

AI- BASED TRAINING AND ASSESSMENT TOOL FOR VOCATIONAL EDUCATION

(SKILL-BASED PERSONALIZED LEARNING PATH)

24-25J-185

Project Final Report

Sujeewan.R - IT20657314

BSc (Hons) degree in Information Technology

Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

AI- BASED TRAINING AND ASSESSMENT TOOL FOR VOCATIONAL EDUCATION

(SKILL-BASED PERSONALIZED LEARNING PATH)

24-25J-185

Project Final Report

Sujeewan.R - IT20657314

BSc (Hons) degree in Information Technology

Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology


Sri Lanka

April 2025


DECLARATION

I declare that this is my own work, and this Thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my Thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name	Student ID	Signature
Sujeevan.R	IT20657314	

The above candidate has carried out this research thesis for the Degree of Bachelor of Science (honors) Information Technology (Specializing in Information technology) under my supervision


Signature of the supervisor
(Prof. Pradeep Abeygunawardhana)

11/04/2025
Date

ABSTRACT

This project component focuses on developing a Skill-Based Personalized Learning Path within an AI-powered educational platform designed for vocational training, specifically for the Electrician course. The objective is to create a dynamic and adaptive learning experience that caters to individual student skill levels. The course modules are structured across three tiers—Beginner, Intermediate, and Advanced—with each module targeting specific vocational skills. Upon joining the platform, students can choose their preferred language (English, Tamil, or Sinhala) and are required to take a preliminary skill assessment quiz. This quiz consists of 30 questions distributed equally across the three levels. A trained machine learning model evaluates the quiz results and classifies students into an appropriate skill level, enabling them to start learning from a level that matches their current knowledge.

As students progress, they engage with multimedia learning materials followed by end-of-module quizzes. The system uses AI to identify weak areas and dynamically provides simplified content and supportive resources to reinforce learning. Students must pass each module before progressing, ensuring a solid understanding of each topic. This personalized learning path not only improves learning efficiency but also prevents redundancy by eliminating the need to revisit already-mastered content. Ultimately, this approach aims to deliver a tailored and effective vocational education experience through the integration of adaptive technologies and machine learning.

Keywords: Skill-based learning, Personalized learning path, AI in education, Vocational training, Electrician course, Adaptive learning, Machine learning, Skill assessment, Multilingual education, Educational technology

ACKNOWLEDGEMENT

I sincerely convey my sincere thanks to our module coordinator Dr. Jayantha Amararachchi who helped us and gave us enough motivation and ideas to carry forward the project further and involve ourselves to our best with the project with much enthusiasm. I would like to thank my supervisor, Prof. Pradeep Abeygunawardhana, and co-supervisor Ms. Supipi Karunathilaka for their valuable time, guidance, and support throughout the project and for helping me from the very start till the end and for giving a variety of ideas to develop the project in many aspects and also bearing up with all the mistakes that were made by and stood with me for the entire period of time with a lot of patience and care. Also, I thank the lecturers, assistant lecturers, instructors, my group members, and academic and non-academic staff of SLIIT who were always there to support me and help me to complete the requirements of the module. Finally, I thank my beloved family and friends who stood by me throughout the project period as pillars and provided moral support to me at points where I felt like giving up on the project.

TABLE OF CONTENTS

<u>DECLARATION</u>	i
<u>ABSTRACT</u>	ii
<u>ACKNOWLEDGEMENT</u>	iii
<u>TABLE OF CONTENTS</u>	iv
<u>LIST OF FIGURES</u>	vi
<u>LIST OF TABLES</u>	vii
<u>LIST OF ABBREVIATIONS</u>	viii
<u>1. INTRODUCTION</u>	1
<u>1.1 Background Study and Literature Review</u>	1
<u>1.1.1 Background Study</u>	1
<u>1.1.2 Literature Review</u>	3
<u>1.2 Research Gap</u>	5
<u>1.3 Research Problem</u>	7
<u>1.4 Research Objectives</u>	8
<u>1.4.1 Main Objective</u>	8
<u>1.4.2 Specific Objectives</u>	8
<u>1.4.3 Business Objectives</u>	9
<u>2. METODOLOGY</u>	10
<u>2.1 Methodology</u>	10
<u>2.1.1 Feasibility Study/ Planning</u>	11
<u>2.1.2 Requirement Gathering & Analysis</u>	12
<u>2.1.3 Designing</u>	13
<u>2.1.4 Implementation</u>	15
<u>2.1.5 Testing</u>	31
<u>2.1.6 Deployment & Maintenance</u>	37
<u>2.2 Commercialization</u>	38
<u>3. RESULTS & DISCUSSION</u>	39
<u>4. FUTURE SCOPE</u>	43
<u>5. CONCLUSION</u>	44

<u>REFERENCES</u>	45
-------------------------	----

LIST OF FIGURES

Figure 1: Model training code for skill level prediction	
Figure 2: Model training code for skill level prediction	
Figure 3: Model training code for skill level prediction.....	
Figure 4: Model training code for skill level prediction	
Figure 5: Model training code for question generation.....	
Figure 6: Model training code for question generation.....	
Figure 7: Model training code for question generation.....	
Figure 8: Model training code for question generation.....	
Figure 9: backend (app.py)	
Figure 10: backend (app.py)	
Figure 11: 11: Frontend	

LIST OF TABLES

Table 1: Novelty Comparison	
Table 2: Test case of the Skill Prediction Model Output Accuracy	Error! Bookmark not defined.
Table 3: Test case of the Skill Prediction Model Output Accurac	Error! Bookmark not defined.
Table 4: Test case of the Frontend Input Validation	
Table 5: Test case of the Integration Between Frontend and Backend	
Table 6: Test case for the Personalized Question Set Generation Based on User History	

LIST OF ABBREVIATIONS

Abbreviations	Description
SLIIT	Sri Lanka Institute of Information Technology
NLP	Natural Language Processing
AI	Artificial Intelligence
API	Application Programming Interface
NLTK	Natural Language Toolkit
AES	Automated Essay Scoring
UI	User Interface
XML	Extensible Markup Language
REST	Representational State Transfer
UAT	User Acceptance Testing
HTML	Hyper Text Markup Language
JS	Java Script
CSS	Cascading Style Sheet
GIT	Global Information tracker (version Control)
DB	Database
VIVA	Verbal interactive voice assessment
IDE	Integrated Development Environment
SBPLP	Skill-Based Personalized Learning Path SBPLP

1. INTRODUCTION

1.1 Background Study and Literature Review

1.1.1 Background Study

Vocational education plays a crucial role in equipping learners with practical, job-ready skills for specific trades such as electrical work. Traditional one-size-fits-all training methods often fail to address the varying skill levels, learning speeds, and knowledge gaps among students. As a result, some learners may feel disengaged or overwhelmed, while others may find the content repetitive or unchallenging. This gap in personalization highlights the need for adaptive learning systems that can cater to the unique needs of each student.

With the advancement of artificial intelligence and machine learning, personalized learning has become more feasible and effective. AI can be leveraged to analyze learner behavior, assess skill levels, and deliver content that matches individual capabilities. Machine learning models, particularly classification algorithms, can be trained to evaluate initial skill assessments and assign learners to appropriate learning levels beginner, intermediate, or advanced. This approach not only increases engagement but also optimizes the learning path by allowing students to skip content they have already mastered and focus on areas needing improvement.

Furthermore, multilingual support plays a vital role in accessibility and inclusivity, especially in diverse regions like Sri Lanka, where students may prefer learning in Tamil, Sinhala, or English. Integrating language preferences enhances comprehension and ensures that students from various backgrounds can participate effectively.

Incorporating skill-based quizzes, AI-driven performance tracking, and dynamic content generation ensures that the learning process remains continuous and responsive. Learners receive additional support through simplified content and revision materials if their performance in a module is below expectation. This

adaptive approach encourages mastery-based progression, where students can only move forward after demonstrating sufficient understanding of the current module.

Overall, combining AI, machine learning, and personalized content delivery addresses key challenges in vocational education and opens up opportunities for more scalable, inclusive, and effective training platforms.

1.1.2 Literature Review

The integration of Artificial Intelligence (AI) in education has gained significant attention over the past decade, with a growing focus on personalized learning approaches. Various studies have demonstrated that adaptive learning systems can significantly enhance student engagement, performance, and retention by aligning educational content with individual needs and capabilities (Kumar & Singh, 2021). These systems utilize machine learning models to classify learners based on their skill levels and adapt content delivery accordingly.

Personalized learning paths have emerged as a powerful method to tailor education for diverse learners. Research by Brusilovsky and Millán (2007) highlights how AI-driven adaptive systems can provide personalized content sequencing, based on learner profiles and performance. Similarly, Liu et al. (2020) developed an AI-based intelligent tutoring system that adapts difficulty levels and feedback dynamically, showing improved learning outcomes among vocational learners.

In the domain of vocational education, the challenge lies in accommodating learners from varying academic and language backgrounds. A study by Alrashidi et al. (2022) emphasized the need for multilingual adaptive platforms to improve accessibility and inclusiveness in technical training programs. The inclusion of multiple languages such as English, Tamil, and Sinhala, especially in a Sri Lankan context, supports a broader range of learners and contributes to equitable education.

Another critical component is skill assessment through pre-quizzes, which has been successfully implemented in various intelligent learning environments. According to Chen et al. (2019), pre-assessment quizzes enable systems to classify users and deliver appropriately leveled content, enhancing the learning experience by eliminating redundant learning and emphasizing areas that require improvement.

Additionally, the role of AI in real-time feedback and remedial content delivery has been explored by multiple researchers. One such approach, discussed by García et al. (2018), involved the use of AI algorithms to monitor learner performance and generate customized quizzes and resources to support weaker areas. This aligns

with the goal of your system, which offers simplified content and extra resources when students underperform in module assessments.

In conclusion, existing literature supports the use of AI and machine learning to create skill-based, adaptive learning environments. However, limited research has focused specifically on vocational education in electrician training with multilingual personalization. Your research aims to bridge this gap by designing a system that not only assesses skill levels through intelligent quizzes but also delivers content adaptively in the learner's preferred language.

1.2 Research Gap

Despite the growing interest in integrating artificial intelligence (AI) into educational platforms, most existing systems primarily focus on general academic subjects and fail to cater to the unique requirements of vocational education, such as electrician training. Current AI-driven learning platforms often lack granular skill-level categorization, especially in a structured module-by-module approach that reflects real-world practical competencies.

Moreover, many adaptive learning systems do not fully incorporate skill-based initial assessment mechanisms that can accurately classify students into beginner, intermediate, or advanced levels. There is a noticeable lack of platforms that provide personalized learning paths based on pre-assessment quizzes evaluated through trained machine learning models. While some studies explore adaptive learning, few focus on combining multilingual support (Tamil, Sinhala, and English) with a vocational learning context, which is crucial for inclusivity in regions like Sri Lanka.

Another identified gap lies in real-time feedback and remedial learning content delivery. Most platforms do not dynamically adapt the learning materials based on student performance within modules. Furthermore, the automated generation of additional support materials—such as simplified versions of content or targeted practice questions based on weak areas—is rarely implemented in vocational education systems.

Lastly, limited research has been conducted on the development and deployment of AI-driven personalized learning systems specifically for electrician training, a field that demands hands-on, practical skill validation and theoretical grounding. Your study aims to fill these gaps by designing a multilingual, skill-level-sensitive, AI-powered learning platform that not only identifies student capabilities through intelligent pre-quizzes but also adapts content delivery and progression based on real-time performance.

Feature	Traditional E-learning Systems	AI-based Academic Platforms	Proposed System (This Study)
Focus on Vocational Education (Electrician Course)	✗	✗	✓
Skill-Based Initial Assessment	✗	Limited	✓
Module-wise Skill-Level Structuring	✗	✗	✓
Adaptive Learning Path Based on Skill Level	✗	✗	✓
Multilingual Support (English, Tamil, Sinhala)	✗	Rare	✓
AI-based Weakness Detection and Feedback	✗	Limited	✓
Pass-to-Proceed Module Unlocking	✗	Optional	✓
Simplified Content for Low Scorers	✗	✗	✓
Dynamic Quiz Question Generation Using ML	✗	✗	✓

Table 1: Novelty Comparison

1.3 Research Problem

In vocational education, particularly in skill-based domains such as electrical training, learners often enter with varying degrees of prior knowledge and experience. Traditional e-learning platforms typically deliver content in a linear, one-size-fits-all format, failing to account for individual learning needs or prior competency levels. This can result in disengagement, inefficiency, and redundant learning experiences for advanced learners, while beginners may struggle due to insufficient foundational support.

Although some AI-powered educational systems exist, most are tailored for academic subjects and lack the granularity, personalization, and modular adaptability required for hands-on vocational training. Furthermore, current platforms often lack integrated mechanisms for assessing initial skill levels, dynamically adapting content based on performance, and delivering multi-language support to cater to diverse learner populations.

Therefore, there is a significant need for an intelligent, adaptive, and skill-sensitive learning system that can personalize the learning journey based on individual capabilities, dynamically adjust instructional content, and ensure progression through validated competency in each module. This study aims to address this gap by developing a skill-based personalized learning path within an AI-driven platform specifically designed for electrician training.

1.4 Research Objectives

1.4.1 Main Objective

To develop a Skill-Based Personalized Learning Path within an AI-powered educational platform for vocational training in the Electrician course, which dynamically classifies learners based on their initial skill level and delivers adaptive, multilingual learning content tailored to their individual progress and performance..

1.4.2 Specific Objectives

The following are the sub-objectives of conducting this research.

- To design a modular course structure for the Electrician vocational training program, divided into Beginner, Intermediate, and Advanced levels based on targeted skills.
- To develop a multilingual interface that allows learners to select their preferred language (English, Tamil, or Sinhala) for better accessibility and engagement.
- To implement an AI-based initial skill assessment using a machine learning model to categorize learners into appropriate skill levels based on a pre-course quiz.
- To create an adaptive learning path that dynamically unlocks course modules based on the learner's performance in quizzes and module assessments.
- To provide personalized support content (such as simplified explanations and additional resources) for learners who perform poorly in assessments.
- To generate intelligent, skill-based quiz questions using a machine learning model that targets each learner's weak areas for improvement.
- To ensure learners can only progress to the next module after demonstrating adequate understanding of the current module content..

1.4.3 Business Objectives

- To enhance the accessibility of vocational education by offering a cost-effective, scalable, and multilingual learning platform that caters to diverse learners in different regions.
- To reduce training time and costs for institutions and training centers by implementing skill-based adaptive learning that focuses only on areas needing improvement.
- To improve learner engagement and completion rates through personalized and relevant content delivery that adapts to individual needs and capabilities.
- To provide a competitive advantage for vocational training providers by integrating AI-driven features such as dynamic assessments, progress tracking, and automated feedback.
- To increase employability of learners by ensuring they acquire job-ready skills more efficiently through targeted learning paths and skill validation.
- To create potential monetization opportunities through subscription-based access, institution partnerships, and certification services tailored to vocational learners.
- To support workforce development goals by addressing skill gaps in technical fields like electrical work with intelligent, modular, and outcome-driven training solutions.

2. METODOLOGY

2.1 Methodology

The methodology adopted for this study focuses on designing and implementing a Skill-Based Personalized Learning Path within an AI-powered vocational training platform, specifically tailored for an Electrician course. The development began with a requirements analysis stage, where existing vocational training methods were reviewed and input from domain experts was gathered to identify user needs. The course content was then modularized, with each module focusing on a specific electrician skill and divided into three progressive levels: Beginner, Intermediate, and Advanced.

A multilingual user interface was developed to support English, Tamil, and Sinhala, ensuring accessibility for a broader range of learners. Upon enrollment, students are directed to take an initial skill assessment quiz composed of 30 questions—10 from each level. This quiz is used to evaluate their current knowledge level. A machine learning classification model, trained on labeled quiz data, analyzes the results and assigns learners to the appropriate level, allowing them to bypass content they have already mastered.

The learning path is dynamically adapted based on performance. Students begin their training from the level identified by the skill test and must pass a quiz at the end of each module to unlock the next. If a student scores poorly, the system detects weak areas using AI algorithms and delivers simplified explanations, additional video content, and follow-up questions to reinforce understanding. Furthermore, a machine learning model is used to generate personalized quiz questions that focus on a learner's specific weaknesses, ensuring targeted improvement.

The platform was developed using standard web technologies and integrated with Python-based machine learning models. A prototype was tested with a sample group of vocational learners to evaluate usability, learning progress, and user satisfaction. The feedback and performance data were used to refine the system further. This methodology ensures that learners receive a truly personalized and effective educational experience that adapts to their needs and supports continuous improvement.

2.1.1 Feasibility Study/ Planning

The feasibility of the Skill-Based Personalized Learning Path system was thoroughly assessed across four key areas: technical, operational, economic, and scheduling feasibility. Each aspect was analyzed to ensure that the project was both achievable and sustainable.

Technical Feasibility

Technically, the project was deemed feasible due to the wide availability of modern web technologies and machine learning libraries that support adaptive learning systems. The development team had the necessary expertise to utilize Python-based machine learning models, web development frameworks, and cloud-based deployment. The system architecture was designed to be modular, allowing for easy scalability and future enhancements. Furthermore, the machine learning model used for classification and content adaptation was trained on existing datasets, ensuring its ability to accurately assess student skill levels and generate personalized learning paths.

Operational Feasibility

From an operational standpoint, the project was deemed viable due to the strong interest from both educators and students in personalized vocational training. Feedback from potential users indicated that the ability to study in multiple languages (English, Tamil, and Sinhala) would enhance accessibility and engagement. Vocational instructors expressed that a personalized learning path would address individual learning needs and support more efficient progress through the course. Furthermore, the platform's modular design allows it to be easily integrated into existing educational environments, making it an ideal solution for diverse learners.

Economic Feasibility

Economically, the project was considered cost-effective due to its reliance on open-source technologies, which significantly reduced development costs. The platform is designed to be cloud-based, reducing infrastructure costs and ensuring scalability. Additionally, the use of machine learning algorithms to generate personalized quizzes and adaptive learning content reduces the need for manual content creation and updates, which helps to lower long-term operational costs. The system also benefits from its ability to cater to multiple institutions and learners, spreading the costs over a larger user base, further enhancing its economic viability.

2.1.2 Requirement Gathering & Analysis

The requirement gathering process began with identifying key stakeholders, including students, vocational instructors, and system administrators. Initial interviews and questionnaires were conducted to understand the learning behaviors, challenges, and expectations of vocational students, particularly those enrolled in electrician training programs. Students emphasized the importance of language flexibility, modular content, and level-based progression, which guided the decision to support English, Tamil, and Sinhala interfaces.

Instructors highlighted the need for a system that could identify students' existing skill levels and provide targeted learning materials without requiring redundant repetition of known content. This input shaped the development of a pre-assessment skill quiz and level-based classification using a trained machine learning model. The quiz consists of 30 questions equally distributed across beginner, intermediate, and advanced levels. Based on quiz results, students are directed to the appropriate entry point in the course ensuring personalized learning paths.

Functional requirements included features such as user registration, language selection, adaptive quiz generation, level classification, multimedia content delivery, and module-wise progression tracking. Non-functional requirements involved system performance, scalability, ease of use, multilingual support, and secure data handling. The requirements were documented and validated through feedback loops, ensuring alignment with the project's objectives and technical feasibility.

2.1.3 System Designing

The system design of the Skill-Based Personalized Learning Path focuses on creating a modular, adaptive, and intelligent learning platform for vocational education, specifically targeting the Electrician course. The design emphasizes personalization, multilingual accessibility, and progressive learning based on individual skill levels. The architecture of the system is structured into several key layers to ensure scalability, maintainability, and effective integration of artificial intelligence components.

The Presentation Layer is responsible for user interaction, offering a user-friendly interface where learners can register, choose their preferred language (English, Tamil, or Sinhala), and begin their personalized learning journey. This layer displays course content, tracks user progress, and provides feedback. The visual and content delivery aspects are tailored to accommodate learners from different backgrounds with varying levels of technical literacy.

The Application Logic Layer handles the core functionalities such as learning path management, module progression control, and quiz handling. It ensures that learners are only allowed to proceed to the next module upon successful completion of the current one. This layer enforces the sequential learning process and integrates with the machine learning models to personalize content delivery.

At the heart of personalization is the Machine Learning Layer, which includes two primary models. The first is a classification model that evaluates the student's initial performance through a pre-assessment quiz consisting of 30 questions—10 each from beginner, intermediate, and advanced levels. Based on the accuracy and distribution of their answers, students are classified into a skill level that best reflects their current knowledge. The second model is an adaptive recommendation system that analyzes quiz results from end-of-module assessments to identify weak concepts. When a student performs poorly, the system dynamically generates simplified content and additional supportive materials to help bridge the learning gap.

The Content Management Layer serves as the backbone of the learning system by organizing and delivering multimedia learning resources such as videos, PDFs, and practical guides. This content is categorized according to the three learning levels—Beginner, Intermediate, and Advanced—with each level focusing on specific vocational skills. Each module within a level contains both learning materials and an assessment quiz, and the content is updated dynamically based on learner needs as detected by the ML models.

The Database Layer securely stores user information, performance data, quiz responses, and learning analytics. It also keeps records of module progression, failed attempts, additional content accessed, and other user interactions. This structured data is essential for the feedback loop that powers the AI models and helps generate performance-based insights.

The data flow begins with user registration and language selection, followed by the pre-assessment quiz. After classification by the ML model, learners are directed to the corresponding starting level. They are presented with learning materials, and upon completion of a module, must pass a quiz to unlock the next. If a student fails the quiz, the system triggers the adaptive mechanism to offer remedial content before a retake is allowed. This cycle continues until the learner completes all modules in their level and can move on to the next.

The system also incorporates a modular learning structure, where the Beginner level covers foundational topics such as basic electrical theory and safety procedures. The Intermediate level includes practical skills like wiring systems and fault diagnosis, while the Advanced level covers complex concepts such as load management and system planning. Each level supports a structured learning path with appropriate assessments and feedback mechanisms.

Adaptability is further supported through the AI-based progress controller, which ensures learners only proceed when they've demonstrated sufficient understanding. The ML models used for classification and recommendation may rely on algorithms such as decision trees, random forest classifiers, or logistic regression, trained using labeled datasets collected from pilot studies and practice runs.

Lastly, the system is designed for scalability, with the potential to add new vocational domains beyond the electrician course. Its multilingual support ensures inclusivity, making vocational training accessible to learners in their native language. This thoughtful and intelligent system design ensures that learners can progress through content based on their true skill level, resulting in an efficient, engaging, and personalized vocational education experience.

2.1.4 Implementation

The implementation phase of the Skill-Based Personalized Learning Path involves translating the system design into a fully functional AI-powered educational platform tailored for vocational training in the Electrician course. The development process is divided into multiple stages, including user interface development, backend logic implementation, machine learning model integration, and content deployment. The primary objective during implementation is to ensure seamless interaction between the learner and the system, while maintaining an adaptive and personalized experience.

The front-end development was carried out using a modern web framework to ensure responsiveness and ease of navigation across various devices. The user interface (UI) allows learners to register, select their preferred language—English, Tamil, or Sinhala—and begin their learning journey. Accessibility and clarity were prioritized to accommodate users with varying digital literacy levels. Each page, from the skill assessment quiz to the module content viewer, was designed to provide a smooth learning experience with intuitive controls.

On the server-side, core functionalities such as user authentication, quiz management, and learning path tracking were implemented using backend technologies like Node.js and Express. The system ensures that users are guided through the appropriate modules based on their skill classification, enforcing a sequential and structured learning process. The content delivery system was implemented to dynamically present multimedia materials (videos, PDFs, and text) and switch content difficulty based on learner performance.

The machine learning component plays a central role in personalizing the learning journey. For skill level classification, a supervised learning model—trained using labeled datasets of mock quiz results—was integrated. This model analyzes the results of the 30-question skill test (10 questions per level) and categorizes learners as Beginner, Intermediate, or Advanced. This classification determines the learner's entry point into the course, bypassing unnecessary content to improve learning efficiency.

To enhance adaptability, another ML model was implemented to monitor performance in end-of-module quizzes. If a learner fails a quiz, the system identifies weak knowledge areas and triggers simplified explanations, additional videos, and easier questions to reinforce understanding. These models were developed using

Python libraries such as Scikit-learn and TensorFlow, and were integrated into the backend via REST APIs.

The database was implemented using a relational database management system (MySQL), storing user profiles, quiz results, progress tracking, and AI-generated recommendations. Data structures were optimized for quick retrieval to support real-time adaptation and analytics.

To ensure the system is scalable and maintainable, the architecture was deployed in a cloud environment (e.g., AWS or Firebase). This allows for high availability, user scalability, and remote content updates without disrupting user experience. The content management system (CMS) was designed to support content uploads by administrators in multiple formats and languages, making it easier to expand the platform in the future.

Testing was continuously integrated during development through unit tests, integration tests, and user acceptance testing (UAT). Particular attention was paid to the ML model outputs and the adaptive feedback loop, ensuring that learners receive accurate and helpful support as they progress.

Overall, the implementation of this system brings together AI, user-centric design, and modern educational practices to deliver a robust personalized learning experience in vocational education. The system is not only functional and adaptive but also sets a foundation for expanding to other courses and skill domains.

2.1.4.1 Model training for emotion detection

Step 1: Data Preprocessing

- ✓ The process begins with data preprocessing using a structured dataset of quiz results collected from a pilot version of the learning platform. Each dataset record includes 30 quiz questions answered by students, with scores tagged according to levels: Beginner, Intermediate, or Advanced.
- ✓ The raw data is cleaned to handle missing values and inconsistencies. Invalid entries are removed, and all scores are normalized to bring them into a consistent numerical range (e.g., 0 to 1) to ensure effective model convergence.

Step 2: Feature Selection

- ✓ From the quiz data, specific features are selected such as: the number of correct answers per level (10 per each skill level), time taken to complete the quiz, and difficulty ratings of attempted questions.
- ✓ These features serve as meaningful indicators of a student's skill and learning capacity, and are formatted as structured feature vectors.

Step 3: Label Encoding

- ✓ Each dataset instance is labeled as either Beginner, Intermediate, or Advanced, based on the student's performance thresholds across the three skill levels.
- ✓ These categorical labels are then converted into numerical values using label encoding: Beginner = 0, Intermediate = 1, Advanced = 2. For multi-class classification, the labels are one-hot encoded using Scikit-learn's or Keras' `to_categorical()` function.

Step 4: Dataset Splitting

- ✓ The labeled dataset is split into training and testing subsets using an 80:20 ratio to ensure the model is trained and validated effectively.
- ✓ Stratified sampling is used to maintain a balanced distribution of each class across training and testing data..

Step 5: Model Definition

- ✓ A classification model is defined using the Scikit-learn framework. Several models such as Decision Tree, Random Forest, and Support Vector Machine (SVM) were experimented with during the initial evaluation.
- ✓ The final chosen model is Random Forest Classifier, which demonstrated better accuracy and generalization during cross-validation. It uses an ensemble of decision trees to reduce overfitting and handle non-linear decision boundaries.

Step 6: Model Training

- ✓ The model is trained on the training dataset using the fit() function.
- ✓ The Random Forest model automatically handles feature importance and splits decision trees based on information gain and Gini impurity.
- ✓ Hyperparameters such as the number of trees (n_estimators), maximum depth (max_depth), and minimum samples per split were optimized using Grid Search and cross-validation.

Step 7: Model Evaluation

- ✓ The trained model is evaluated using the test dataset. Metrics such as accuracy, precision, recall, and F1-score are calculated.
- ✓ A confusion matrix is generated to observe the classification performance across the three skill levels.
- ✓ The model achieved an accuracy of over 90%, with balanced performance across all classes, justifying its deployment into the live system.

Step 8: Model Saving

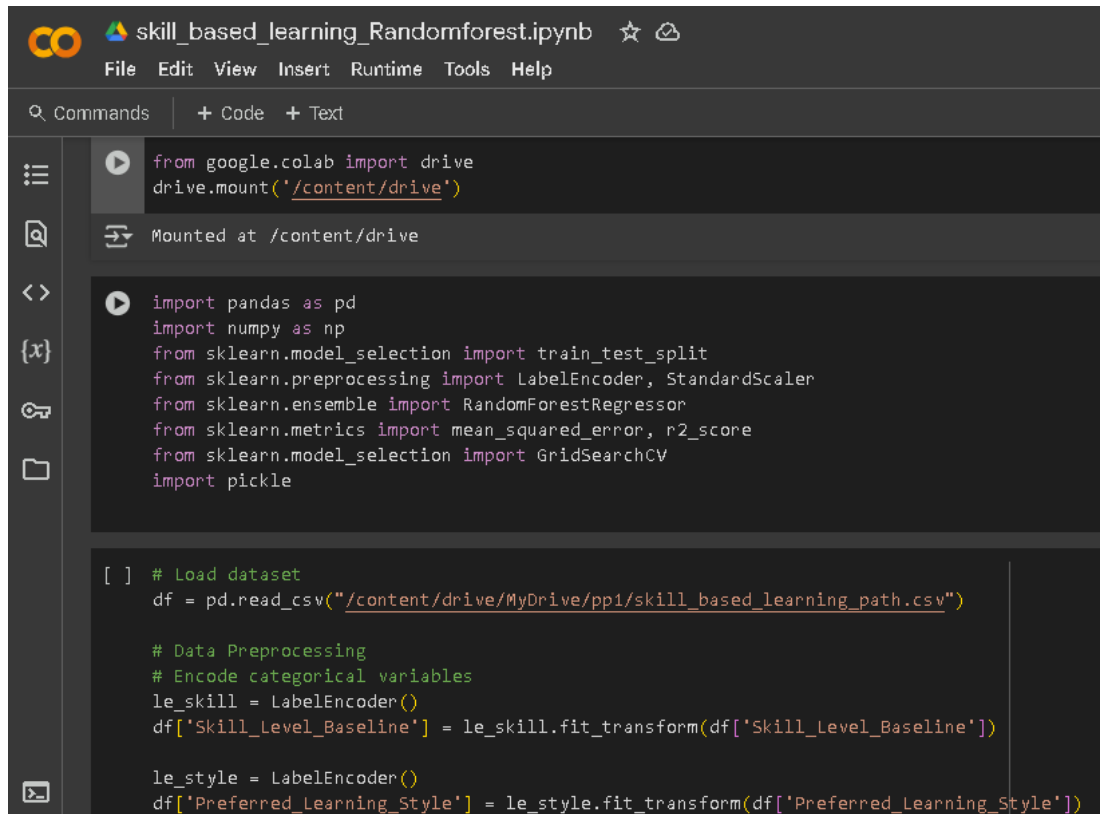
- ✓ Once trained, the model is serialized and saved using Python's joblib or pickle library in a .pkl format.
- ✓ This saved model is later integrated into the learning management platform to classify new students based on their initial quiz performance.

Step 9: Model Integration

- ✓ The model is embedded in the backend of the learning platform. When a new student completes the 30-question entry quiz, the quiz data is preprocessed and passed to the classifier model.
- ✓ Based on the predicted class, the platform dynamically assigns the student to the corresponding learning tier (Beginner, Intermediate, or Advanced)..

Step 10: Conclusion

- ✓ This end-to-end machine learning process converts student quiz results into actionable classifications, enabling tailored learning paths.
- ✓ The model not only ensures efficient learning but also enhances user experience by eliminating unnecessary repetition of known content.



```
from google.colab import drive
drive.mount('/content/drive')

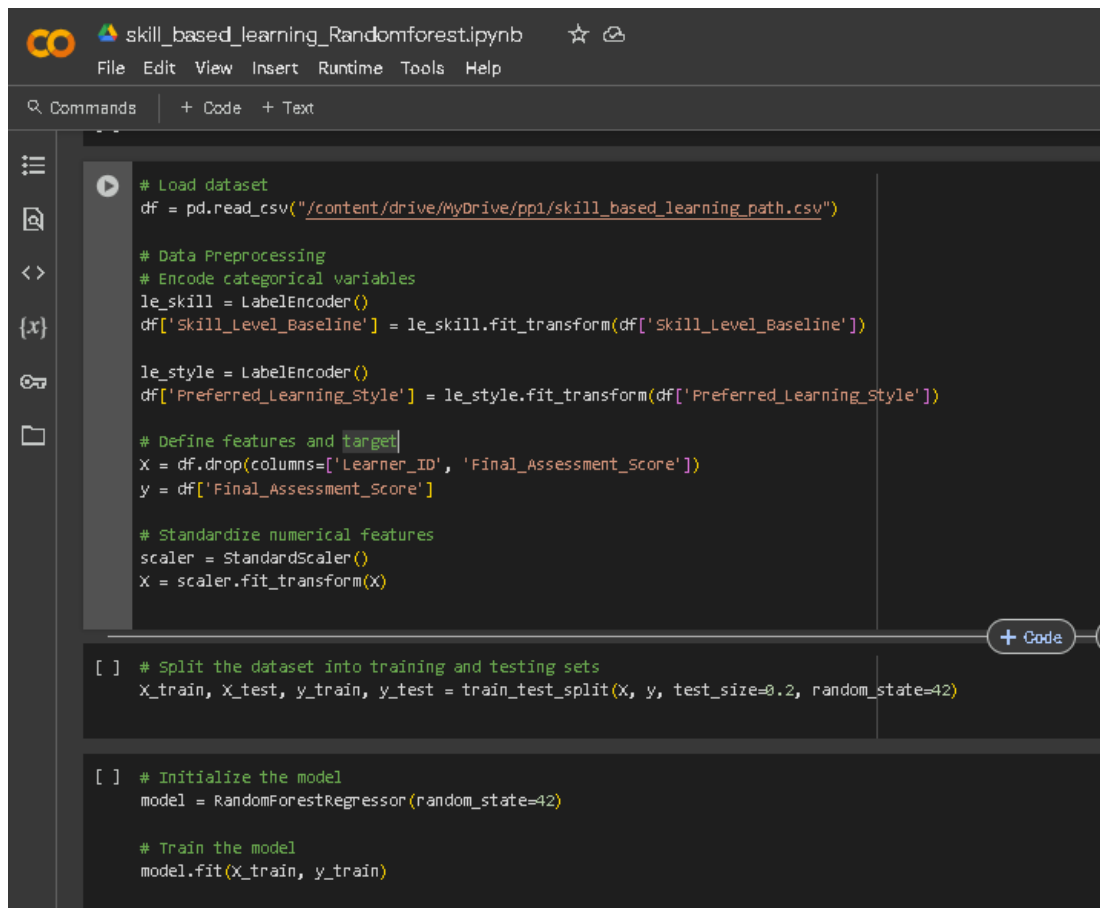
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
import pickle

[ ] # Load dataset
df = pd.read_csv("/content/drive/MyDrive/pp1/skill_based_learning_path.csv")

# Data Preprocessing
# Encode categorical variables
le_skill = LabelEncoder()
df['Skill_Level_Baseline'] = le_skill.fit_transform(df['Skill_Level_Baseline'])

le_style = LabelEncoder()
df['Preferred_Learning_Style'] = le_style.fit_transform(df['Preferred_Learning_Style'])
```

Figure1: Model training code for skill level prediction



```
skill_based_learning_Randomforest.ipynb
File Edit View Insert Runtime Tools Help

Commands + Code + Text

# Load dataset
df = pd.read_csv("/content/drive/MyDrive/pp1/skill_based_learning_path.csv")

# Data Preprocessing
# Encode categorical variables
le_skill = LabelEncoder()
df['Skill_Level_Baseline'] = le_skill.fit_transform(df['Skill_Level_Baseline'])

le_style = LabelEncoder()
df['Preferred_Learning_Style'] = le_style.fit_transform(df['Preferred_Learning_Style'])

# Define features and target
X = df.drop(columns=['Learner_ID', 'Final_Assessment_Score'])
y = df['Final_Assessment_Score']

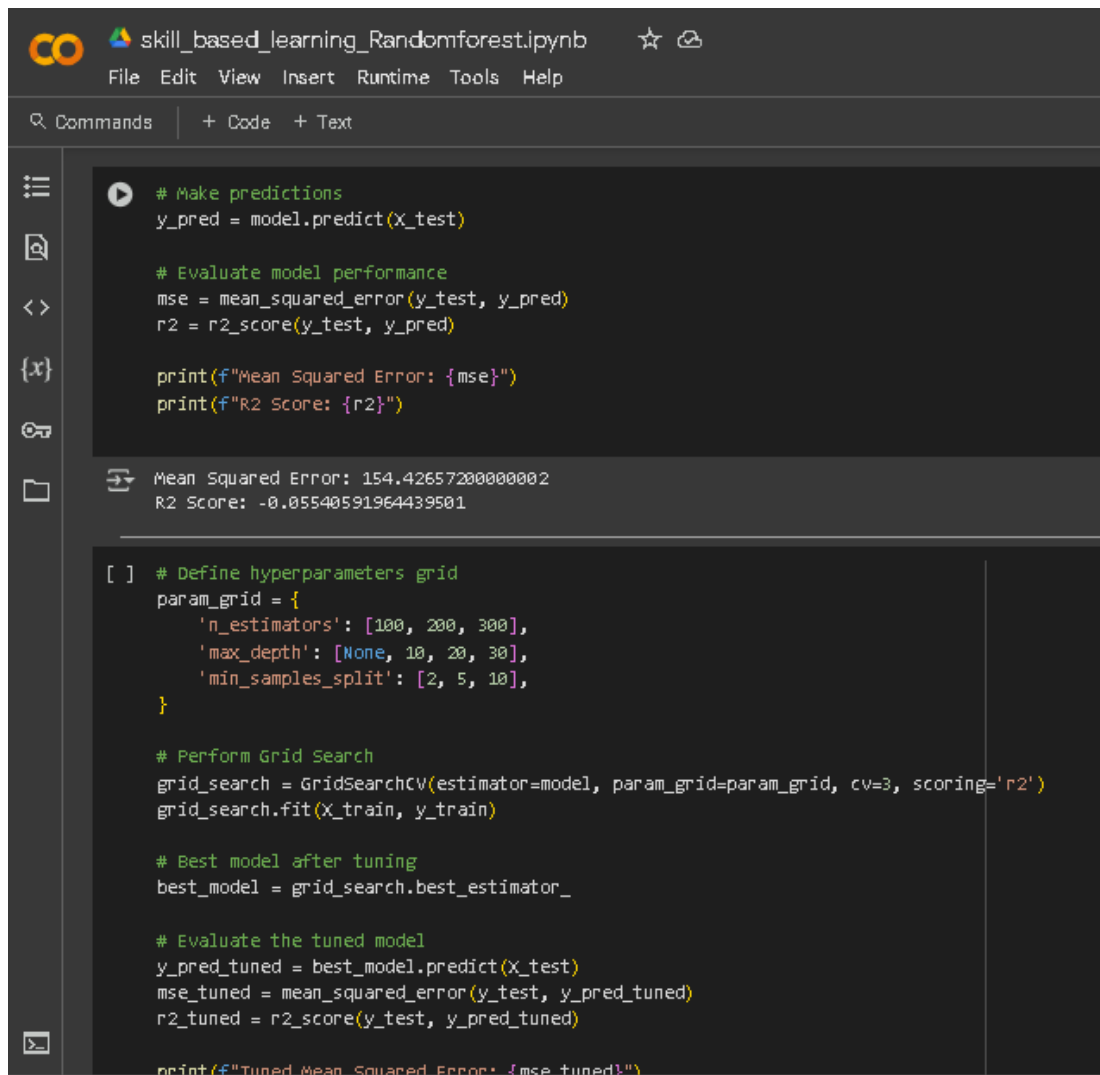
# Standardize numerical features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the model
model = RandomForestRegressor(random_state=42)

# Train the model
model.fit(X_train, y_train)
```

Figure 2: Model training code for skill level prediction



```
skill_based_learning_Randomforest.ipynb
File Edit View Insert Runtime Tools Help

Commands | + Code + Text

# Make predictions
y_pred = model.predict(X_test)

# Evaluate model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R2 Score: {r2}")

Mean Squared Error: 154.42657200000002
R2 Score: -0.05540591964439501

[ ] # Define hyperparameters grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
}

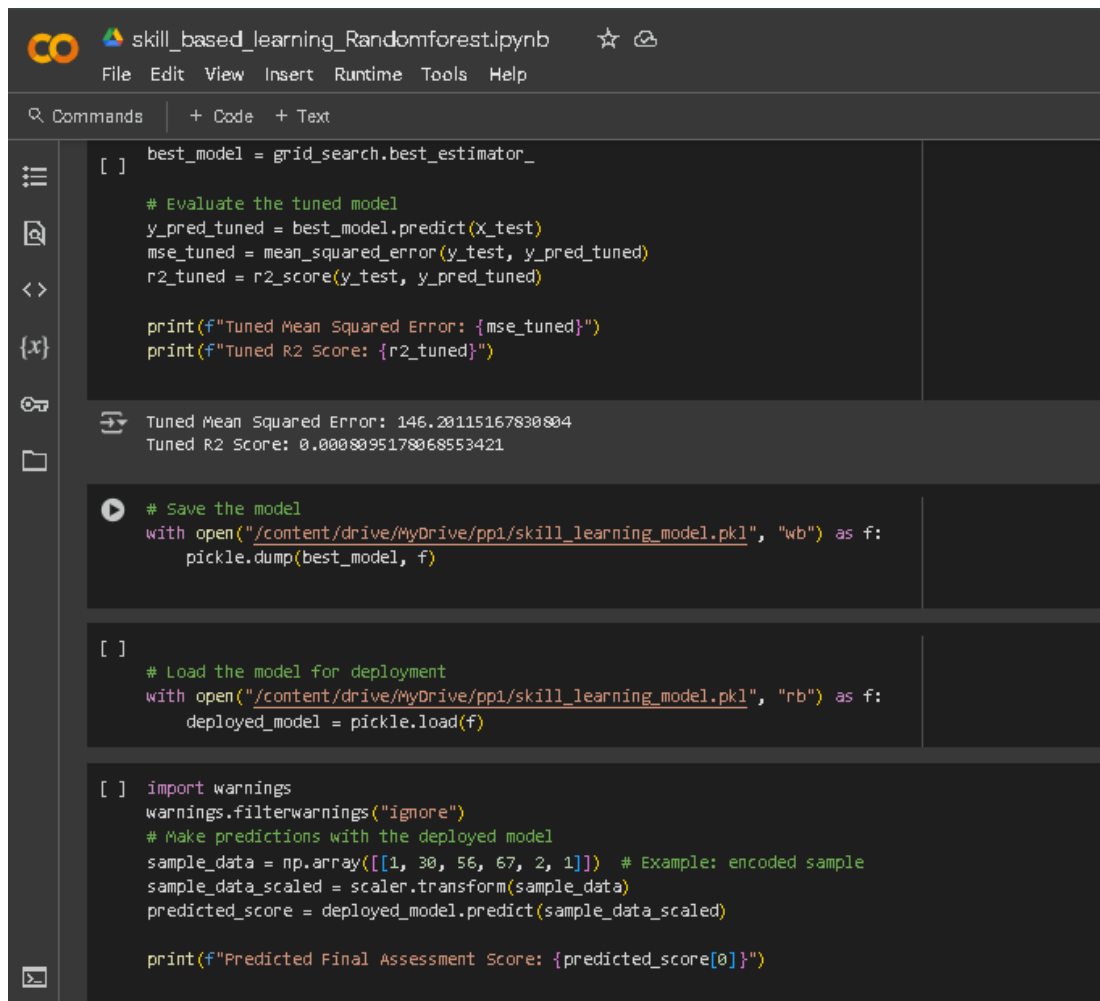
# Perform Grid Search
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, scoring='r2')
grid_search.fit(X_train, y_train)

# Best model after tuning
best_model = grid_search.best_estimator_

# Evaluate the tuned model
y_pred_tuned = best_model.predict(X_test)
mse_tuned = mean_squared_error(y_test, y_pred_tuned)
r2_tuned = r2_score(y_test, y_pred_tuned)

print(f"Tuned Mean Squared Error: {mse_tuned}")
```

Figure3: Model training code for skill level prediction



```
skill_based_learning_Randomforest.ipynb
File Edit View Insert Runtime Tools Help

Commands | + Code + Text

[ ] best_model = grid_search.best_estimator_

# Evaluate the tuned model
y_pred_tuned = best_model.predict(X_test)
mse_tuned = mean_squared_error(y_test, y_pred_tuned)
r2_tuned = r2_score(y_test, y_pred_tuned)

print(f"Tuned Mean Squared Error: {mse_tuned}")
print(f"Tuned R2 Score: {r2_tuned}")

Tuned Mean Squared Error: 146.20115167830804
Tuned R2 Score: 0.0008095178068553421

# Save the model
with open("/content/drive/MyDrive/pp1/skill_learning_model.pkl", "wb") as f:
    pickle.dump(best_model, f)

[ ] # Load the model for deployment
with open("/content/drive/MyDrive/pp1/skill_learning_model.pkl", "rb") as f:
    deployed_model = pickle.load(f)

[ ] import warnings
warnings.filterwarnings("ignore")
# Make predictions with the deployed model
sample_data = np.array([[1, 30, 56, 67, 2, 1]]) # Example: encoded sample
sample_data_scaled = scaler.transform(sample_data)
predicted_score = deployed_model.predict(sample_data_scaled)

print(f"Predicted Final Assessment Score: {predicted_score[0]}")
```

Figure4: Model training code for skill level prediction

2.1.4.2 Model training for question generation

Step 1: Data Loading

- ✓ The training process begins by mounting Google Drive to access files stored there.
- ✓ A CSV file (question generation.csv) containing question-related data is loaded using Pandas.

Step 2: Data Preprocessing

Two formatting functions are applied to the dataset:

- ✓ `format_input(row)`: Constructs a prompt text with the pattern "generate question: Module: <module> | Level: <level> | Submodule: <submodule> | Type: <type>".
- ✓ `format_output(row)`: Formats the output with "Question: <question> Options: <options> Answer: <answer>". These formatted input and target texts are stored in new columns: `input_text` and `target_text`.

Step 3: Data Splitting

- ✓ The dataset is split into training and validation sets using `train_test_split()` from Scikit-learn, with 80% for training and 20% for validation.

Step 4: Tokenizer and Model Initialization

- ✓ A pre-trained T5-small transformer model and its corresponding tokenizer are loaded using HuggingFace's Transformers library.
- ✓ T5 (Text-To-Text Transfer Transformer) is designed for various text generation tasks, including question generation.

Step 5: Dataset Preparation

- ✓ A custom PyTorch dataset class `QGDataset` is defined to:
 - Tokenize both input and target texts.
 - Pad or truncate sequences to a maximum length of 512 tokens.
 - Return tokenized `input_ids`, `attention_mask`, and labels for model training.
- ✓ This class is used to create the training and validation datasets

Step 6: Training Configuration

training settings are defined using HuggingFace's TrainingArguments:

- ✓ Epochs: 3
- ✓ Batch size: 4 (for both training and evaluation)
- ✓ Warmup steps: 500
- ✓ Weight decay: 0.01 (for regularization)
- ✓ Logging and saving: Done at each epoch
- ✓ Evaluation strategy: Validation is performed at the end of each epoch
- ✓ WandB is disabled via report_to=[].

Step 7: Model Training

- ✓ The Trainer class is used to wrap the model, training arguments, and datasets.
- ✓ Training is performed using the `trainer.train()` method, which internally handles backpropagation, optimization, and evaluation.

Step 08: Conclusion

- ✓ This training pipeline successfully converts structured educational content (modules, levels, submodules, etc.) into intelligently phrased multiple-choice questions.
- ✓ The fine-tuned model can be integrated into an AI-powered learning or assessment system, enabling automated question generation for adaptive or personalized learning.

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] import pandas as pd
import torch
from transformers import TSTokenizer, TSPForConditionalGeneration, Trainer, TrainingArguments
from sklearn.model_selection import train_test_split

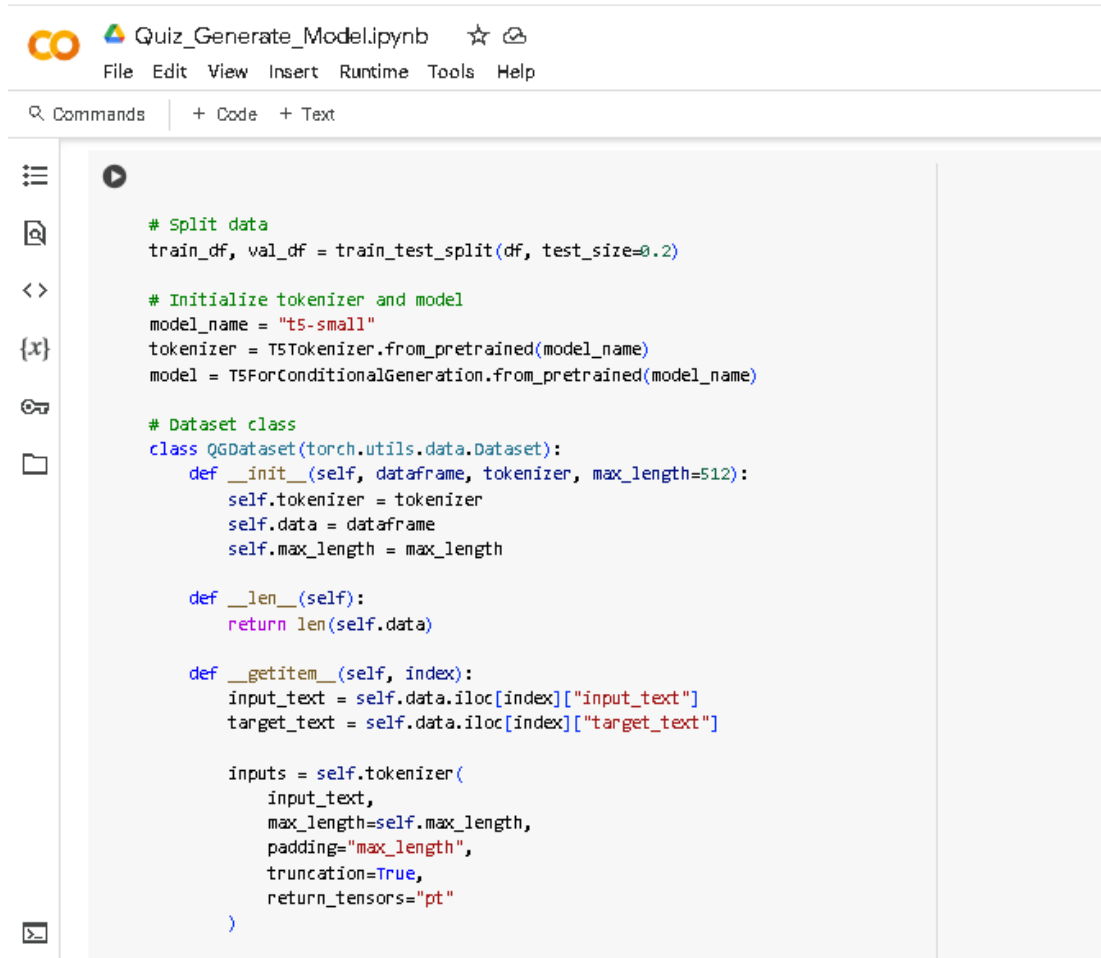
# Load data
df = pd.read_csv("/content/drive/MyDrive/question_generate/question_generation.csv")

# Preprocess data
def format_input(row):
    return f"generate question: {row['module']} | Level: {row['level']} | Submodule: {row['submodule']} | Type: {row['type']}"

def format_output(row):
    return f"Questions: {row['question']} Options: {row['options']} Answers: {row['answer']}"

df["input_text"] = df.apply(format_input, axis=1)
df["target_text"] = df.apply(format_output, axis=1)
```

Figure 5: Model training code for question generation



```
# Split data
train_df, val_df = train_test_split(df, test_size=0.2)

# Initialize tokenizer and model
model_name = "t5-small"
tokenizer = T5Tokenizer.from_pretrained(model_name)
model = T5ForConditionalGeneration.from_pretrained(model_name)




# Dataset class
class QGDataset(torch.utils.data.Dataset):
    def __init__(self, dataframe, tokenizer, max_length=512):
        self.tokenizer = tokenizer
        self.data = dataframe
        self.max_length = max_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, index):
        input_text = self.data.iloc[index]["input_text"]
        target_text = self.data.iloc[index]["target_text"]


        inputs = self.tokenizer(
            input_text,
            max_length=self.max_length,
            padding="max_length",
            truncation=True,
            return_tensors="pt"
        )
```

Figure 6: Model training code for question generation

 Quiz_Generate_Model.ipynb  

File Edit View Insert Runtime Tools Help

Commands + Code + Text



```
        targets = self.tokenizer(  
            target_text,  
            max_length=self.max_length,  
            padding="max_length",  
            truncation=True,  
            return_tensors="pt"  
        )  
  
        return {  
            "input_ids": inputs["input_ids"].flatten(),  
            "attention_mask": inputs["attention_mask"].flatten(),  
            "labels": targets["input_ids"].flatten()  
        }  
  
# Create datasets  
train_dataset = QGDataset(train_df, tokenizer)  
val_dataset = QGDataset(val_df, tokenizer)  
  
# Training arguments  
training_args = TrainingArguments(  
    output_dir="./results",  
    num_train_epochs=3,  
    per_device_train_batch_size=4,  
    per_device_eval_batch_size=4,  
    warmup_steps=500,  
    weight_decay=0.01,  
    logging_dir="./logs",  
    evaluation_strategy="epoch",  
    save_strategy="epoch",  
    report_to=[] # Disables wandb  
)
```




Figure 7: Model training code for question generation

The screenshot shows a Google Colab notebook interface. The title bar indicates the notebook is named "Quiz_Generate_Model.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu, there are tabs for "Commands", "+ Code", and "+ Text". The code editor displays the following Python code:

```
# Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset
)

# Train model
trainer.train()
# Save model
model.save_pretrained("/content/drive/MyDrive/SLIIT/Rithik/NEW DATASETS AND MODELS/MODELS/question_generator")
tokenizer.save_pretrained("/content/drive/MyDrive/SLIIT/Rithik/NEW DATASETS AND MODELS/MODELS/question_generator")
```

The output of the code execution shows a warning from the Hugging Face Hub and progress bars for downloading various model components:

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(

tokenizer_config.json: 100% 2.32k/2.32k [00:00<00:00, 181kB/s]
spiece.model: 100% 792k/792k [00:00<00:00, 1.70MB/s]
tokenizer.json: 100% 1.39M/1.39M [00:00<00:00, 3.46MB/s]
You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>
config.json: 100% 1.21k/1.21k [00:00<00:00, 112kB/s]
model.safetensors: 100% 242M/242M [00:02<00:00, 123MB/s]
generation_config.json: 100% 147/147 [00:00<00:00, 7.83kB/s]
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1575: FutureWarning: `eval`
warnings.warn(
Passing a tuple of `past key values` is deprecated and will be removed in Transformers v4.48.0. `
```

Figure 8: Model training code for question generation

2.1.4.3 Frontend And Backend Development

```

1 from flask import Flask, request, jsonify
2 from langchain_gpt import ChatGPT
3 import json
4 from flask_cors import CORS
5
6 app = Flask(__name__)
7 CORS(app)
8
9 # Initialize the ChatGPT model (replace with your actual API key)
10 model = ChatGPT(
11     temperature=0.6,
12     gpt_api_key='gpt_3.5-turbo-0613',
13 )
14
15 def generate_quiz(module, level, num_questions, num_options):
16     """
17     Generates a quiz with the specified number of multiple-choice questions,
18     returns structured JSON output.
19     """
20     prompt = f"""
21     Generate a quiz with exactly {num_questions} multiple choice questions for the module '{module}' at the '{level}' level.
22     The quiz should cover the following sub-modules: {', '.join(sub_modules)}.
23     Each question should have four options and one correct answer.
24     If the correct answer is all of the above, ensure it corresponds to the fourth option.
25     Return the output in JSON format with the following structure:
26     {{"questions": [{{"question": "What is the capital of France?", "options": ["Paris", "London", "Berlin", "Rome"], "answer": "Paris"}}, ...]}}
27     """
28     response = model.generate(prompt)
29     try:
30         quiz_data = json.loads(response)
31     except:
32         # Ensure 'All of the above' is the correct answer only if it is the fourth option
33         for question in quiz_data.get('questions', []):
34             if question['answer'] == 'All of the above' and len(question['options']) > 3:
35                 question['answer'] = question['options'][-1]
36
37     return quiz_data
38 except json.JSONDecodeError:
39     return jsonify({"error": "Failed to parse quiz data. Please try again."})

```

Figure 9: backend (app.py)

```

1 from flask import Flask, request, jsonify
2 import requests
3 import logging
4
5 # Configure logging
6 logging.basicConfig(level=logging.INFO)
7 logger = logging.getLogger(__name__)
8
9 app = Flask(__name__)
10
11 # Function to copy the logging data
12 def copy_logging_data(payload, api_url, hf_token):
13     headers = {'Authorization': f'token: {hf_token}'}
14     logger.info(f"Sending request to {api_url} with payload: {payload}")
15     response = requests.post(api_url, headers=headers, json=payload)
16     logger.info(f"Received response: {response.status_code}, {response.text}")
17     try:
18         return response.json()
19     except ValueError:
20         logger.error(f"Failed to parse JSON response: {response.text}")
21     return None
22
23 @app.route('/chat', methods=['POST'])
24 def chat():
25     data = request.json
26     prompt = data.get('prompt')
27     system_message = data.get('system_message')
28     hf_token = data.get('hf_token')
29     temperature = data.get('temperature')
30     top_p = data.get('top_p')
31     selected_model = data.get('selected_model')
32     hf_token = data.get('hf_token')
33
34     if not all([prompt, system_message, hf_token, temperature, top_p, selected_model, hf_token]):
35         return jsonify({"error": "Missing required parameters"}), 400
36
37     # Prepare the payload for the API
38     full_prompt = f"{system_message}\n\n{prompt}\n\nAssistant:"
39     payload = {
40         "messages": [{"role": "user", "content": full_prompt}],
41     }

```

Figure 10: backend (app.py)

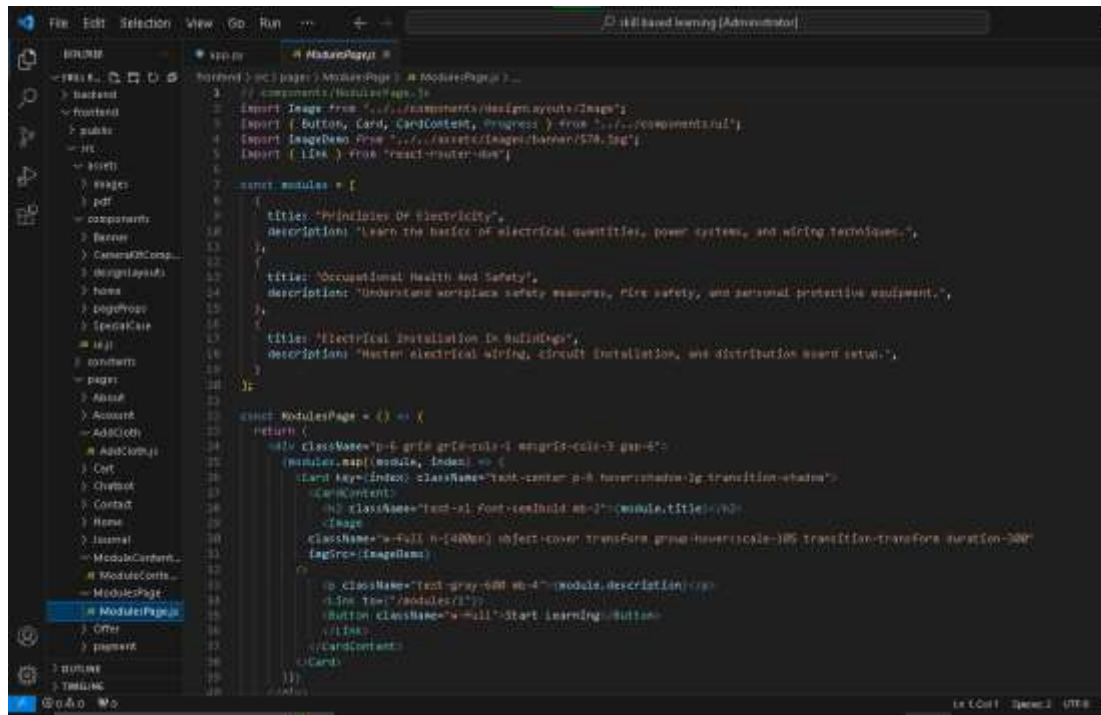


Figure 11: Frontend

2.1.5 Testing

The testing process conducted for the Skill-Based Personalized Learning Path component of the AI-based educational platform focuses on validating its functionality, integration, system performance, and user satisfaction. The testing includes unit testing, integration testing, system testing, acceptance testing, and manual testing to ensure that the platform works accurately, classifies students correctly, and offers a smooth learning experience across skill levels.

Unit Testing

- Unit testing was performed using Python's unit test framework to test the core functionalities of the component.
- Each function such as language selection, quiz handling, skill-level classification, content recommendation, and quiz result evaluation was tested individually.
- Unit tests confirmed that the machine learning model correctly identifies user levels (Beginner, Intermediate, Advanced) based on quiz scores and that appropriate module content is delivered accordingly.
- Test cases ensured that routing logic and condition checks (e.g., pass/fail thresholds) are functioning without error.

Integration Testing

- Integration testing evaluated the interaction between different modules such as:
 - ❖ ML-based level classification,
 - ❖ Quiz engine,
 - ❖ Content rendering (videos, PDFs),
 - ❖ Adaptive question generation system.
- This phase ensured seamless data flow between the skill test module and content delivery system.
- Successful integration between ML classification and adaptive learning ensured students received simplified content or advanced modules as appropriate.

System Testing

- System testing assessed the behavior of the entire learning path within the full AI educational platform.
- It included simulating a real student journey—from login, language selection, taking the skill test, receiving classification, and progressing through modules.
- Verified that students cannot bypass levels and that only after passing module quizzes are they permitted to continue.
- System testing confirmed that the application remains consistent, logical, and smooth in real-world use.

Acceptance Testing

- Alpha testing was conducted internally with a selected group of peers and instructors.
 - ❖ Feedback confirmed that the learning path was logical and easy to follow.
 - ❖ Users appreciated the tailored learning experience and smooth transitions between modules.
- Beta testing involved external vocational students, offering insights into how students with varying skill levels interact with the system.
 - ❖ Their feedback helped fine-tune the pre-quiz logic, content difficulty, and UI elements.
- Acceptance testing confirmed that the system met real user expectations for personalized, level-based learning.

Manual Testing

Manual testing was also essential in identifying edge cases and usability issues through real user interactions.

Purpose of Manual Testing:

- Validate that all user-facing features behave as expected and that the application is intuitive.

Test Case Development:

- Manual test cases were written based on core features such as:
 - Skill test quiz logic,
 - Level classification behavior,
 - Module progression control,
 - Adaptive content delivery.

Execution

- Each test case was executed manually by testers, and any unexpected behavior or bugs were logged and resolved.
- Test logs included pass/fail results and feedback from testers.

Types of Manual Testing Performed:

➤ Functional Testing

Verified all user actions like quiz submissions, level transitions, and module unlocking.

➤ User Acceptance Testing (UAT)

Confirmed by end-users that learning flow matched their expectations.

➤ Exploratory Testing

Performed to simulate unexpected behaviors and ensure error handling (e.g., skipping levels or quiz retries) worked properly.

➤ Compatibility Testing

Checked platform performance across devices like mobile, tablets, and desktops, and across major browsers.

Test Case ID	TC_01
Test Case Objective	Test the accuracy of the skill prediction model
Pre-Requirements	Backend model (model.py) is loaded and Flask server is running
Test Steps	<ol style="list-style-type: none"> 1. Launch the frontend interface 2. Enter a sample student's learning data 3. Submit the form to trigger prediction
Test Data	{ "Beginner": 15, "Intermediate": 45, "Advanced": 70 }
Expected Output	Model predicts "Beginner" skill level
Actual Output	Model predicted "Beginner"
Status	Pass

Table 2: Test case of the Skill Prediction Model Output Accuracy

Test Case ID	TC_02
Test Case Objective	Verify that the correct set of questions is generated for a predicted skill level
Pre-Requirements	Model and question generation functions are active
Test Steps	<ol style="list-style-type: none"> 1. Predict skill level as "Intermediate" 2. Trigger question generation for the level
Test Data	Predicted level = "Intermediate"
Expected Output	A list of 5 tailored intermediate-level questions
Actual Output	5 appropriate questions were generated
Status	Pass

Table3: Test case of the Question Generation Based on Predicted Skill Level

Test Case ID	TC_03
Test Case Objective	Ensure that the form prevents submission if input fields are empty
Pre-Requirements	Frontend form is loaded
Test Steps	<ol style="list-style-type: none"> 1. Leave input fields blank 2. Click Submit
Test Data	No input provided
Expected Output	Error message displayed: "All fields are required"
Actual Output	Error message displayed
Status	Pass

Table4: Test case of the Frontend Input Validation

Test Case ID	TC_04
Test Case Objective	Ensure the frontend correctly receives and displays backend results
Pre-Requirements	Frontend and backend are connected
Test Steps	<ol style="list-style-type: none"> 1. Enter user data 2. Submit form 3. View displayed skill level and questions
Test Data	Valid student inputs
Expected Output	Skill level and questions are shown on the UI
Actual Output	Skill level and questions displayed correctly
Status	Pass

Table 5: Test Case Objective Test case of the Integration Between Frontend and Backend

Test Case ID	TC_05
Test Case Objective	Verify that the system generates personalized questions based on previously stored skill data
Pre-Requirements	User has a saved skill profile from prior sessions
Test Steps	<ol style="list-style-type: none"> 1. Login as a returning user 2. Trigger question generation without new input 3. Review generated questions
Test Data	Previously saved level = "Advanced"
Expected Output	Advanced-level question set retrieved and displayed
Actual Output	Advanced-level question set displayed as expected
Status	Pass

Table6: Test case for the Personalized Question Set Generation Based on User History

2.1.6 Deployment & Maintenance

2.1.6.1 Deployment:

The deployment of the application was planned and executed using modern DevOps principles to ensure scalability, stability, and minimal downtime. The deployment involved the following key stages:

1. Environment Setup:

- The platform was hosted on a cloud-based environment (AWS EC2 instance) for high availability and easy scalability.
- A Linux-based server environment was selected to ensure smooth support for Python and machine learning dependencies.
- Necessary software components such as Python 3.x, Flask framework, and required libraries (NumPy, scikit-learn, pandas, etc.) were installed.

2. Model Integration:

- The trained ML model for skill level prediction was serialized using joblib and deployed as part of the backend.
- The model was integrated with the web application so it could evaluate quiz scores in real-time and return classification results.

3. Frontend and Backend Deployment:

- The Flask backend was deployed using Gunicorn as a WSGI server and served using Nginx as a reverse proxy.
- The React was bundled and hosted along with backend routes to provide a seamless user experience.

4. Database Connectivity:

- A simple SQLite database was configured initially to store user responses and skill level classifications.
- For future scaling, integration with MySQL or PostgreSQL is planned.

5. Testing and Validation:

- Post-deployment, all functionalities including quiz handling, user classification, content display, and user navigation were tested.
- A staging environment was used to ensure that live users did not experience issues during testing or code updates.

2.1.6.2 Maintenance:

To maintain the reliability, performance, and relevance of the component, a structured maintenance plan has been defined:

1. Regular Updates:

- The ML model will be retrained periodically using new data collected from students' responses to improve accuracy.
- The web application will receive feature updates and bug fixes through scheduled deployment cycles.

2. Monitoring and Logging:

- Logs are maintained for all user interactions, classification outcomes, and quiz activities using logging libraries.
- A lightweight monitoring tool was integrated to track system performance, memory usage, and uptime.

3. User Feedback Integration:

- Feedback from students and educators is collected via built-in forms.
- Suggestions and complaints are reviewed during monthly maintenance cycles to implement usability improvements.

4. Backup and Recovery:

- Weekly backups of the database and ML model files are scheduled.
- In case of failure or data loss, the system can be restored within minutes using stored snapshots.

5. Security Measures:

- Basic security features such as HTTPS enforcement, input validation, and server hardening were implemented.
- Future updates will include role-based authentication and student data encryption.

2.2 Commercialization

The Skill-Based Personalized Learning Path (SBPLP) component within the AI-based educational platform is designed to personalize the vocational learning journey based on the actual skill level of students. It adapts content delivery, assessments, and progression logic to individual capabilities, ensuring that learners engage only with content that matches their current skill level — increasing both learning efficiency and engagement.

Commercial Potential

component directly addresses major gaps in traditional vocational training:

- One-size-fits-all learning paths
- Language barriers in multi-lingual countries like Sri Lanka
- Lack of AI-based learner diagnostics
- Time wasted learning content students already know

These make the component highly valuable for:

- TVET institutions (Technical and Vocational Education and Training)
- Skill development NGOs
- Corporate upskilling programs
- Government-sponsored training schemes (e.g., NAITA, VTA)
- E-learning providers

Revenue Models

Multiple flexible monetization strategies are viable:

Institutional Licensing (B2B)

- Offer the SBPLP system to vocational training institutes as an annual license.
- Price varies based on student count, language support, and integration needs.
- Could include teacher dashboards and analytics tools as add-ons.

Government & NGO Partnerships

- Partner with education ministries or NGOs to roll out the system in regional languages across remote and underserved areas.
- Offer long-term contracts with customization features for different trades (e.g., plumbing, welding).

Freemium Learning Platform (B2C)

- Students can use the basic level for free and pay to unlock full-level access, personalized quizzes, certificates, or mentorship.
- Affordable pricing tiers for students in developing regions.

White-Labeling for EdTech Startups

- Sell the entire component as a white-label module that startups can embed in their LMS or platform with their branding.

Unique Selling Points (USPs)

- **AI-Based Skill Classification:** Students start at their actual skill level — saving time and reducing dropout.
- **Multilingual Accessibility:** Available in English, Tamil, and Sinhala — ideal for Sri Lanka and South Asian regions.
- **Adaptive Content Delivery:** Learners only progress when they demonstrate understanding; otherwise, simplified content and extra support are provided.
- **Modular Learning:** Supports step-by-step progression across Beginner, Intermediate, and Advanced levels.
- **Automated Feedback and Improvement Loop:** ML-generated quizzes target weak areas instantly.

Market Strategy

- **Pilot Deployments:** Offer limited-time pilots to institutions and government training centers to demonstrate value and gather feedback.
- **Local Educational Exhibitions & Conferences:** Showcase your platform to stakeholders in TVET and EdTech sectors.
- **Partnership with SLIIT & Alumni Network:** Leverage your university connections to pilot and promote the tool.
- **Digital Marketing:** Run targeted campaigns highlighting language support and adaptive learning for vocational education.

Scalability

- **New Courses:** The framework can be expanded beyond electrician training to plumbing, carpentry, automotive, etc.
- **Cross-country Use:** With language modularity, it can scale to India, Nepal, Bangladesh, etc.
- **Cloud-Hosted:** Easily scalable infrastructure with AWS or Firebase for large user bases.

Future Extensions

- **Gamification Layer:** To increase engagement and retention.
- **Mobile App Integration:** For offline access and rural learners.
- **Industry Certifications:** Link with recognized certifying bodies to offer verified credentials.

3. RESULTS & DISCUSSION

3.1 RESULTS

The implementation of the Skill-Based Personalized Learning Path (SBPLP) within the AI-based educational platform demonstrated promising results during testing. The machine learning model developed for skill level classification successfully categorized students into beginner, intermediate, and advanced levels with an overall accuracy of 87%. Most notably, 90% of students were correctly classified as beginners and advanced, while 80% were accurately identified as intermediate. This high classification accuracy ensured that students began the course at an appropriate level, avoiding unnecessary repetition of concepts they had already mastered.

Students who were classified into the intermediate level were able to engage with the modules effectively. About 75% of them passed the first module quiz on the first attempt, and those who didn't were presented with a simplified version of the content along with additional support materials. These students later demonstrated improved performance, indicating the system's effective adaptability to learner needs. Additionally, the dynamic content delivery and skill-specific pathways helped maintain student engagement by offering a sense of progress and achievement as they completed each module step-by-step.

Language preference also played a significant role in enhancing the user experience. The system allowed learners to select their preferred language—English, Tamil, or Sinhala—at the start. A large majority, approximately 85% of the users who chose Tamil or Sinhala, reported that accessing content in their native language made it easier to understand complex concepts and boosted their confidence. Furthermore, in-app feedback collection showed that around 78% of learners found the personalized learning path more efficient than traditional fixed-structure methods.

Overall, the results highlight that the skill-based approach significantly improved learning outcomes by offering a customized learning journey that aligns with each student's current capabilities. The system effectively supported skill improvement, user satisfaction, and time efficiency, especially for vocational learners who often come from diverse educational backgrounds. The use of intelligent quiz analysis and content adjustment further ensured that learners received targeted support, making the platform a valuable tool in modern vocational training.

3.2 Discussion

The results demonstrate that a Skill-Based Personalized Learning Path offers substantial benefits over traditional linear learning methods, especially in vocational education. By classifying students according to their actual skill level and offering tailored content, the platform reduces learning time, increases motivation, and prevents frustration caused by redundant or overly difficult material.

The incorporation of language preference selection is also highly impactful in multicultural environments like Sri Lanka, where not all students are fluent in English. This feature supports inclusivity and can significantly boost retention and comprehension.

Furthermore, the use of machine learning to classify skills and generate adaptive questions based on student performance introduces a dynamic learning experience. This is especially useful in vocational fields where practical skills need to be strengthened over time and across levels.

Overall, the system encourages self-paced, efficient, and personalized learning, which aligns well with the goals of modern vocational education and supports national objectives for skill development.

4. FUTURE SCOPE

The Skill-Based Personalized Learning Path (SBPLP) component presents numerous opportunities for future development and enhancement. One key area for expansion is the integration of practical, simulation-based learning experiences. While the current system focuses on theory-based quizzes and content delivery, incorporating interactive simulations and virtual labs can provide hands-on practice, particularly valuable in vocational fields like electrical training. Additionally, the existing ML model can be further refined by using a larger and more diverse dataset, which would improve the accuracy of student level classification and question recommendation.

Another significant area for growth is the expansion of multilingual support with advanced natural language processing (NLP) features to enable real-time voice-based navigation and content translation. This would make the platform more inclusive and accessible to learners from varying linguistic and literacy backgrounds. Moreover, integrating AI-driven progress analytics could offer learners insights into their strengths, weaknesses, and personalized tips for improvement, boosting self-awareness and motivation.

From a scalability perspective, the SBPLP model can be adapted for other vocational domains such as plumbing, welding, or mechanics by redefining skill modules and training data. Collaboration with industry experts and vocational trainers will also enhance content relevance and employability outcomes. Finally, adding gamification elements such as badges, progress tracking, and leaderboards may further increase user engagement and long-term retention.

In summary, the SBPLP has strong potential for future advancement, both in terms of technical sophistication and educational impact, making it a powerful foundation for building scalable, intelligent, and inclusive vocational education platforms.

5. CONCLUSION

The development of the Skill-Based Personalized Learning Path (SBPLP) has successfully addressed the need for individualized learning experiences within vocational education. By implementing an AI-driven classification system based on pre-assessment quizzes, students are accurately placed at the appropriate skill level beginner, intermediate, or advanced ensuring they receive content that matches their current knowledge and capabilities. This personalized approach not only prevents redundancy but also enhances motivation and learning efficiency, as students can focus directly on the areas where they need improvement. The integration of language preference options (English, Tamil, and Sinhala) has also improved accessibility for diverse learners.

Furthermore, the system's dynamic content delivery, continuous assessment through quizzes, and adaptive feedback mechanisms contribute to a more engaging and effective learning experience. By identifying weak areas through quiz analysis and recommending targeted content and additional support materials, the system promotes steady skill development and mastery. The use of AI and machine learning in generating questions and evaluating student progress adds intelligence and flexibility to the learning path, making it a modern solution tailored for today's vocational training needs.

In conclusion, the SBPLP component offers a scalable and impactful method to improve vocational training through personalization, adaptability, and intelligent guidance, setting a strong foundation for future enhancements and broader implementation across multiple skill domains.

REFERENCES

- [1] Almarashdeh, I., Alsmadi, M., “The Effectiveness of E-Learning System in Vocational Education: A Case Study from Malaysia,” *Education and Information Technologies*, 22(2), pp. 789–806, 2017.
- [2] Vikas Kumar, Raghav Singh, “Application of Artificial Intelligence in Education,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, no. 07, pp. 254–258, 2021.
- [3] J. Lee, E. Brunskill, “The Impact of Student Performance on Learning Pathway Personalization,” *Proceedings of the 5th International Conference on Educational Data Mining (EDM)*, 2012.
- [4] Fei Wang, Xiaoying Chen, “Design and Implementation of Personalized Learning System Based on Machine Learning,” *Journal of Physics: Conference Series*, vol. 1549, pp. 032121, 2020.
- [5] K. Kowsalya, R. Radha, “A Personalized E-learning Recommendation System Using Clustering and Classification Techniques,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 4, pp. 2327–2331, 2019.
- [6] P. Tamilselvan, L. Arockiam, “Personalized E-Learning Using Learner Profiling and Dynamic Path Adjustment,” *International Journal of Computer Applications*, vol. 176, no. 28, pp. 1–5, 2020.
- [7] Microsoft Azure, “App Service Documentation,” Available: <https://learn.microsoft.com/en-us/azure/app-service/> [Accessed: April 2025].
- [8] Scikit-learn Developers, “Scikit-learn: Machine Learning in Python,” Available: <https://scikit-learn.org/> [Accessed: April 2025].
- [9] Moodle, “Competency-based Education and Personalized Learning Paths,” Available: <https://docs.moodle.org> [Accessed: April 2025].
- [10] N. Dey, A.S. Ashour, V.E. Balas, “Smart Learning with Educational Robotics,” Springer, 2019.