**CHAPTER THREE**

**RESEARCH METHODOLOGY**

**3.1 Overview of existing system**

Present AI-based learning systems primarily focus on individual features such as personalized tutoring, content recommendations and automated study planning. However, their impact diminished because they rely on predefined datasets and static learner models. This approach limited real-time adaptability and short-term personalization features provided to its users although the AI-driven planners promoted engagement, structured learning and time management, the systems struggled to provide features required for truly individualized learning experiences such as real-time updates, strong personalization for each learner and comprehensive multi-feature support expected in the modern educational environment.

**3.1.1 Limitations of the Existing system**

1. Reliance on static data and limited adaptability to real-time updates.
2. Short term feedback with little focus on adaptive improvements.
3. Lack of full integration of tutoring, quiz generation, and study planning within a single system.
4. Weak support for diverse document types and multimodal inputs.
5. Minimal personalization beyond basic recommendations.

**3.2 Overview of the Proposed System**

The proposed system aims to enhance the student learning experience through the integration of multiple AI models and machine learning algorithms that deliver personalized educational support. It creates an intelligent learning environment that dynamically adapts to each student’s learning style and progress, improving engagement, comprehension, and overall academic performance.

The system adopts a multi-layered AI architecture that combines conversational interaction, predictive analytics, content recommendation, and adaptive feedback mechanisms to provide a more interactive and study-focused learning experience. It automatically generates study materials from uploaded documents, the system provides a personalized library for proper organization of study-related file types and includes a smart study planner to organize daily schedules and exam preparation.

The proposed **ACE Intelligent Study Companion** further improves on existing systems by integrating **content summarization, quiz generation, and personalized study planning** into a single adaptive platform. It supports multiple file types (PDF, images, PowerPoint, Excel, and text) and connects to external APIs for real-time learning updates. Through predictive and adaptive learning powered by machine learning, ACE ensures that recommendations, summaries, and quizzes continuously evolve with each student’s performance making it more flexible, scalable, and effective than previous AI-based learning systems.

**3.2.1 Benefits of the Proposed System**

The proposed system offers significant advantages over traditional educational platforms:

1. **Personalized Learning Experience**: Adaptive content delivery based on individual performance metrics, study habits, and learning preferences.
2. **Early Intervention Capabilities**: Predictive analytics identify struggling students before exam failure, enabling timely corrective actions.
3. **Comprehensive Academic Support**: Integrated platform for tutoring, assessment, content summarization, and study organization eliminates the need for multiple disconnected tools.
4. **Accessibility and Flexibility**: Cloud-based architecture accessible via web and mobile interfaces, supporting voice input for hands-free interaction.
5. **Data-Driven Insights**: Real-time performance tracking with metrics including attendance percentage, study hours, assignment completion rates, and participation levels.
6. **Automated Administrative Tasks**: Intelligent generation of study materials, quizzes, and schedules reduces manual preparation time for students.
7. **Enhanced Engagement**: Gamification elements (streaks, badges, levels) combined with conversational AI create an interactive learning environment that sustains motivation.

**3.3 Data Collection Methods**

To ensure the system’s reliability and improve the accuracy of its machine learning predictions, data were collected from both **primary** and **secondary** sources using various digital tools and APIs. Each source contributes to building a comprehensive dataset that supports intelligent recommendations, performance prediction, and adaptive learning.

#### **3.3.1 Primary Data Sources**

**a. Student Performance Datasets (Kaggle)**  
Two publicly available datasets from **Kaggle** were utilized to train and evaluate the performance prediction model:

1. **Student\_study\_habits.csv** – Contains data on study hours, attendance rate, sleep duration, and participation levels, which help in understanding behavioral factors affecting academic outcomes.
2. **Student\_performance\_data.csv** – Provides demographic details, tutoring records, number of absences, and grade classifications used to label and validate predictive outputs.  
   These datasets were downloaded, cleaned, and preprocessed using **Python (Pandas, NumPy, and Scikit-learn)** for integration into the machine learning pipeline.

**b. User Profile Data (Firebase Firestore)**  
The system continuously gathers **real-time user data** through **Firebase Firestore**, a NoSQL cloud database. This tool allows structured and secure data collection during user interactions. The information collected includes:

1. **Demographic details** – Name, age, gender, and date of birth.
2. **Academic information** – Course of study, subjects, degree level, and school type.
3. **Behavioral metrics** – Study hours per week, attendance percentage, and quiz completion rates.
4. **Activity logs** – Streaks, attendance records, and engagement patterns.

Firestore automatically timestamps and synchronizes all records, ensuring data consistency across devices.

**c. Interaction Data (Chat and Document Inputs)**  
User interaction data are captured through the **frontend chat interface** and stored in Firestore. This includes:

1. User queries and AI-generated responses during each chat session.
2. Session metadata such as timestamps, session IDs, and conversation titles.
3. Uploaded files (PDF, images, PowerPoint, Excel) and corresponding generated summaries.
4. Quiz results, flashcard progress, and recommendation histories.  
   This data supports personalization and continuous system learning.

#### **3.3.2 Secondary Data Sources**

**External APIs** are used to enrich the system’s dataset with relevant educational materials and contextual information:

1. **YouTube Data API v3**: Retrieves academic and tutorial video recommendations based on user subjects or study topics.
2. **Serper API (Google Search API)**: Fetches real-time supplementary learning resources and reference materials to support AI-generated content and feedback.

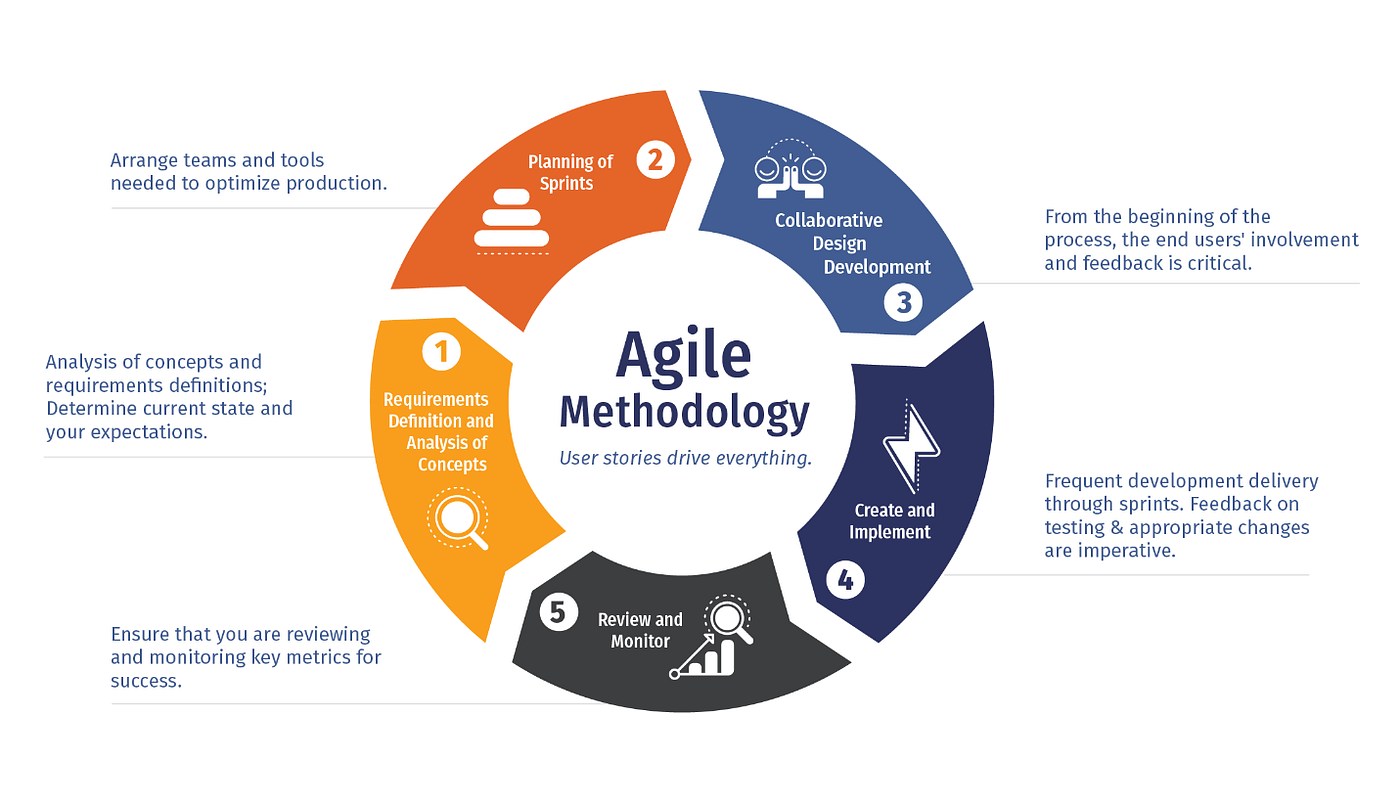
These APIs ensure the system provides **updated, diverse, and relevant learning content** while enhancing personalization and engagement.

**3.4 Software Development Methodology**

Agile methodology was adopted as the Software Development Life Cycle for this project, this approach prioritizes on flexibility, iterative and constant improvement method allowing the system develop to response to change of the user requirements overtime and provides continuous integration of AI features and components throughout the development process.

This methodology follows a repetition of cycles of planning, implementation, review and reflection, to ensure that there is continuous improvement in each cycle. The frameworks within the Agile approach include the Scrum and the Kanban, they both focus on teamwork, transparency and incremental delivery for results.

The Smart AI Study Companion System utilized the Agile methodology approach for its capacity to response effectively to changes made, and also maintain continuous progress done throughout the software development process.



**Phase 1: Requirements Definition and Analysis of Concepts**  
This is the initial phase where identifying the objectives, user needs and functional requirements. The Use case and feasibility analyses were used for defining the core components of the AI-powered predictions, chat summarization and document analysis

**Phase 2: Planning of Sprints**

The project will then be divided into simple manageable sprints. Each sprint has a focus on implementing individual functions including authentication, chat management, and AI model integration this approach ensures adaptive feedback and continuous progress.

**Phase 3: Collaborative Design and Development**

This phase involves the collaboration of frontend and backend teams to develop the system design which will include the components and interfaces. UML diagrams (use case, class, sequence) and architectural blueprints provide guidance for the implantation of the Flask backend, React frontend and the Firestore database.

**Phase 4: Create and Implement**

The implementation takes place in the phase it involves the core functionalities. The Flask backend was integrated with Firebase Authentication, Gemini AI, Cloudinary, while React handled real-time user interactions. The integration of the XGBoost classifier for predictive analysis takes place.

**Phase 5: Review and Monitor**  
Testing and reviews were done continuously to ensure the reliability of the system. Unit and integration testing were conducted for each individual sprint, while performance metrics such as model accuracy and response time were monitored for optimization. User feedback was collected to improve the systems performance.

**3.5 Requirements Analysis**

The requirement analysis defines the expectations and constraints that guide the development of the ACE Intelligent Study Companion. In 3.6.1, the functional requirements describe the essential capabilities expected from the perspective of the user such as authentication, quiz and summary generation, study planning, and performance tracking. Complementing these functional requirements, 3.6.2 the non-functional requirements also describes that are important to the systems usability, scalability and accessibility.

**3.5.1 Functional Requirements**

Functional requirements are requirements that describe the operations of the system and what the system is capable of doing. The functional requirements are as follows:

|  |  |
| --- | --- |
| **ACE-FR ID** | **FUNCTIONAL REQUIREMENTS** |
| **ACE-FR1** | The system shall allow users to register with their email and password or authenticate via Firebase Authentication. |
| **ACE-FR2** | The system shall allow users to log in using their credentials. |
| **ACE-FR3** | The system shall allow users to manage their profiles. |
| **ACE-FR4** | The system shall allow users sign out. |
| **ACE-FR5** | The system shall allow users to delete their accounts. |
| **ACE-FR6** | The system shall provide a multi-session AI chat with voice input and output, context awareness, and web search capabilities. |
| **ACE-FR7** | The system shall extract and summarize text from user documents (PDFs, Excel, PowerPoint) and images, and store the results in user collections. |
| **ACE-FR8** | The system shall generate and manage quizzes and flashcards from uploaded content. |
| **ACE-FR9** | The system shall create personalized study plans, track reminders, and send notifications based on due dates. |
| **ACE-FR10** | The system shall recommend relevant YouTube videos and store users’ recommendation history. |
| **ACE-FR11** | The system shall track users’ study progress through streaks, badges, levels, and attendance. |
| **ACE-FR12** | The system shall log users’ study activity, calculate attendance, and reset weekly study hours. |
| **ACE-FR13** | The system shall predict users’ academic performance based on collected data and suggest improvements. |

**3.5.2 Non Functional Requirements**

Non-functional requirements describe the requirements imposed on the system, they define qualities such as performance, scalability, security, reliability, and maintainability of the software. The non-functional requirements of this system are as follows:

|  |  |
| --- | --- |
| **ACE-NFR ID** | **NON-FUNCTIONAL REQUIREMENTS DESCRIPTION** |
| **ACE-NFR1** | The system should respond to normal API requests in under 5 seconds and AI answers in under 15 seconds, and support many users at the same time. |
| **ACE-NFR2** | The system should be scalable to accommodate a growing user base and increased data without compromising performance . |
| **ACE-NFR3** | The system should keep user data safe by using secure login, checking credentials and storing API keys securely |
| **ACE-NFR4** | The system should be reliable with the use multiple backup AI models |
| **ACE-NFR5** | The system should be easy to maintain by using clean code and logging errors. |

### 3.6 Machine Learning Methodology

### A data-driven machine learning approach was used to create a reliable model that predicts student’s performance based on their current academic performance. The systematic approach followed consists of problem definition, data preparation, model design, training, and evaluation to ensure models validity.

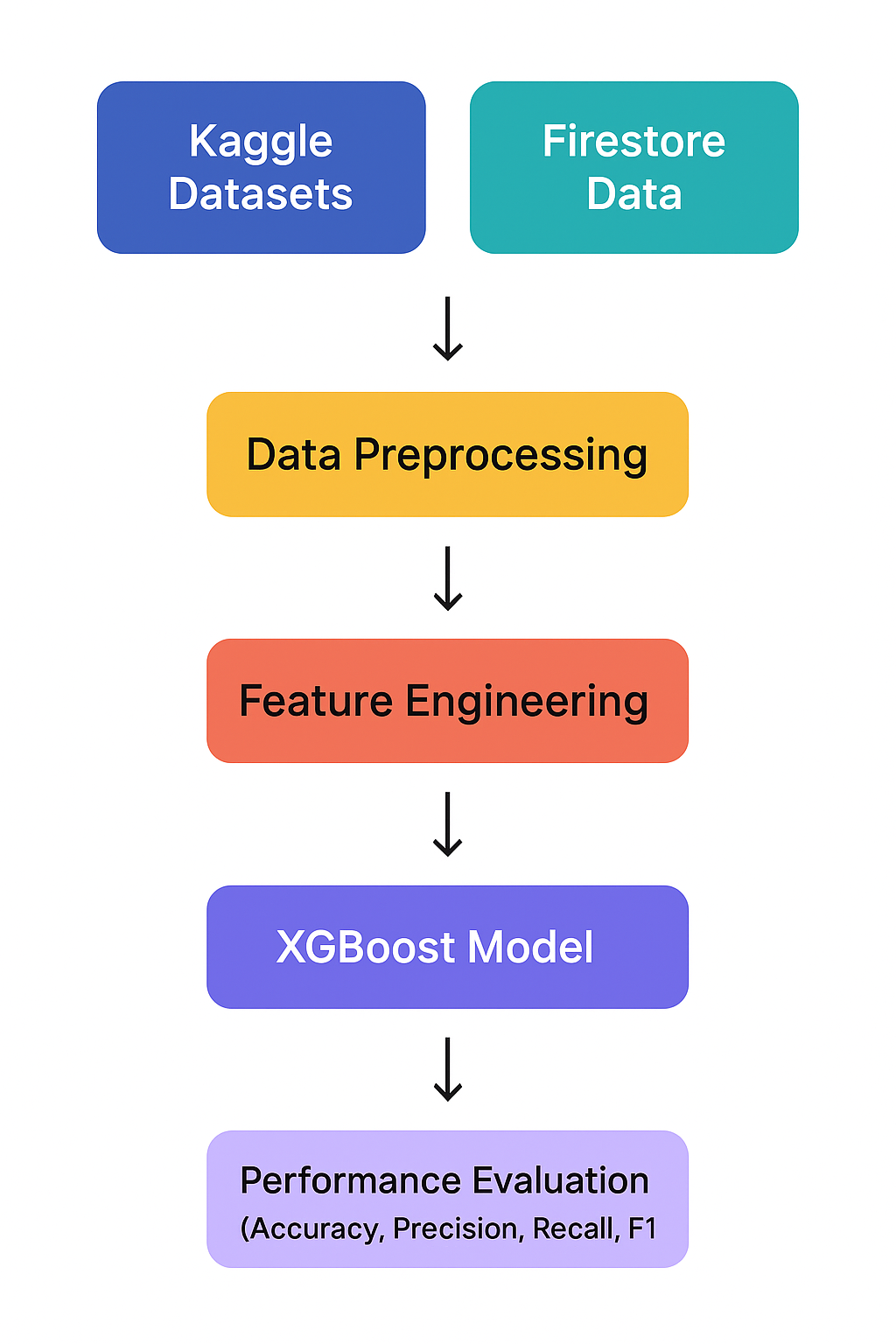
#### **3.6.1 Problem-Solving Approach**

The primary objective was to develop a predictive framework capable of identifying factors influencing student performance and providing actionable insights. The process began with defining key variables such as attendance, study habits, and behavioral metrics. The workflow involved collecting, cleaning, and transforming educational datasets into a suitable form for machine learning analysis.

#### **3.6.2 Design of the ML Framework**

The machine learning framework for the Smart AI Study Companion follows five major phases:

1. **Data Acquisition:**  
   The datasets were obtained from Kaggle and integrated with real-time user data stored in Firebase Firestore to create a comprehensive dataset.
2. **Data Preprocessing:**  
   Data was cleaned, normalized, and transformed to ensure consistency and handle missing or noisy values.
3. **Feature Engineering:**  
   Features such as attendance rate, study hours, and quiz scores were selected and encoded to improve interpretability and performance.
4. **Model Selection:**  
   XGBoost Classifier was chosen for its efficiency on tabular datasets, built-in regularization, and ability to handle non-linear feature interactions.
5. **Model Evaluation:**  
   Model performance was measured using standard metrics accuracy, precision, recall, and F1-score to ensure generalization and reliability.



#### **3.6.3 Machine Learning Algorithm and Tools**

XGBoost was preferred over alternatives (Logistic Regression, Random Forest, SVM, Neural Networks) for its computational efficiency, interpretability, and suitability for structured academic data. The model was implemented using **Python** due to its rich ecosystem of ML libraries particularly **Scikit-learn**, **XGBoost**, and **Pandas** which support rapid experimentation and reproducibility.

#### **3.6.4 Data Preparation and Cleaning Methods**

Data preprocessing techniques ensured consistency across different data sources. Techniques applied included:

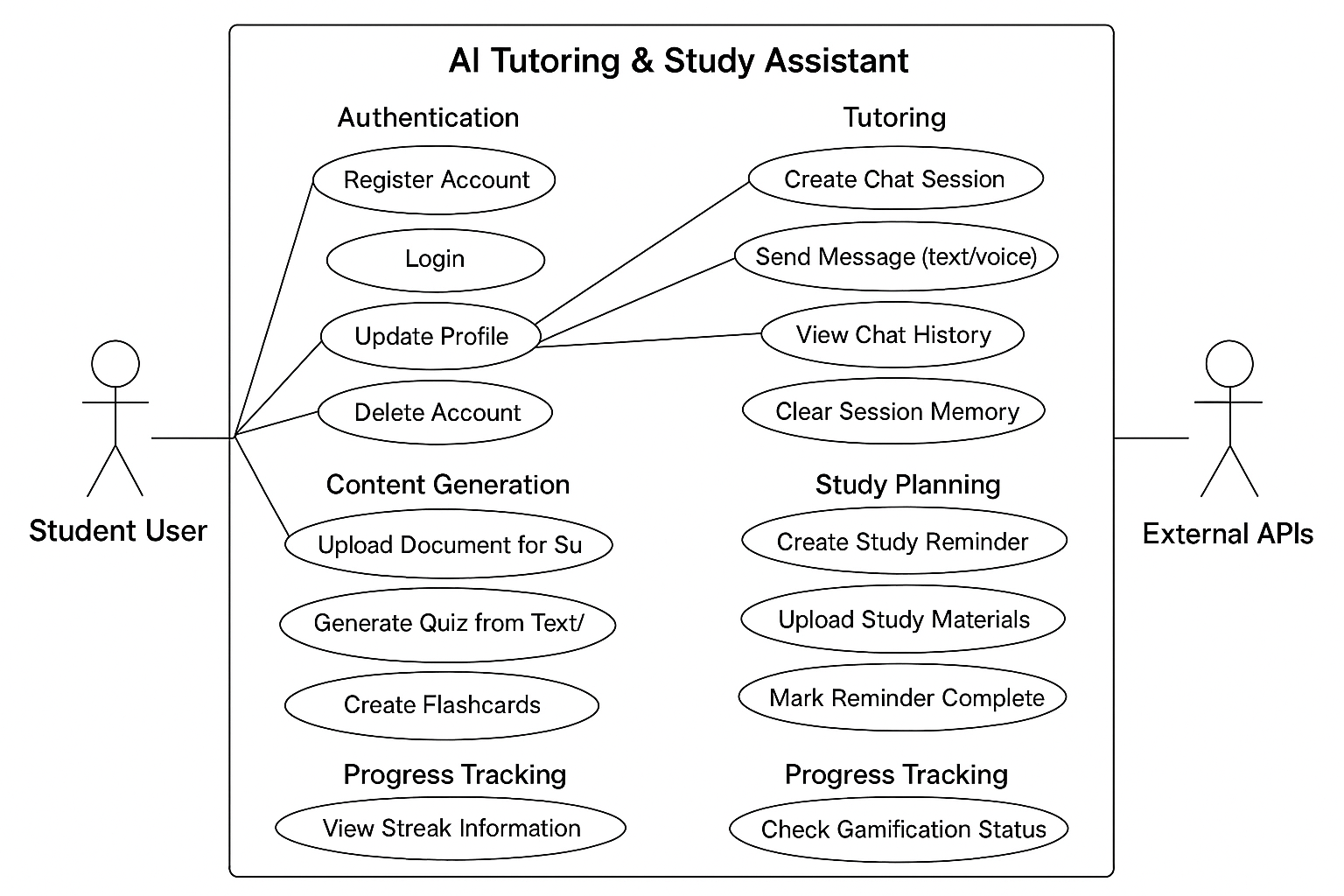
1. **Handling missing values** through imputation or exclusion where necessary.
2. **Normalization and scaling** using standardization to equalize feature contribution.
3. **Categorical encoding** for demographic and participation variables.
4. **Outlier detection** through visual and statistical methods to maintain realistic data distribution.

#### **3.6.5 Model Training and Validation**

The dataset was divided using a ratio of (80:20) split 80% for this ensures balances, where 80% is for training to ensure the model to generalizes and 20% is used for testing and ensuring adequate validation of results. To avoid the model from overfitting Cross-validation was applied to ensure the models stability, hyperparameter tuning ensured and guided the model performance results the parameters tuned include learning rate, max\_depth, n\_estimators this ensures accuracy and reliability of the model performance.

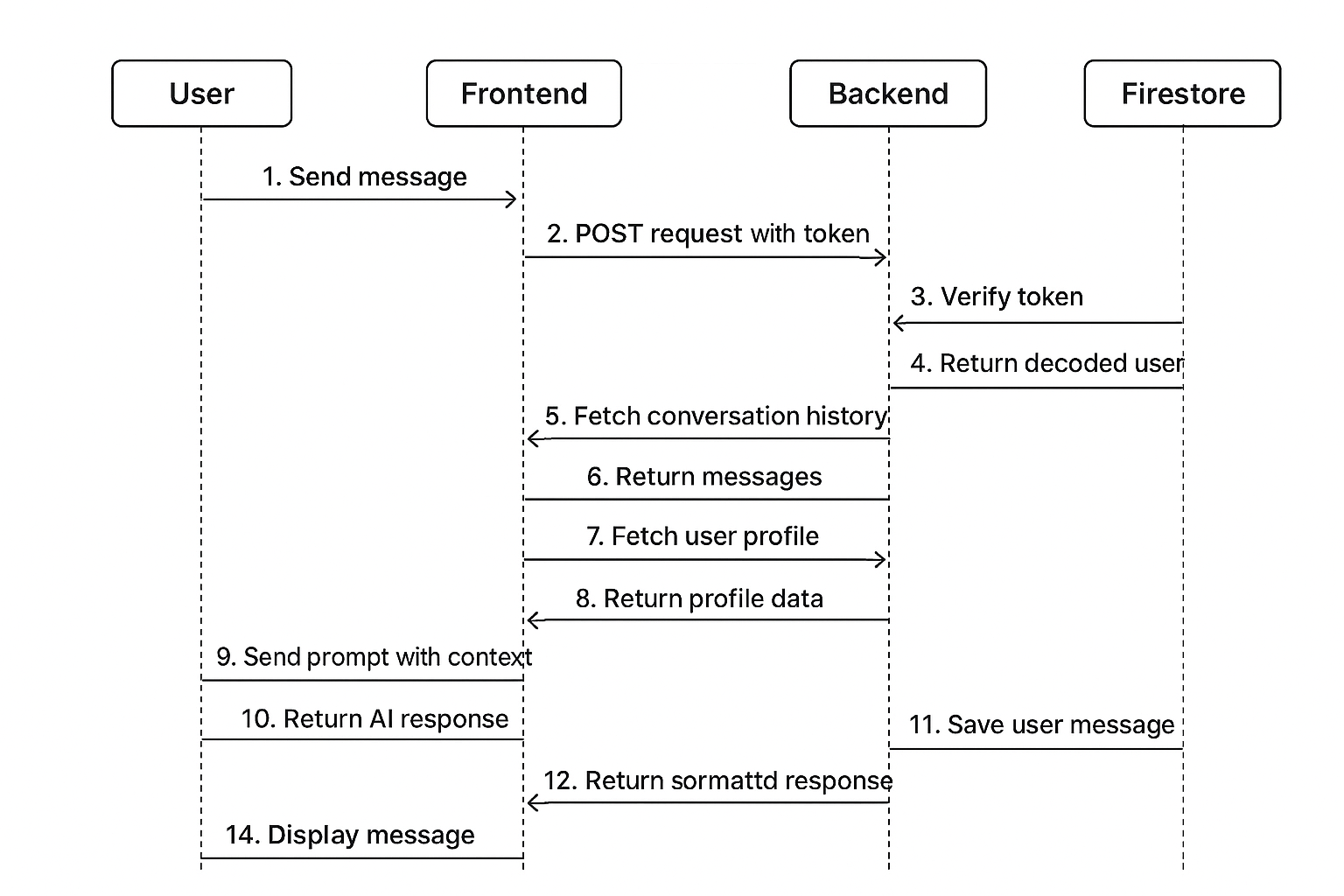
Evaluation metrics confirmed the models effectiveness in the prediction results of the student performances which was based on behavioral patterns

**3.7 Use Case Modeling Diagrams**

A Use Case Diagram is a graphical representation of the interactions between the users(actors) and the system. showcasing various used cases or functionalities of the system.

**3.8 Sequence Diagram**

A sequence diagram is a type of interaction diagram that illustrates operations between within different components another in a specific order.

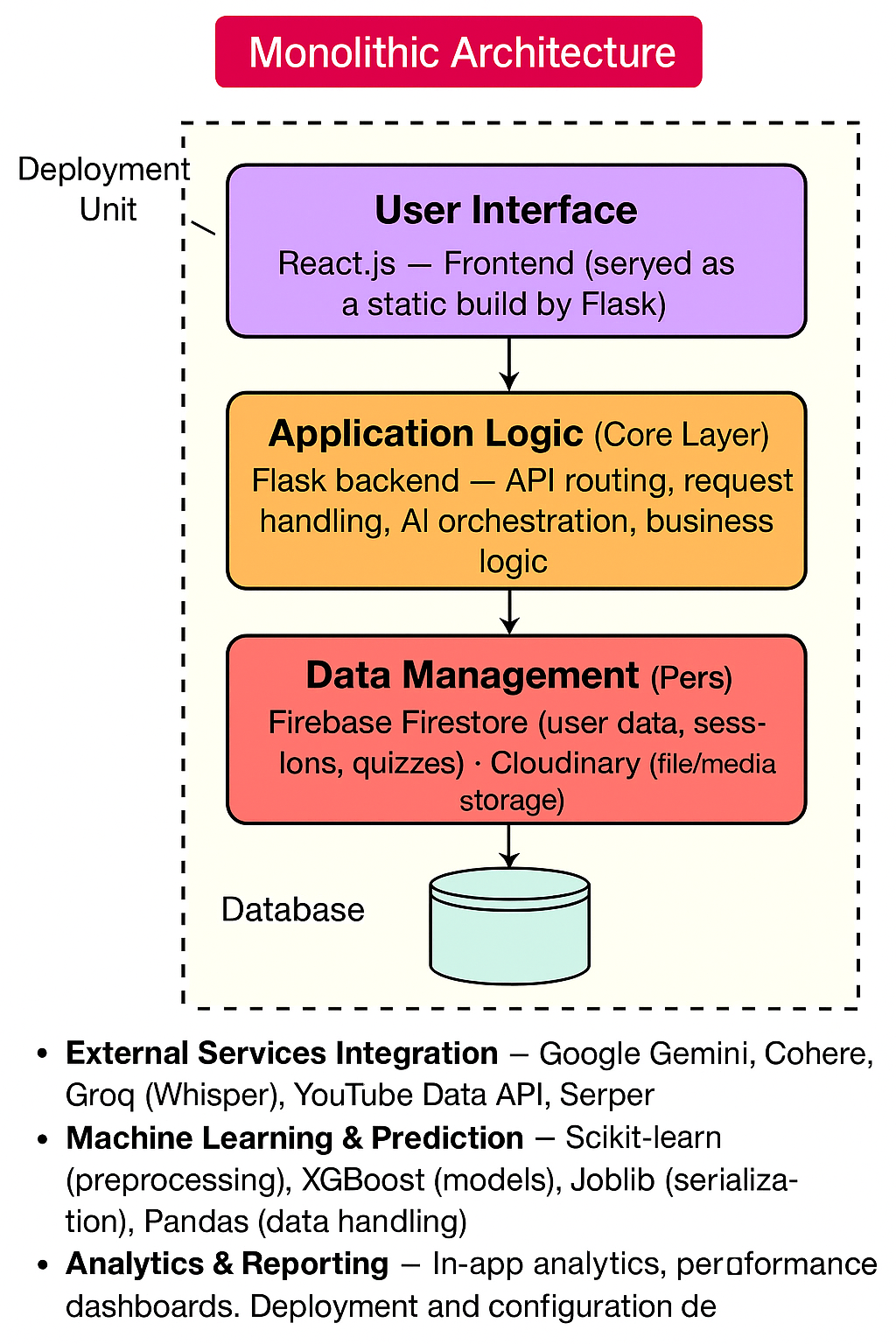


**3.9 System Architecture**

The system is designed and built on a monolithic Architecture, that combines all of the application's components into a single, indivisible unit. Under the architecture the user interface, application logic and data management are all created and maintained as one unified framework. This architecture enables seamless communication between all different components, allows simplified deployment and centralized control.

The Flask backend serves as the application’s core managing API routing, business logic, AI integration, and database interactions while the React.js frontend provides an interactive and responsive user interface.

All modules, including AI-driven chat, quiz generation, performance prediction, and analytics, operate within this single codebase. Data storage and retrieval are handled through FirebaseFirestore and Cloudinary, with additional functionality supported by external APIs such as Google Gemini, Cohere, Groq (Whisper), YouTube Data, and Serper.

The entire system is containerized using Docker for consistent deployment across environments, ensuring a scalable and maintainable monolithic solution.

**3.10 Main Menu Description**

The system interface is organized into key modules. The **Dashboard** gives an overview of study progress, recent chats, predictions, and reminders. The **AI Tutor** allows chatting, voice input, and integrated web search. **Study Materials** handles PDF uploads, summarization, and saved summaries. **Assessments** supports quiz and flashcard creation with score tracking. The **Study Planner** manages reminders and AI-generated study plans. **Recommendations** provides personalized video suggestions. **Performance Analytics** visualizes predictions and progress trends. The **Profile** module manages user details, subjects, gamification stats, and account settings.

**3.11 Input Design**

The input design for the system allowed was crafted to handle multiple user interactions of multiple types and different types of content format. Features like user entering plain text for chat session, quizzes and profile updates which must contain valid name, email and phone number before they can be accepted. Multiple file uploads include PDFs, Excel, Word, images for OCR, audio files for transcription, the study documents are stored in Cloudinary. The JSON inputs handle various features like user registration, activity logs, quiz scores, reminder updates. The API inputs uses strict validation to ensure correct email and valid fields are properly checked before processing is done.

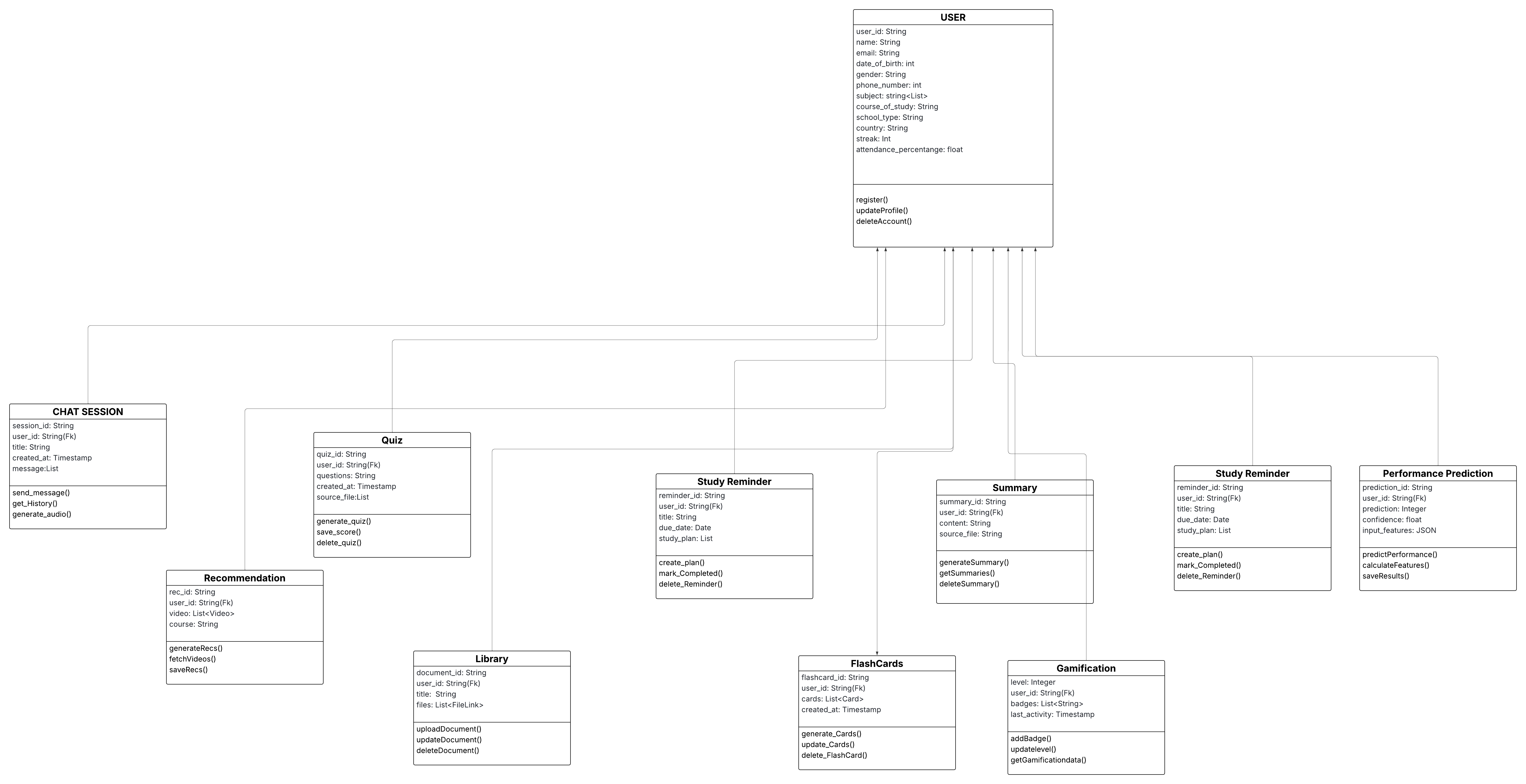
**3.12 Output Design**

The system generates structured and user-friendly outputs tailored for both frontend rendering and user interpretation. API responses are returned in standardized JSON format, with clear success or error messages and ISO 8601 timestamps. AI-generated outputs are refined for readability, using formatted text, bullet points, and structured study plans. Prediction results include binary outcomes (pass/fail), confidence scores, contextual feedback, and normalized input data. Content outputs such as summaries, quizzes, flashcards, and recommendations are provided in structured JSON formats. Frontend visualizations render streak calendars, performance graphs, and achievement badges dynamically. File outputs include secure Cloudinary URLs, downloadable summaries, and JSON-based quiz exports for offline use.

**3.13 Database Design**

The database design for the system describes how the data is structured to provide efficient storage management. Its No-SQL cloud database is used for its scalability and real-time retrieval of data which is critical to its operations. Data is stored within the collections and documents, each user has its own collection and likewise the related sub-collections such as chat-sessions, messages, reminders and quizzes and predictions.

The data structure provides flexible and dynamic storage with the support of a one-to-many relationship making it best suited for AI-driven features. The UML describes key features including User, chat-session, message, performance prediction, study-reminder, andQuiz*.*

****

### ****3.14 DEVELOPMENT TOOLS****

The development tools we selected for this project helped with design, implementation, testing, and deployment of the system. These tools improved developer productivity and ensured smooth interaction between the frontend, backend, and machine learning components.

1. **UI/UX**: This was used to create an intuitive and user-friendly interface, allowing seamless navigation and an engaging experience across all devices.
2. **IDE (Integrated Development Environment):** We used Visual Studio Code for writing, debugging, and managing both frontend and backend code efficiently. It has extensions that support Flask, React, and Firebase integration.
3. **React.js**: This JavaScript library helped create an interactive, responsive, and component-based frontend interface. It allows for efficient state management and real-time updates.
4. **Flask**: This lightweight Python web framework was used to develop the backend logic, manage RESTful API endpoints, perform input validation, and integrate AI models.
5. **Firebase Firestore**: This cloud-based NoSQL database offers real-time data synchronization for user profiles, chat sessions, quizzes, and reminders.
6. **Cloudinary**: This cloud service is for managing media assets like PDFs and images. It allows secure upload, transformation, and retrieval via HTTPS links.
7. **Docker:** This was used to containerize the backend application, ensuring consistent deployment environments, version control, and easier dependency management.
8. **Gemini Console**: This was used to integrate the Google Gemini API, which powers the chatbot’s main conversational intelligence and academic assistance functions.
9. **Groq Console:** This serves as a backup inference console, integrating Whisper for quick and efficient speech-to-text processing and a secondary AI fallback.
10. **Cohere Console:** This provides extra natural language processing capabilities as a backup large language model for text generation and semantic understanding.
11. **Jupyter Notebook**: This was used during the model development phase to train and evaluate the predictive analytics model with Python libraries like Scikit-learn and XGBoost.
12. **Cloud Platform:** A cloud environment, like Google Cloud Platform (GCP) or Render, can host the Flask backend, AI APIs, and Firebase integrations. This ensures scalability, continuous availability, and secure deployment.

**3.15 Main Menu Description**

The system interface is organized into key modules. The **Dashboard** gives an overview of study progress, recent chats, predictions, and reminders. The **AI Tutor** allows chatting, voice input, and integrated web search. **Study Materials** handles PDF uploads, summarization, and saved summaries. **Assessments** supports quiz and flashcard creation with score tracking. The **Study Planner** manages reminders and AI-generated study plans. **Recommendations** provides personalized video suggestions. **Performance Analytics** visualizes predictions and progress trends. The **Profile** module manages user details, subjects, gamification stats, and account settings.