.
* Database Interaction (MongoDB): MongoDB will be connected with Django through PyMongo, and it will handle user data and session details. This connection will handle operations related to CRUD (Create, Read, Update, Delete).
* Error Handling: There are error handling techniques followed to catch and manage errors, thus providing meaningful error messages to the end-users.
* Middleware: Django middleware handles activities like request processing, logging, and handling errors for the robustness of the application.

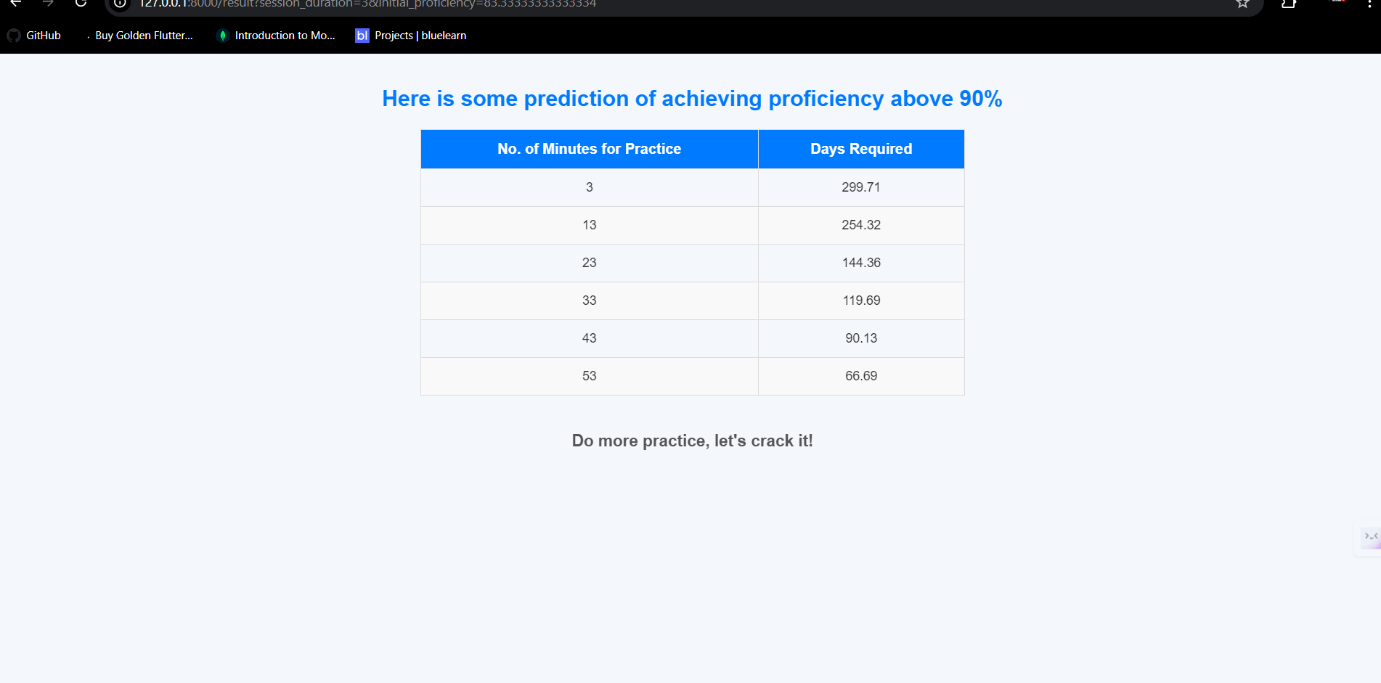
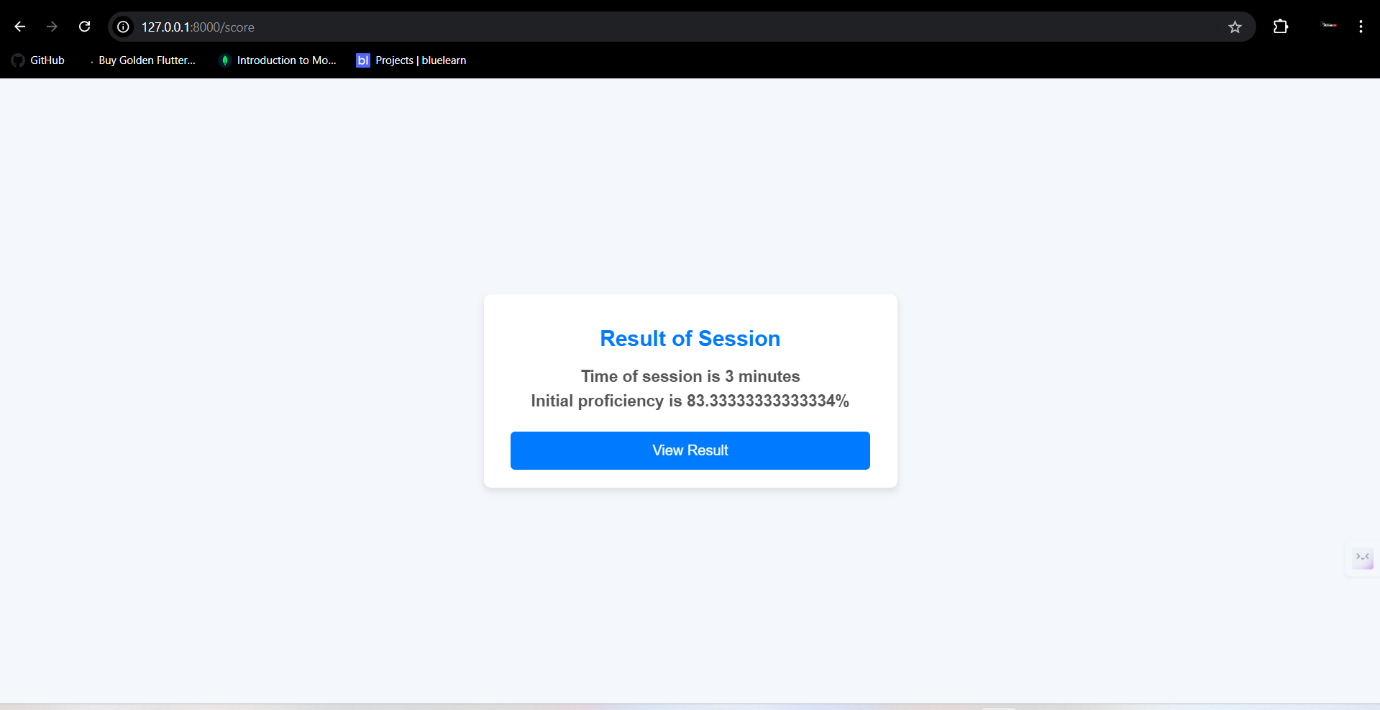
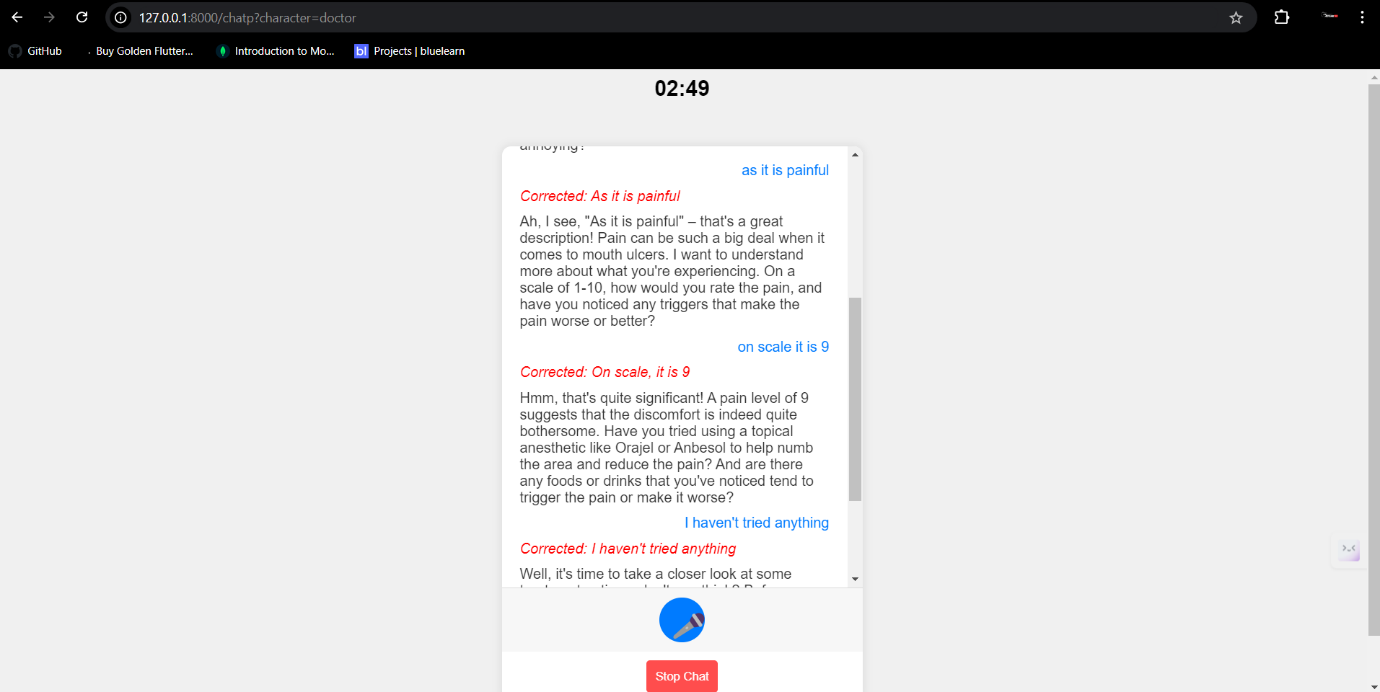
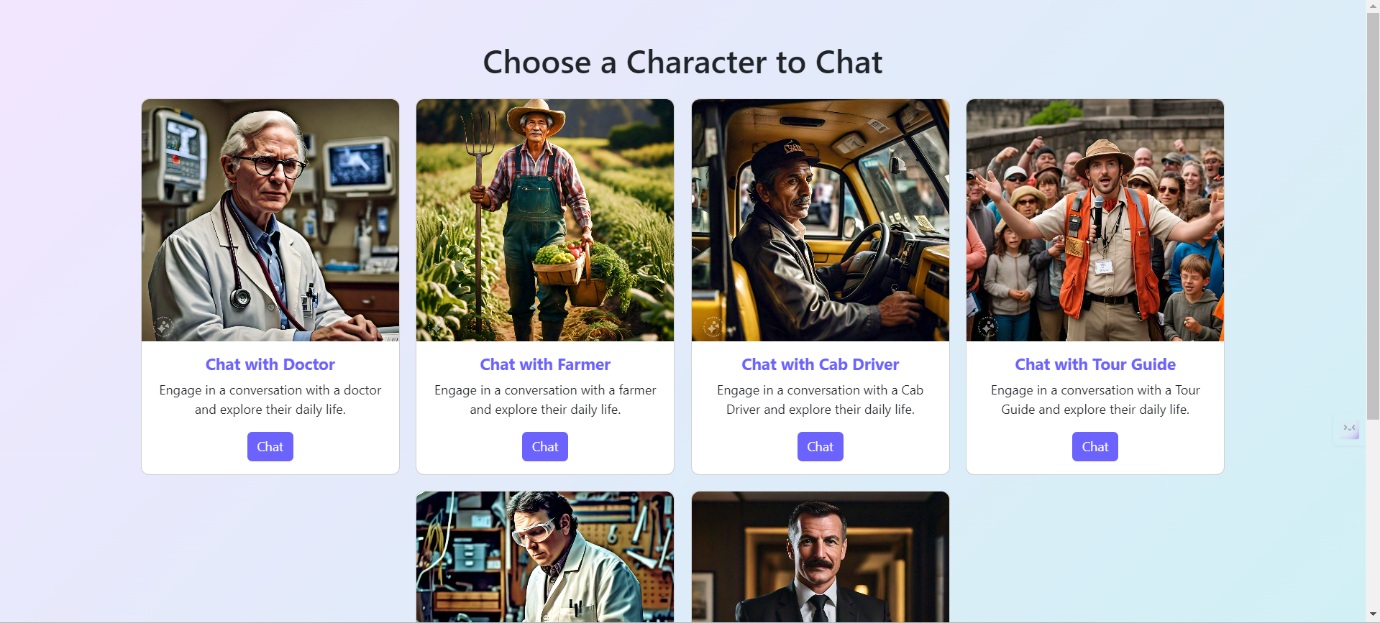
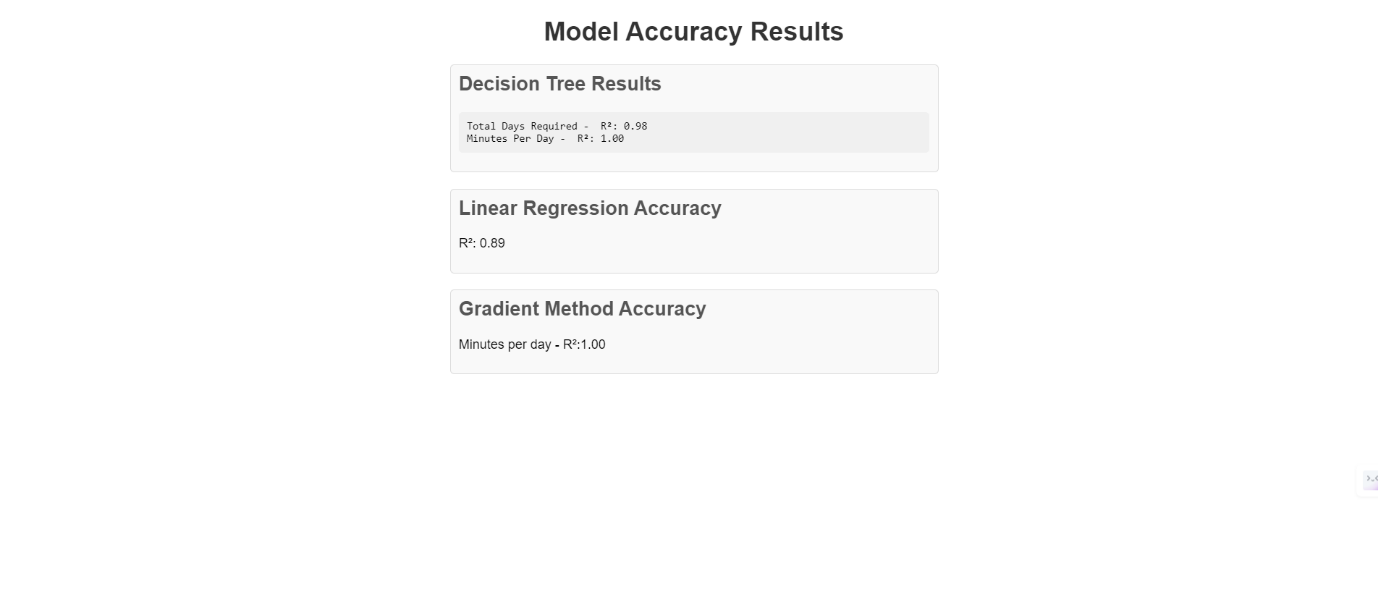
**[VII]Flow Diagram:**

****

**[VIII] Results and Output**

****

****

****

**[VIII]Future Scope**

The next scope of the project is to integrate MongoDB for scalable data management, high accuracy in predictions involving complex models such as random forests and neural networks, and to evolve into an adaptive learning system with real-time performance analysis. All these features - multilingual support, mobile application development, and gamification - can greatly promote engagement and accessibility with the system for users. In addition, external APIs for incorporating state-of-the-art speech recognition and pronunciation feedback capabilities as well as chat AI will be integrated to improve the functionality of the platform. Widening the base of learners covering a wide demographic range in addition to incorporating social learning features including peer assessment and group challenges will enhance learning further, making it comprehensive and collaborative. The project can also expand to become an all-in-one language learning platform by incorporating different levels of proficiency as well as different learning paths depending on the performance of the users. The future versions can involve artificial intelligence inputting content suggestions for users, contextual learning recommendations, and adaptive exercises driven by every user's progress. The addition of other languages will make the system more diverse and global-friendly. Multilateral collaborations will be enhanced through integration with language learning applications and educational platforms. Also, the reach of such a platform would expand with real-time analytics about the progress of users and detailed insights into areas for improvement that cultivate continuous learning in the user experience.

**[IX]Conclusion:**

Thus, this project can be termed as innovative towards the improvement of proficiency in the English language through the intelligent application of machine learning. Exploiting predictive models, like linear regression, this would allow the system to provide deeper insights on days of practice versus the amount of study time needed every day to reach the desired level of proficiency. The direct usage of speech-to-text and text-to-speech functionality will make it more like a conversation rather than an exercise to be completed. While this current implementation serves as a great base to stand on for further development, there is always room to make it more effective by including MongoDB for more effective data handling and AI-based personalized learning paths. This project clearly shows that technology can transform language learning into a personal, data-driven, and interactive experience.

**[IX]References:**

1. https://direct.mit.edu/tacl/article/doi/10.1162/tacl\_a\_00310/96485/Machine-Learning-Driven-Language-Assessment
2. Enhancing English Proficiency Test Evaluation: Leveraging Artificial Intelligence for Result Classification <https://ieeexplore.ieee.org/document/10458530>
3. Relationship between English Listening Proficiency and Cognitive Styles under the Web-based Instructional Model <https://ieeexplore.ieee.org/document/9534522>
4. Cross-Lingual Transfer Learning in NLP: Enhancing English Language Learning for Non-Native Speakers <https://ieeexplore.ieee.org/document/10544031>
5. Creating a Machine Learning Based Web Application Using Django <https://towardsdatascience.com/creating-a-machine-learning-based-web-application-using-django-5444e0053a09>
6. Decision tree regression algorithm <https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/>
7. Linear regression algorithm <https://www.geeksforgeeks.org/ml-linear-regression/>
8. Gradient boosting method <https://www.geeksforgeeks.org/ml-gradient-boosting/>
9. Crew ai documention <https://docs.crewai.com/introduction>
10. Language tool python <https://pypi.org/project/language-tool-python/>