International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

# Smart Course Recommendation System

**N Sudha Laxmaiah[1],Bheem Reddy Akshaya[2],Konjarla Anvitha[3],Samudrala Kavya[4]**

[1]Assistant Professor;Department Of Computer Science And Engineering Bhoj Reddy Engineering College For Women Hyderabad India.

[2,3,4]B.Tech Students; Department Of Computer Science And Engineering Bhoj Reddy Engineering College For Women Hyderabad India.

Mail Id;sudha.nama@slv-edu.in[1],bheemreddyakshaya@gmail.com[2],konjarlaanvitha7@gmail.com[3], samudralakavya@gmail.com[4]

***Abstract***

*Selecting appropriate academic courses is a significant challenge for many students due to limited guidance, excessive online resources, and unclear career pathways. To address this issue, this paper proposes a Smart Course Recommendation System that assists learners in identifying suitable courses based on their interests, academic history, and career aspirations. The system analyzes student profiles, educational background, and learning preferences to generate personalized course suggestions.The architecture of the system includes several key modules such as student management, course repository, recommendation engine, career guidance module, and feedback analysis unit. The recommendation engine acts as the central component, employing machine learning techniques including content-based filtering, collaborative filtering, and data-driven analytics to provide accurate and adaptive course suggestions. Furthermore, the system incorporates real-time feedback mechanisms that enable the model to improve its recommendation accuracy over time. The scalable and user-friendly design allows the platform to support a large number of students simultaneously while functioning as an intelligent academic advisor.*

***Keywords:***

*Smart Course Recommendation, Personalized Learning, Recommendation Systems, Machine Learning, Content-Based Filtering, Collaborative Filtering, Educational Data Mining, Student Profiling, Career Guidance Systems, Intelligent Learning Platforms.*

## INTRODUCTION

In the modern educational landscape, students often encounter difficulties when selecting appropriate learning paths and career directions. The rapid expansion of online learning resources and academic programs has created numerous opportunities, but it has also introduced confusion in identifying the most suitable courses. Many learners struggle to determine where to begin their academic journey and how to progress toward their desired professional goals.To address this challenge, a Smart Course Recommendation System is proposed to assist students in making informed educational decisions. The system provides personalized guidance by analyzing various factors such as a student's interests, academic performance, previous learning activities, and long-term career aspirations. Based on these parameters, the platform recommends relevant courses that align with individual learning objectives.By integrating intelligent data analysis and recommendation techniques, the platform functions as a digital academic advisor. It supports students in selecting appropriate courses, developing relevant skills, and moving confidently toward their targeted career opportunities.

### Existing System

In traditional educational environments, course selection is typically performed through manual decision-making processes. Students often rely on guidance from teachers, parents, or peers when choosing courses or career paths. Although such advice can be helpful, it does not always reflect the individual interests, abilities, or long-term objectives of the learner.Because of these limitations, there is a growing need for an intelligent and scalable system capable of providing personalized, data-driven course recommendations that support students throughout their learning journey.

### Proposed System

The proposed work presents a Smart Course Recommendation System designed to provide students with a structured and personalized learning pathway aligned with their academic interests and career objectives. Unlike traditional approaches that rely on random suggestions or limited counseling support, the proposed system applies intelligent data analysis to identify the most relevant courses for each learner.The system evaluates multiple parameters including a student's academic records, subject preferences, learning patterns, and long-term career goals. Based on this information, the recommendation engine generates suitable course suggestions that begin with fundamental concepts and progressively move toward advanced and specialized topics. This step-by-step learning progression ensures that students develop the required foundational knowledge before moving to higher levels of expertise.

**Functional Requirements**

The functional requirements define the core operations that the Smart Course Recommendation System must perform in order to support both administrators and students. The system is designed with two primary modules: the Admin Module and the Student Module, each responsible for specific functionalities within the platform.

**Admin Module**

enables administrators to manage and maintain the overall functionality of the system. Administrators are provided with secure login and authentication mechanisms to ensure that only authorized users can access administrative controls. Through this module, administrators can create and manage career profiles that define different career paths available for students. They can also view, update, or delete career roadmaps, which include the sequence of courses, required skills, and examinations associated with each career option. In addition, administrators are responsible for managing educational pathways by specifying required academic qualifications and skill sets for particular professions. The system also allows administrators to publish announcements and guidance tips that help students stay informed about career opportunities, important updates, and academic advice. Once administrative tasks are completed, administrators can securely log out of the system.

**Student Module**

focuses on providing personalized career guidance and course recommendations to learners. Students can begin by registering on the platform and creating an account that stores their academic information and personal interests. After successful registration, students can log in to the system and update their profiles with details such as educational background, subject preferences, and existing skills. Based on this information, students can request career predictions generated by the recommendation system. The platform analyzes the provided data and suggests potential career options along with a confidence score that indicates the reliability of each recommendation. Students are also able to explore career roadmaps that describe the required courses, skills, and examinations needed to achieve specific professional goals. Furthermore, the system provides access to announcements and career guidance tips published by administrators. Students can log out of the platform after completing their activities.

**Non-Functional Requirements**

Non-functional requirements describe the quality attributes that ensure the efficiency, reliability, and usability of the Smart Course Recommendation System. These requirements focus on the overall performance and operational standards that the system must maintain.Performance is a critical factor, as the system should generate course and career recommendations quickly in order to provide a smooth user experience. The platform must process student information efficiently and produce accurate suggestions without noticeable delays. Usability is another important requirement, meaning the interface should be intuitive and easy to navigate so that students and administrators can interact with the system without technical difficulties.Finally, availability ensures that the platform can be accessed through the web at any time, enabling students to obtain guidance whenever needed.

**Computational Resources**

**Software Resources**

Software resources represent the technological tools and platforms required to design, develop, and operate the Smart Course Recommendation System. The software requirements specification outlines the expected behavior and capabilities of the system rather than focusing on implementation details. This documentation provides a clear understanding of what the system must accomplish and serves as a foundation for system design, cost estimation, and project planning. It also assists the development team in organizing tasks, monitoring progress, and ensuring that the system meets all required functionalities.The proposed system is designed to operate on Windows 11 or later versions of the operating system. The backend database management is handled using MySQL, which stores student data, course details, and recommendation information. The development environment used for coding and testing is Visual Studio Code, which supports efficient programming and debugging. The user interface is developed using HTML, CSS, and JavaScript to create an interactive and responsive web platform. The main programming language used for implementing the system logic is Python, while the Django framework is used to build the web application and manage server-side operations.

**Hardware Resources**

Hardware resources refer to the physical computing infrastructure required to support the development and execution of the system. These requirements ensure that the platform operates efficiently without hardware limitations. The system can be developed and deployed on a computer equipped with an Intel i5 processor or an equivalent processing unit capable of handling moderate computational tasks. A minimum of 4 GB of RAM is required to support system operations and application execution. Additionally, a 500 GB hard disk is recommended to store system files, databases, and project resources. These hardware specifications are sufficient for both development and testing purposes of the Smart Course Recommendation System.

**Software Process Model**

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

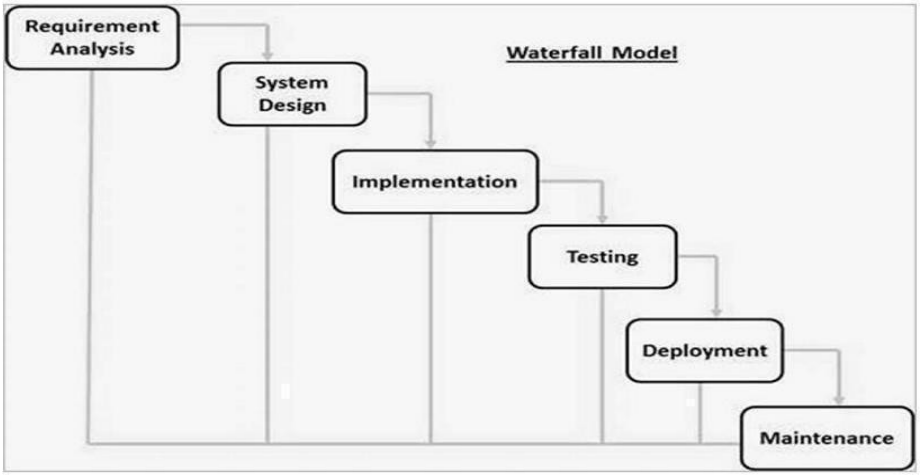



**Fig.1. Software Process Model**

A software process model provides a systematic framework for managing and organizing the software development lifecycle. It defines the sequence of phases involved in building a software system, including requirement analysis, system design, implementation, testing, deployment, and maintenance. By following a structured development model, software teams can ensure that the project progresses in an organized and controlled manner.Several process models are commonly used in software engineering. The Waterfall Model follows a linear sequence of phases, where each stage must be completed before moving to the next. The V-Model emphasizes validation and verification at each stage of development, ensuring that testing activities correspond to development phases. The Incremental Model focuses on developing the system in smaller functional components, allowing gradual integration of features. The Spiral Model combines iterative development with risk management to improve system quality. The Agile Model emphasizes flexibility, collaboration, and rapid feedback, enabling development teams to adapt to changing requirements.

**DESIGN**

System design focuses on translating project requirements into a structured solution that can be implemented efficiently. This phase involves defining the overall architecture, identifying system components, and planning how data flows between different modules. The design process begins with a clear definition of the problem and the objectives of the Smart Course Recommendation System. It includes planning for data collection, storage, and processing mechanisms required to support personalized recommendations.

Another important part of the design stage is **feature engineering**, where relevant attributes such as student interests, academic performance, and skill sets are identified and prepared for analysis. Appropriate algorithms and model structures are then selected to process this information and generate meaningful course or career recommendations. Additionally, strategies for model training and evaluation are defined so that the system's performance can be assessed using suitable metrics.

**Architecture**

System architecture describes the overall structure of the project and the interaction among different components involved in processing user requests. It explains how various modules cooperate to perform tasks such as collecting student information, analyzing data, generating recommendations, and presenting results to the user. An architectural description provides a conceptual representation of the system that enables developers to understand how different parts of the application are organized and how they communicate with each other.

The architecture of the Smart Course Recommendation System can be broadly categorized into **software architecture** and **technical architecture**. These two perspectives help define both the logical design of the application and the underlying technological infrastructure.

**Software Architecture**

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

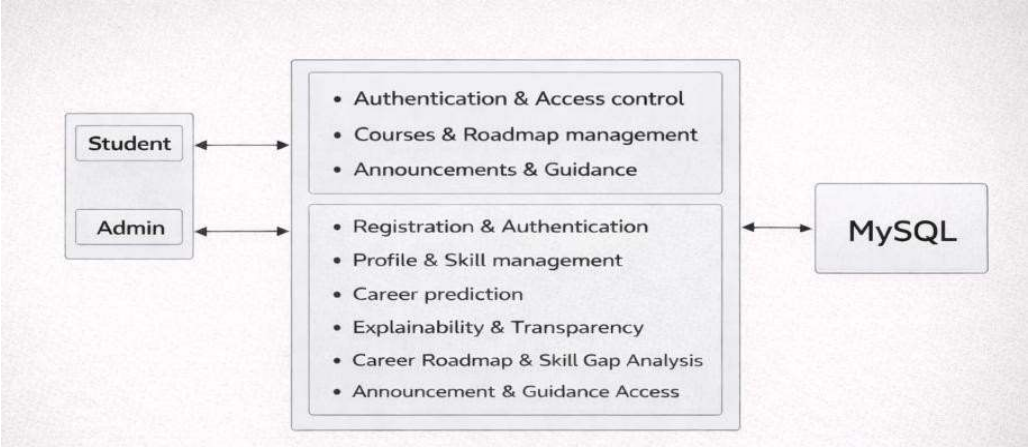



**Fig.2 Software Architecture**

Software architecture represents the logical organization of the system's components and their relationships. It defines how different modules such as user management, course data management, recommendation processing, and feedback analysis interact with one another. A well-designed software architecture improves system reliability, maintainability, and security.Architectural design tools and methodologies assist developers in identifying potential design flaws during the early stages of development. By analyzing the fundamental structure of the application, developers can detect possible vulnerabilities, evaluate risks, and ensure that security mechanisms are incorporated effectively. This approach helps reduce software defects and enables the development team to build a robust and scalable platform.In the Smart Course Recommendation System, the software architecture ensures smooth communication between the frontend interface, the backend processing layer, and the database that stores student and course information. This layered approach improves system stability and simplifies future upgrades or maintenance.

**Technical Architecture**

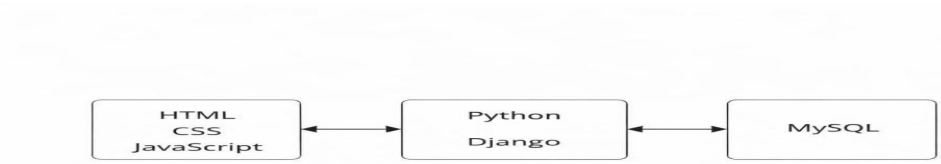



**Fig.3 Technical Architecture**

Technical architecture refers to the technological infrastructure that supports the functioning of the system. It provides a blueprint that defines how hardware resources, software frameworks, databases, and network components interact to deliver the required functionality.The technical architecture outlines the arrangement and interdependence of these elements to ensure that system requirements are satisfied. It includes server configurations, application frameworks, database management systems, and communication protocols used within the project. By defining these elements clearly, the system can achieve efficient data processing, secure information handling, and reliable performance.

**UML DIAGRAMS**

Unified Modeling Language (UML) is a widely used graphical modeling language that helps represent the design of software systems. It provides standardized diagrams that allow developers and stakeholders to visualize the structure and behavior of an application before implementation. UML diagrams serve a role similar to engineering blueprints, offering a clear representation of how different components interact within a system.UML integrates modeling concepts from several object-oriented methodologies, including the Booch method, Object Modeling Technique (OMT), and Object-Oriented Software Engineering (OOSE). By combining these approaches into a single standard notation, UML enables developers to design, analyze, and document complex systems in an organized and consistent manner.

**Use Case Diagram**

A Use Case Diagram is a behavioral diagram that illustrates how users interact with a system to

achieve specific goals. It identifies the external entities, known as actors, and the actions or services provided by the system. Each use case represents a particular function that the system performs in response to user input.In the Smart Course Recommendation System, use case diagrams illustrate how administrators manage career data and announcements, while students interact with the platform to obtain career predictions and course recommendations.

**Class Diagram**

A Class Diagram is a structural UML diagram that represents the static structure of a system. It describes the classes involved in the system, along with their attributes, methods, and relationships. This diagram plays an essential role in object-oriented design by defining how different components of the application are organized.In the Smart Course Recommendation System, the class diagram includes classes such as student profiles, career profiles, recommendation modules, and database entities. Relationships between these classes define how information flows and how system components interact to deliver personalized recommendations.

**Sequence Diagram**

A Sequence Diagram is an interaction diagram that illustrates how objects communicate with each other in a specific order over time. It focuses on the sequence of messages exchanged between actors and system components during a particular scenario.This diagram helps visualize the workflow of system operations, such as user authentication, profile creation, career prediction, and recommendation generation. By representing the chronological order of interactions, sequence diagrams provide a clear understanding of the dynamic behavior of the system.

**Database Design**

Database design defines how information is structured and stored within the system to ensure efficient data management and retrieval. The Smart Course Recommendation System maintains several collections or tables that store different categories of information required for generating recommendations and managing user activities.The collections store the primary datasets used by the system, including career information, student profiles, and skill requirements. The announcements collection stores updates and guidance messages published by administrators to inform students about new opportunities and important notifications.The system also maintains a skill gap dataset, which identifies missing skills for a particular career based on the student's existing skill set. Finally, the student profile collection stores information related to each user, including academic background, interests, and competencies. These datasets enable the system to perform accurate recommendation analysis and provide personalized guidance.

**IMPLEMENTATION**

Implementation involves converting the designed architecture into a functional software system. In this stage, the technologies, frameworks, and programming tools required for developing the Smart Course Recommendation System are selected and integrated. The implementation phase ensures that the system performs the intended tasks, including user authentication, data processing, recommendation generation, and interaction with the database.

**Technologies Used**

**Frontend Technologies**

The frontend of the application is responsible for presenting information to users and allowing them to interact with the system. **HTML** is used to create the structural layout of web pages, including registration forms, login pages, and recommendation dashboards. **CSS** is applied to style the interface and ensure that the application has a clean and user-friendly design. **JavaScript** enhances interactivity by handling user input, validating form data, and dynamically updating content without requiring page reloads.

**Backend Technologies**

The backend handles server-side operations and manages communication between the frontend interface and the database. **Python**, along with the **Django framework**, is used to implement the application logic and process user requests. Django also provides built-in security features and simplifies database interactions.For data storage, the system utilizes **MySQL**, which maintains structured information such as student profiles, career datasets, and recommendation results. The integration of these technologies ensures efficient data processing and reliable system performance.

**Algorithm Design and Pseudocode**

The recommendation mechanism in the Smart Course Recommendation System is based on analyzing the relationship between a student's existing skills and the requirements of various career roles. The algorithm begins by collecting user information, including skills, academic background, and interests. This data is stored in a structured student profile.In addition to skill matching, the system also evaluates the alignment between the student's interests and the career category. This step further improves recommendation accuracy. The final output includes a ranked list of potential careers, corresponding match scores, and suggested skills that the student should acquire in order to achieve the selected career path.

**TESTING**

Software testing is an essential phase in the software development lifecycle that aims to evaluate whether a system functions according to the specified requirements. The primary objective of testing is to detect errors or defects in the application and ensure that the developed system performs reliably and

efficiently before deployment. As digital technologies continue to influence everyday activities such as online banking, e-commerce, and digital communication, the reliability of software systems has become increasingly important. Even minor defects in software can lead to serious consequences, including financial losses, reduced system performance, or damage to an organization's reputation. Therefore, implementing systematic testing procedures is necessary to ensure the delivery of a high-quality product.In the development of the **Smart Course Recommendation System**, testing was performed to verify the correctness of system functionalities, validate system behavior, and confirm that the application meets the required quality standards. Effective testing helps improve system reliability, enhances user satisfaction, and strengthens the security of the application by identifying vulnerabilities during the development stage. Ultimately, the testing process contributes to building a stable and efficient recommendation platform that can be safely used by students and administrators.

**Dimensions of Testing**

Software testing can be analyzed from several perspectives depending on the scope, methodology, and objectives of the evaluation process. One important dimension involves testing different **layers of the application**, including the database layer, application programming interfaces (APIs), and the user interface. Each layer must be tested independently to ensure proper communication and functionality across the entire system.Another dimension is the **scale of testing**, which may involve testing individual components, modules, or the entire system as a unified application. Testing can also be categorized based on its purpose, such as functional testing to verify system features, performance testing to evaluate system responsiveness under load, and security testing to identify potential vulnerabilities. In addition, testing methodologies may vary between manual exploratory testing, scripted testing procedures, or automated testing tools that improve efficiency and coverage. Considering these dimensions helps ensure a comprehensive evaluation of the software system.

**Stages of Testing**

**Unit Testing**

Unit testing represents the initial stage of the testing process, where individual components or functions of the application are examined independently. Each unit is tested to ensure that it performs the intended operation correctly before integration with other modules. In this stage, developers verify the functionality of specific program elements such as functions, procedures, or small code segments. Techniques such as **white box testing** are commonly used during unit testing, allowing developers to evaluate the internal logic and control flow of the program.

**Integration Testing**

Integration testing is performed after individual units have been successfully validated. During this stage, different modules of the application are combined and tested together to verify their interaction and data exchange. The primary purpose of integration testing is to identify defects that may occur at the interfaces between modules. Even if individual components function correctly in isolation, integration issues can affect the overall performance of the application. Therefore, this testing stage ensures that all system components cooperate effectively.

**System Testing**

System testing involves evaluating the complete application as a unified system. At this level, the entire software solution is tested to determine whether it satisfies the functional and non-functional requirements specified during the design phase. System testing is typically conducted in an environment that closely resembles the real operating environment in which the application will be deployed. Independent testers often perform this testing stage to ensure objectivity. The goal is to verify that the system performs reliably and meets the expected quality standards before release.

**Acceptance Testing**

Acceptance testing represents the final phase of the testing process and focuses on validating the system from the user's perspective. This stage is commonly referred to as **User Acceptance Testing (UAT)**. The purpose of acceptance testing is to determine whether the developed system fulfills the practical needs and expectations of the end users. During this phase, users interact with the system to verify that its functionalities align with business requirements. Once the system successfully passes acceptance testing, it is considered ready for deployment in the production environment.

**Black Box Testing**

Black box testing is a testing technique in which the functionality of the system is evaluated without examining the internal structure or source code of the application. In this method, testers focus on the inputs provided to the system and analyze the resulting outputs to determine whether the software behaves as expected. Black box testing can be applied at multiple levels of testing, including unit testing, integration testing, system testing, and acceptance testing. This approach is particularly useful for validating system behavior from the user's perspective.

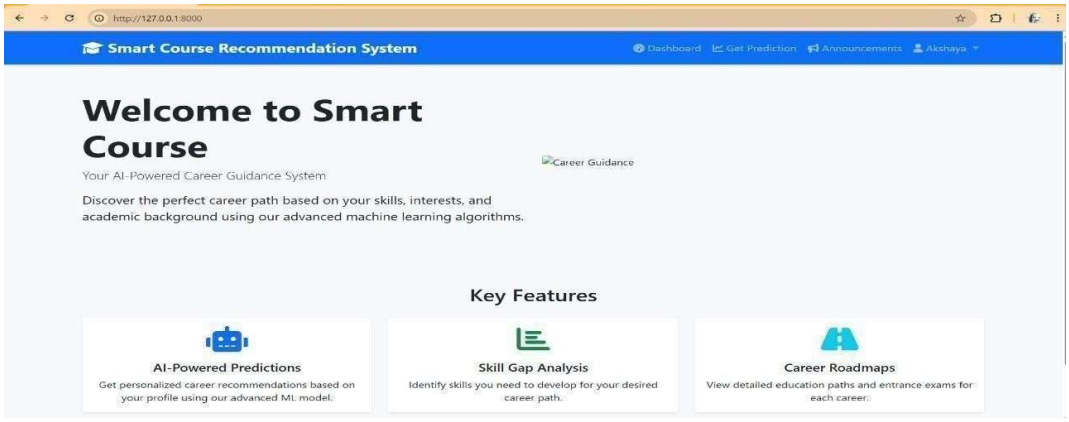
**White Box Testing**

White box testing, also known as structural or glass-box testing, involves examining the internal logic and implementation of the software. In this approach, testers require knowledge of the program's internal structure to design effective test

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

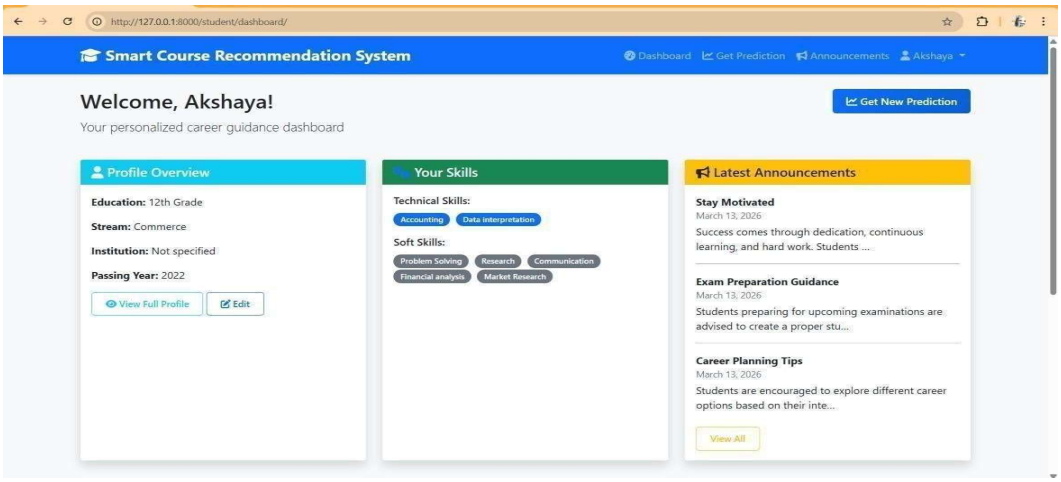
cases. White box testing is generally performed during unit testing and focuses on verifying the correctness of code execution paths. Common techniques used in this approach include **statement coverage**, **branch coverage**, and **path coverage**, which help ensure that different logical paths within the code are thoroughly tested.

**Screenshots**



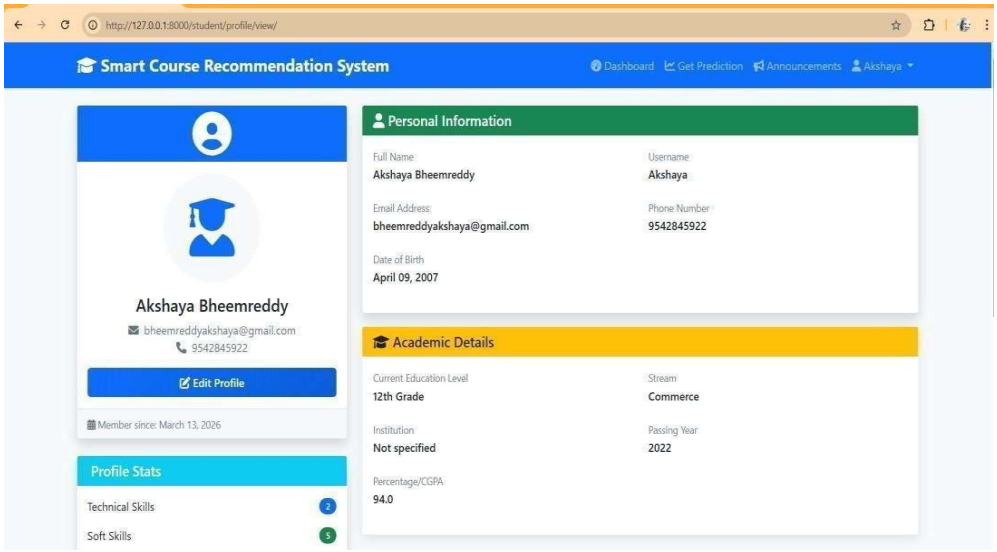

**Screenshot 1 Home page**


**Screenshot 2 Student details**

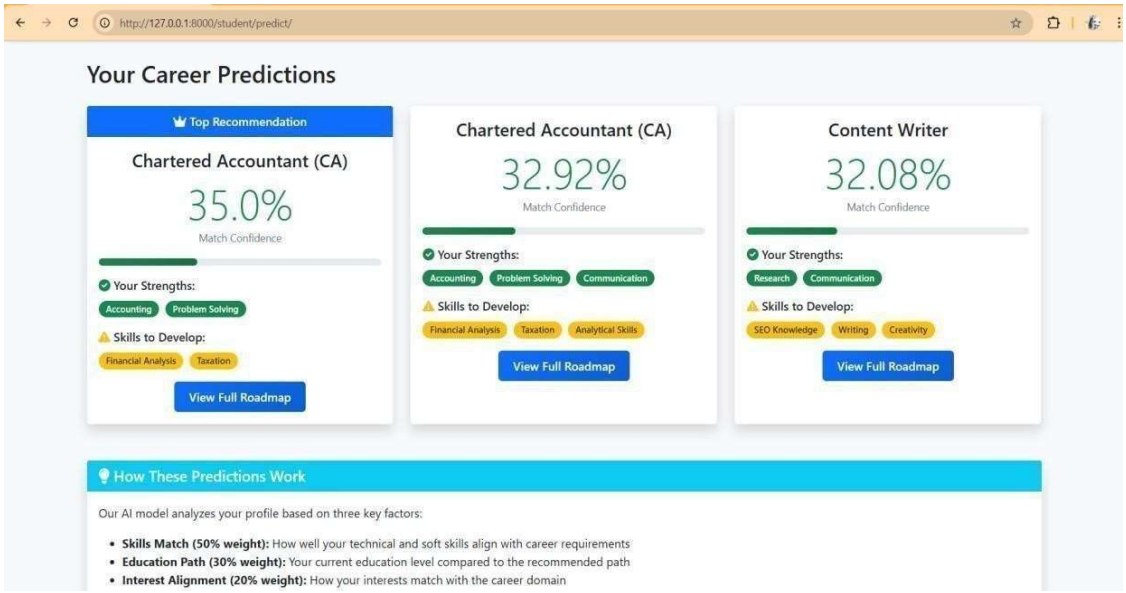



**Screenshot 3 Student Dashboard**

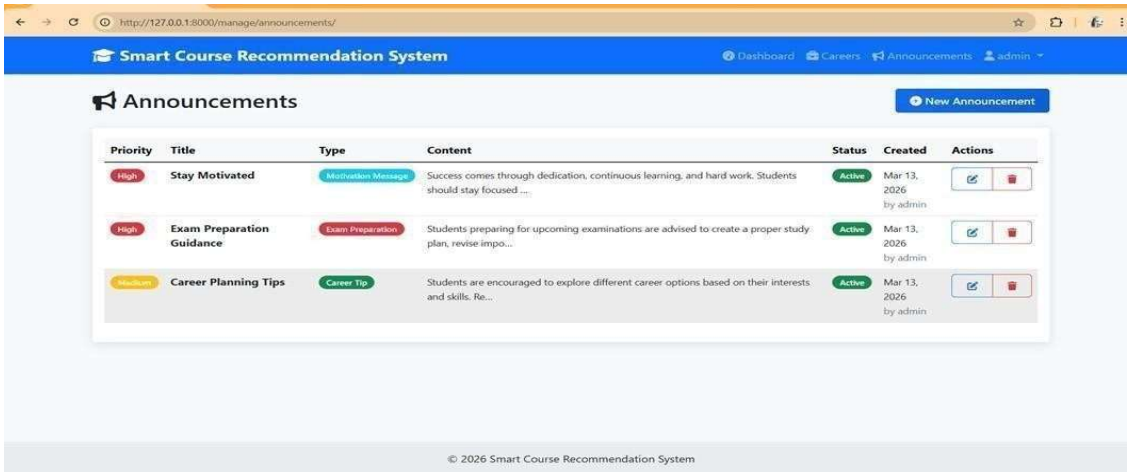
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026




**Screenshot 4 Student Profile**




**Screenshot 5 Student Career Prediction**




**Screenshot 6 Announcements**

**Conclusion**

The **Smart Course Recommendation System** presents an intelligent and structured approach to assisting students in selecting appropriate educational pathways and career directions. The system analyzes multiple factors such as academic performance, existing skills, and personal interests to generate personalized recommendations for suitable career options. By integrating these factors, the platform provides meaningful insights that help students better understand potential career opportunities.In addition to recommending careers, the system provides a **confidence score** for each recommendation, allowing users to evaluate how closely their current profile aligns with specific career paths. The platform also offers detailed career roadmaps that outline the necessary skills, courses, and educational steps required to achieve particular professional goals. Furthermore, the system identifies skill gaps between a student's current abilities and the competencies required for a chosen career, enabling learners to focus on developing the necessary skills.

**Future Scope**

Although the proposed system provides an effective framework for career guidance, several enhancements can further improve its capabilities. Future development may involve integrating advanced **Artificial Intelligence and machine learning techniques** to enhance the accuracy and adaptability of career recommendations. By continuously analyzing new student data and learning patterns, the system could refine its prediction models and deliver more precise suggestions.Another potential enhancement involves the integration of **AI-based conversational assistants or chatbots** that can provide instant responses to student queries and guide users through the recommendation process. Such interactive features would improve user engagement and make the system more accessible to students seeking quick guidance.Additionally, the implementation of **predictive analytics** could allow the platform to analyze market trends and emerging job opportunities. By incorporating real-time labor market data, the system could recommend career paths that are expected to have strong demand in the future. These improvements would enable the Smart Course Recommendation System to evolve into a more intelligent and comprehensive career guidance platform, ultimately helping students make well-informed decisions about their academic and professional development.

**REFERENCES**

[1] H. Zhou, J. Zhang, and Y. Wang, "A comprehensive survey of deep learning-based recommender systems," *Applied Sciences*, vol. 13, no. 20, pp. 1–28, 2023.

[2] Y. Li, X. Chen, and Z. Liu, "Recent developments in recommender systems: A survey," *IEEE Computational Intelligence Magazine*, vol. 19, no. 2, pp. 45–60, 2024.

[3] A. Sami, M. Khalid, and R. Ahmed, "Hybrid recommendation system using deep learning techniques," *Scientific Reports*, vol. 14, pp. 1–15, 2024.

[4] Y. H. Alfaifi, A. Alzahrani, and M. Alharthi, "Recommender systems applications and data sources: A review from 2018 to 2024," *Information*, vol. 15, no. 4, pp. 1–18, 2024.

[5] A. Valencia-Arias, J. A. Morales, and D. Garcia, "Artificial intelligence techniques in recommender systems: Trends and applications," *Machine Learning with Applications*, vol. 16, pp. 100–118, 2024.

[6] O. Remadnia, N. Bensaid, and M. Benabid, "Hybrid collaborative filtering recommendation model for personalized services," *Informatica*, vol. 36, no. 2, pp. 215–229, 2025.