Agent Based Push Mechanism Software Requirements Specification

Version 1.0

Submitted in Partial Fulfillment for the Award of Degree of Bachelor of Technology in Information Technology from Rajasthan Technical University, Kota

MENTOR:

Ms. Manoj Raman

(Dept. of Information Technology)

COORDINATOR:

Dr. Priyanka Yadav

(Dept. of Information Technology)

SUBMITTED BY:

Chinmay Bhatnagar (21ESKIT035)

Abhijay Sharma (21ESKIT002)

DEPARTMENT OF INFORMATION TECHNOLOGY

SWAMI KESHWANAND INSTITUTE OF TECHNOLOGY,MANAGEMENT & GRAMOTHAN

Ramnagaria (Jagatpura), Jaipur – 302017

SESSION 2024-25

Table of Contents

1. Introduction

Agent-Based Push Mechanism refers to a system that proactively delivers notifications by analyzing user behavior and preferences in real time. With the increasing reliance on personalized interactions, businesses and organizations require robust mechanisms to manage diverse notifications, such as alerts, recommendations, updates, and promotions. This approach ensures users receive contextually relevant messages, enhancing engagement and satisfaction.

The primary goal of an Agent-Based Push Mechanism is to dynamically adapt notifications based on user activities and predefined preferences. It streamlines the process of data collection, analysis, and delivery while ensuring scalability and reliability in handling high volumes of interactions. Additionally, such systems often integrate with existing platforms via APIs, enabling seamless communication and data exchange for a comprehensive user experience.

Overall, the Agent-Based Push Mechanism is a vital tool for modern applications, driving efficiency, relevance, and responsiveness in user communication. It addresses the evolving needs of businesses and users, ensuring continuous improvements in interaction quality.

Purpose

The purpose of this Software Requirements Specification (SRS) is to define the complete set of requirements for the Agent-Based Push Mechanism. This system is designed to deliver personalized and context-aware notifications by leveraging autonomous agents to analyze user preferences and behaviors. By enabling real-time adaptability, the system enhances the relevance and timeliness of notifications across diverse platforms and environments.

This SRS aims to fully describe the external behavior of the system, including its functional and non-functional requirements. Functional requirements outline the system’s core capabilities, such as user data analysis, dynamic notification generation, multi-channel delivery, and API integrations. Non-functional requirements specify performance benchmarks, security protocols, and reliability standards, ensuring the system meets expectations under different operational scenarios.

Additionally, the SRS highlights design constraints, such as compatibility with third-party systems, scalability to handle large user bases, and adherence to data privacy regulations. The document also addresses other critical factors, including system fault tolerance, maintainability, and user experience design to ensure seamless and effective communication.

Scope

The Agent-Based Push Mechanism (ABPM) described in this SRS is a robust software solution designed to deliver real-time, personalized notifications using autonomous agents. The system's primary focus is on enhancing the relevance, timeliness, and adaptability of push notifications across diverse platforms and user environments. It achieves this by analyzing user behavior, preferences, and contextual data, generating context-aware notifications that improve user engagement and satisfaction.

This system is intended for businesses, e-commerce platforms, IoT applications, and other use cases where timely and personalized communication is crucial. The key features of the system include agent-based data analysis, real-time notification delivery, multi-channel integration, and support for advanced communication protocols.

The software is associated with several use case models, such as:

- **User Data Analysis:** Agents analyze user behaviors and preferences to generate actionable insights.
- **Notification Generation and Delivery:** The system creates dynamic notifications tailored to individual users and delivers them through appropriate channels.
- **Integration with External Systems:** APIs enable seamless integration with existing platforms and third-party applications.
- **Performance Monitoring and Reporting:** Provides detailed analytics on notification effectiveness and system performance.

Definitions, Acronyms and Abbreviations

This section provides definitions of key terms, acronyms, and abbreviations used throughout this Software Requirements Specification (SRS) to ensure clarity and proper interpretation of the document.

Definitions:

- **Agent-Based Push Mechanism (ABPM):** A system that uses autonomous agents to analyze data and deliver context-aware notifications in real time.
- **Push Notification:** A message sent from the system to users proactively, often without explicit requests.
- **Agent:** A software component that performs specific tasks such as data analysis and notification generation autonomously.
- **Integration:** The process of connecting the system to external applications or platforms through APIs or other communication protocols.
- **Analytics and Reporting:** Features that track the delivery, engagement, and performance of notifications.
- **Real-Time Adaptability:** The ability of the system to adjust and deliver notifications dynamically based on changing user preferences or behaviors.

Acronyms and Abbreviations:

- SRS: Software Requirements Specification
- ABPM: Agent-Based Push Mechanism
- API: Application Programming Interface
- UI: User Interface
- DB: Database
- IoT: Internet of Things
- JSON: JavaScript Object Notation (a data exchange format)
- TLS: Transport Layer Security (a cryptographic protocol for secure communication)

References

1.4 References

[1] https://www.techtarget.com/searchcio/definition/learning-management-system

[2] https://www.scribd.com/document/465000318/Literature-Review-on-LMS

[3] https://www.ispringsolutions.com/blog/lms-requirements/

Technologies to be used

The development of the Agent-Based Push Mechanism (ABPM) system will utilize a combination of front-end, back-end, and database technologies, along with AI frameworks to ensure a high level of adaptability and personalization. Below is an overview of the technologies to be employed:

- Front-End Technologies - HTML (HyperText Markup Language)
- Used for structuring the content of web pages. - CSS (Cascading Style Sheets)
- Used for styling and layout of the user interface to ensure a responsive and visually appealing design - JavaScript
- Back-End Technologies - Node.js
- Database - MySQL

Overview

This Software Requirements Specification (SRS) document is structured to provide a clear and comprehensive description of the Agent-Based Push Mechanism system. It ensures that all stakeholders—developers, testers, administrators, and end-users—gain a thorough understanding of the project's goals and requirements. The document is organized as follows:

Section 2: System Overview

This section provides a high-level description of the **E-Learning Resource Management System**, its main functionalities, and its intended users. It outlines the core features and the system's primary objectives, giving a broad understanding of its purpose.

Section 3: Functional Requirements

This section details the specific functional capabilities of the system, such as content management, user authentication, role-based access control, and tracking of learner progress. Each function is described with clear inputs, processes, and outputs.

Section 4: Non-Functional Requirements

Defines non-functional aspects of the system, such as scalability, performance, security, and reliability requirements. These specifications set clear expectations for how the system should operate under various conditions to ensure consistent performance.

Section 5: System Models

Includes diagrams and models that visually depict the system architecture, workflows, and interactions. Use case diagrams, data flow diagrams, and class diagrams illustrate how system components work together to achieve the desired functionality.

Section 6: Appendices

Contains additional project-related information, such as a glossary of terms, references, and technical specifications. This section may also include implementation notes to assist the development team in understanding detailed aspects of the project.

2. Literature survey

The **Literature Survey** section examines existing research, technologies, systems, and methodologies relevant to the **Agent-Based Push Mechanism** (ABPM). It reviews related work, identifies knowledge gaps, and provides a comparative analysis of existing solutions to guide the development of the ABPM.

2.1 Review of Related Work

The field of agent-based systems has seen significant growth, with various applications in automation, notifications, and recommendation systems. This section explores some of the key works and technologies used in agent-based systems:

- **MELD (Multi-agent E-learning System)**: MELD is a system designed for managing and delivering personalized learning content based on agent-based architectures. While it offers personalized learning paths, the system faces challenges in scalability and real-time adaptation to learner behaviors.
- **Push Notification Systems**: Several existing push notification systems leverage basic agent-based approaches to deliver alerts. For example, Firebase Cloud Messaging (FCM) and OneSignal provide notifications across multiple platforms. However, these systems

lack advanced machine learning algorithms to personalize notifications based on user behavior or context.

- **Recommendation Systems**: Agent-based recommendation systems, such as those used by platforms like Netflix and Amazon, use agents to monitor user interactions and recommend content based on patterns. While these systems excel in personalization, their scalability in a multi-agent setup remains a challenge.
- **Real-time Event-driven Systems**: Event-driven architectures, such as those seen in Kafka or RabbitMQ, are often utilized in systems requiring real-time notifications. Though effective in distributing messages, they typically lack the intelligence needed for context-aware, adaptive push mechanisms.

Learning Resource Repositories: Research into digital libraries and content management systems highlights the growing need for advanced search, categorization, and metadata systems. Platforms like DSpace and OpenEdX store vast amounts of educational content and integrate various multimedia Review of Related Work

2.2 Knowledge gaps

Despite the advancement of agent-based systems, there remain several knowledge gaps in the field that the ABPM system aims to address:

- **Personalization and Context-Awareness**: Existing systems often rely on static rules or historical data to make decisions. There is a lack of intelligent, real-time personalization where the system can dynamically adapt to evolving user behavior and environmental changes.
- **Real-Time Notification Adaptation**: While real-time notifications are a common feature in push systems, the ability to dynamically adjust notifications based on user preferences and actions is still underdeveloped. Leveraging machine learning to personalize notifications at scale remains a major challenge. Current solutions face challenges in scaling to accommodate the growing volume of data generated in dynamic environments, particularly in sectors like IoT, where devices constantly produce real-time data streams

- **Scalability of Multi-Agent Systems**: Scalability is a critical factor for agent-based systems. As the number of users and agents grows, many systems struggle to maintain efficiency and responsiveness. Developing a scalable agent-based push mechanism that can handle a large number of concurrent users and notifications is an area that requires improvement.
- **Scalability of Cloud-Based Systems**: As e-learning grows, scalability becomes an issue. While cloud solutions like AWS and Azure provide scaling capabilities, not all e-learning systems are designed to handle the increasing volume of resources, users, and interactions, especially in large educational institutions or organizations with diverse needs.
- **User Experience**: Many e-learning platforms struggle to maintain a user-friendly interface while providing powerful resource management features. Balancing the complexity of advanced functionalities with ease of use remains a challenge.

2.3 Comparative Analysis

A comparative analysis of existing systems and technologies relevant to ABPM helps identify strengths and weaknesses in different approaches:

Key Observations:

- **MELD** offers personalization through agent-based decision-making but lacks scalability and real-time adaptation.
- **Push Notification Systems (FCM, OneSignal)** are excellent for basic notifications but do not support intelligent, context-aware notifications.
- **Recommendation Systems** excel in personalized content delivery but face challenges in multi-agent setups where personalization and interaction complexity increase.
- **Event-Driven Systems (Kafka, RabbitMQ)** provide robust real-time capabilities but lack agent-based intelligence, making them less suitable for personalized or adaptive notification delivery.
- **Resource Repositories** focus on content storage and management but fail to provide deep integration with performance tracking or personalized learning paths.

2.4 Summary

The Literature Survey highlights significant advancements in agent-based systems and push notification mechanisms, but also points to key gaps that need addressing. While existing systems like MELD, Firebase, and Netflix offer valuable insights into personalization and scalability, there is a clear opportunity to improve the adaptability and real-time decision-making

in agent-based push mechanisms. The ABPM system aims to bridge these gaps by providing a scalable, personalized, and intelligent system that can integrate with existing platforms and offer a seamless, context-aware user experience. By building on existing technologies and addressing the knowledge gaps, ABPM promises to offer a cutting-edge solution in the field of real-time, personalized notifications and content delivery.

3. Specific Requirements

This section defines the detailed requirements for the Agent-Based Push Mechanism (ABPM), which will serve as the basis for the system's design, development, and testing.

3.1 Functional Requirement

Functional requirements describe the core features of the ABPM system:

- User Registration and Authentication: Users (students, teachers, administrators) can register, log in, and manage their accounts.
- Role-Based Access Control: Access to resources and features is determined by user roles.
- Resource Management: Educators and administrators can upload, categorize, and manage educational content.
- Search and Filter: Users can search for resources using keywords or apply filters based on categories or topics.
- Progress Tracking and Reporting: The system tracks learner progress and provides detailed reports.
- Machine Learning Recommendations: Personalized resource recommendations based on user behavior and performance.

3.2 Non Functional Requirements

Non-functional requirements define the system's quality attributes:

- Performance: The ABPM must handle up to 5,000 concurrent users with minimal latency (under 2 seconds per notification).
- Scalability: The system must scale horizontally to handle increased numbers of agents, users, and notifications.
- Security: Implements data encryption, secure login (OAuth 2.0, SSO), and compliance with GDPR.
- Usability: Ensures a user-friendly interface that complies with WCAG 2.1 accessibility standards.

3.3 Hardware Requirements

- Server Requirements:
    - Processor: Intel Xeon or equivalent
    - RAM: 16 GB minimum
    - Storage: SSD with 1TB capacity
    - Network: High-speed Internet connection
- Client Requirements:
    - Device: Desktop, laptop, tablet, or smartphone
    - Browser: Latest versions of Chrome, Firefox, or Safari

3.4 Software Requirements

- Operating System: Linux (for server), Windows/Linux/Mac (for clients)
- Database: MySQL
- Back-End: Node.js, Python
- Front-End: HTML, CSS, JavaScript
- Machine Learning Libraries: Scikit-learn, Keras, PyTorch
- Version Control: Git

3.5 Agile Methodology

The project will follow Agile development practices, including:

- Sprint Cycles: Two-week sprints to deliver incremental improvements.
- Daily Stand-ups: Team sync-ups to track progress.
- Backlog Management: Prioritization of tasks in the product backlog.
- Continuous Integration/Continuous Deployment (CI/CD): Automated testing and deployment pipelines.

3.6 Business Process Model

The business process model for the system includes:

- Content Creation: Educators upload and organize resources.
- Resource Consumption: Students access and interact with content.
- Feedback Loop: Data from user interactions feed into analytics to refine recommendations.

3.7 Supplementary Requirements

- Localization: The system supports multiple languages.
- Audit Trails: Tracks user actions for accountability.
- Offline Access: Limited offline functionality for pre-downloaded resources.
- Notifications: Email and in-app notifications for updates and reminders.

4. System Architecture

The Agent-Based Push Mechanism (ABPM) system follows a modular, scalable architecture to ensure efficient resource management, secure data handling, and seamless user interactions. The system is designed with a client-server model and various communication interfaces to optimize user experience and data flow.

4.1 Client-Server Architecture

The architecture divides responsibilities between the client-side and the server-side, ensuring that each layer performs its respective functions efficiently.

Client-Side:

- Built with HTML, CSS, and JavaScript, the client-side ensures a responsive and user-friendly interface.
- Users interact with the system through a web browser or mobile app, performing actions such as resource browsing, searching, and progress tracking.

Server-Side:

- Developed using Node.js for handling asynchronous operations efficiently.
- Business logic, user authentication, and resource management are managed on the server.
- Python is used for advanced data analytics and machine learning functionalities.

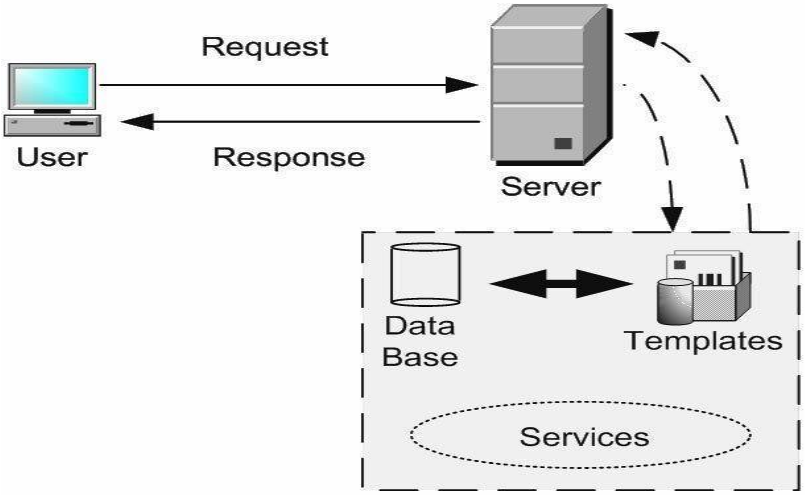
Data Flow:

- When a user performs an action (e.g., searching for a resource), the client sends a request to the server via HTTP/HTTPS.
- The server processes the request, retrieves the required data from the MySQL database, and sends the response back to the client.

Diagram (Conceptual Overview):

- [Client] → [Server] → [Database]
- [Client] ← [Server] ← [Database]

Client-Server Architecture



4.2 Communication Interfaces

The system employs several communication protocols and interfaces to ensure smooth data exchange:

- HTTP/HTTPS: Used for secure communication between the client and server, ensuring data integrity and security during transmission.
- RESTful APIs: The server exposes RESTful APIs for various operations, such as fetching resources, updating user progress, and managing content.
- WebSocket (Optional): For real-time features like live notifications and instant updates.
- Database Communication: The server communicates with the MySQL database using secure connections to fetch and store data.
- Third-Party Integrations: The system can integrate with external services (e.g., payment gateways or cloud storage) using APIs for additional functionalities.

- Authentication: Implements OAuth 2.0 or SSO for secure and efficient user authentication across systems.

5. Design and Implementation

The Agent-Based Push Mechanism (ABPM) system is designed with a modular, scalable architecture to handle resource management, secure data flow, and provide seamless user experiences. This section outlines the key product features of the system.

5.1 Product Features

The **ABPM system** offers a wide range of features to support its core functionalities.

- **User Account Management**: Provides role-based access for students, teachers, and administrators to manage their profiles, login, and permissions.
- **Resource Management**: Teachers and administrators can upload, organize, and categorize learning materials (e.g., videos, documents, quizzes).
- **Search and Filters**: Users can search for educational resources and apply filters such as subject, difficulty, or media type.
- **Analytics and Reporting**: Generates detailed reports on user performance, resource usage, and learning progress.
- **Personalized Recommendations**: Machine learning algorithms recommend resources based on user activity and preferences.
- **Real-Time Updates**: Provides live notifications and updates regarding new content or system changes.
- **Scalability**: The system is designed to scale with increasing users, resources, and data volume, supporting up to 10,000 concurrent users.

5.2 Data Flow diagram

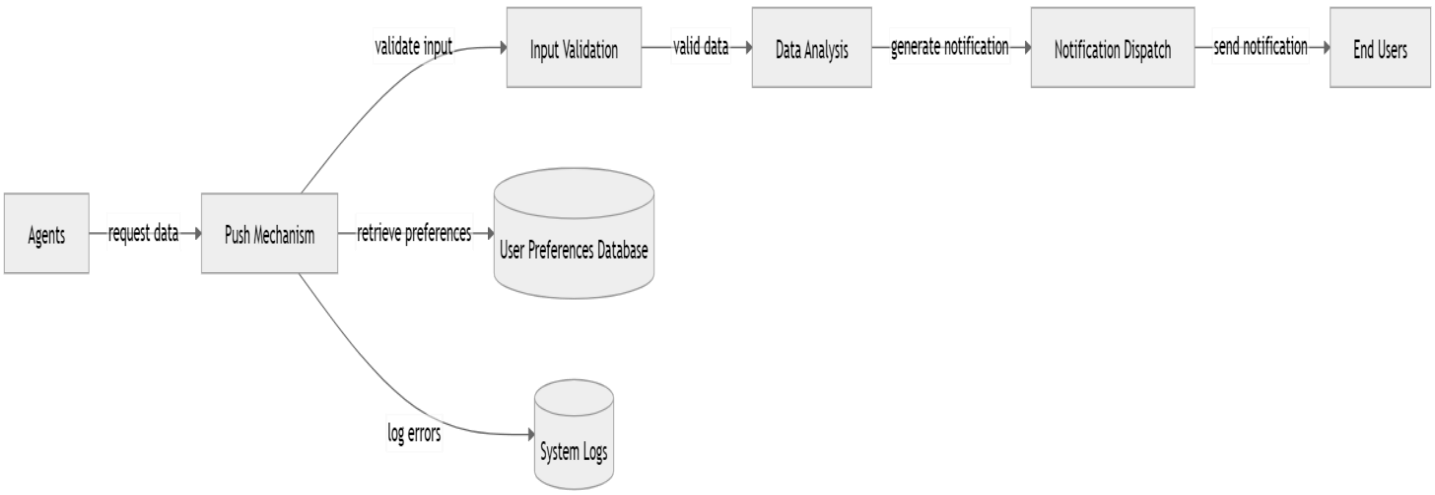


Figure 5.2.1:- Zero Level DFD

Figure 5.2.2 – First Level DFD

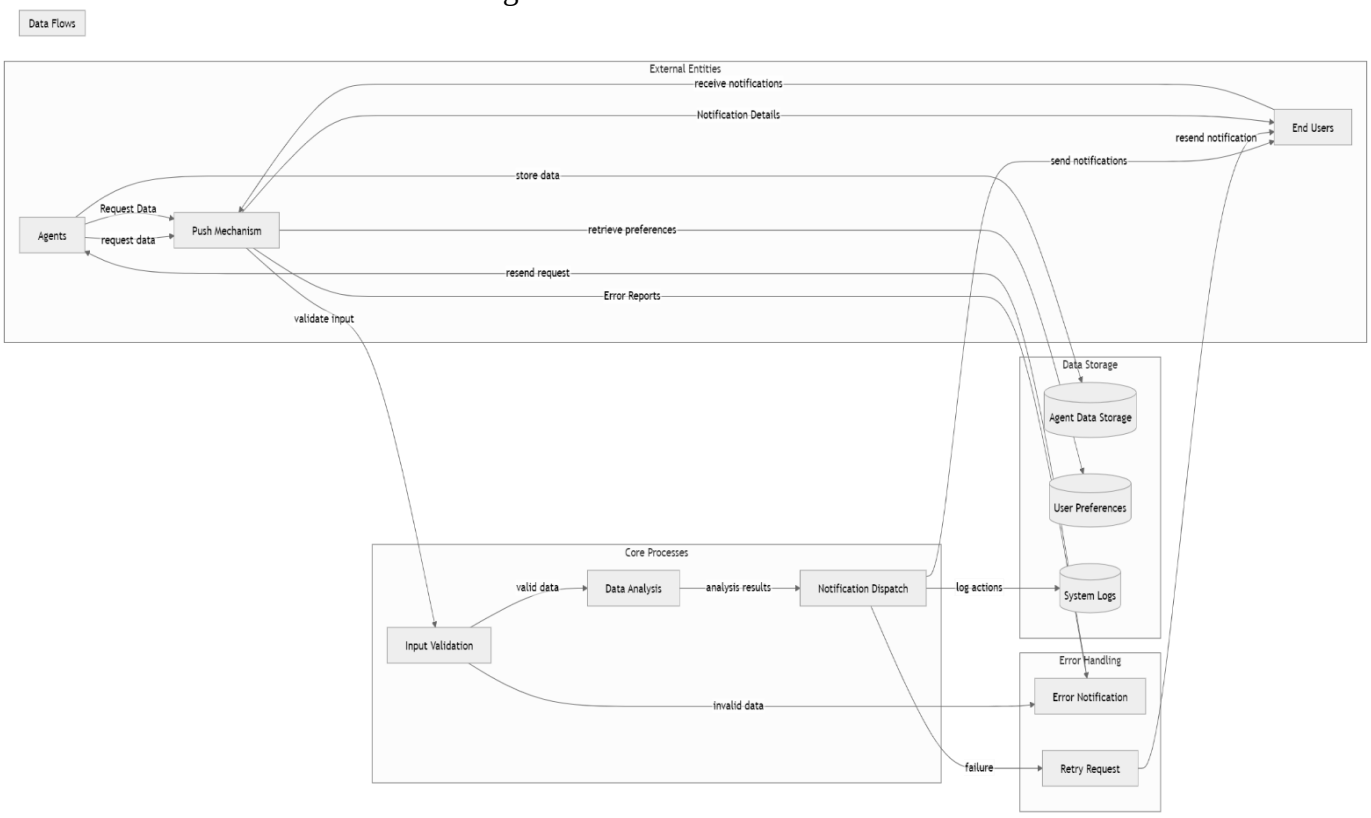
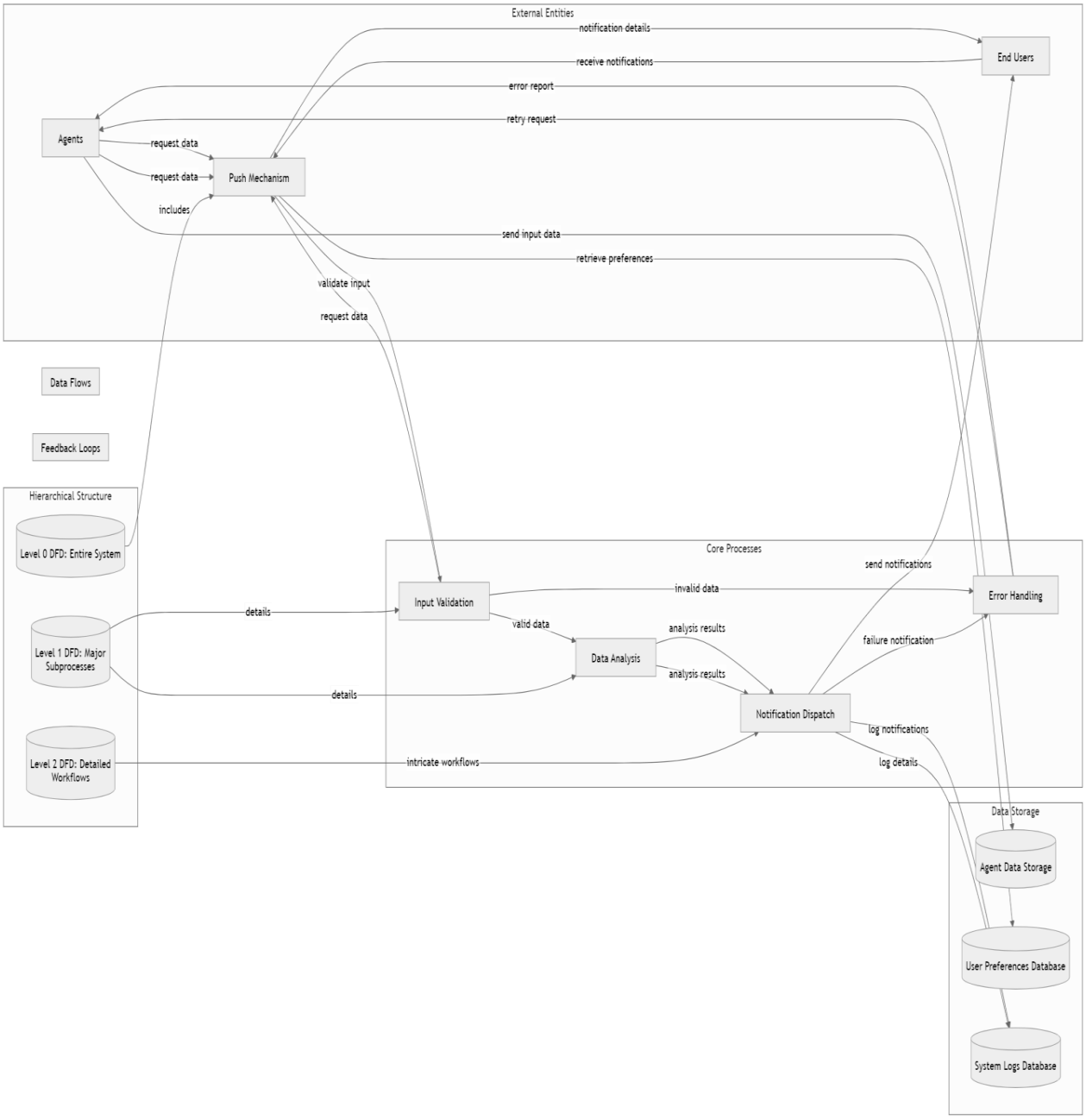


Figure 5.2.3:- Second Level DFD



5.3 ER Diagram

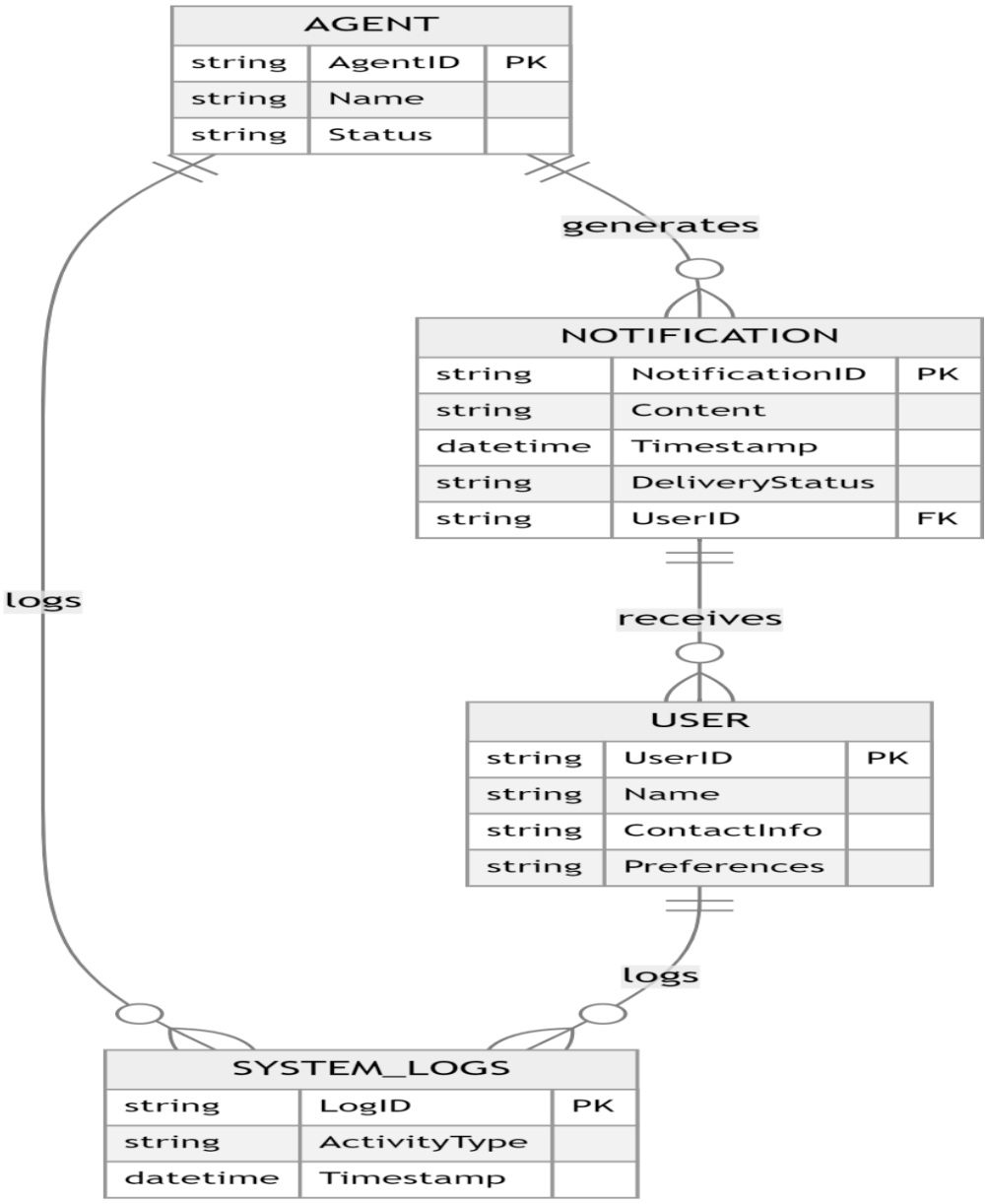


Figure 5.3.1:-ER Diagram for Agent based Push Mechanism

5.4 Class Diagram

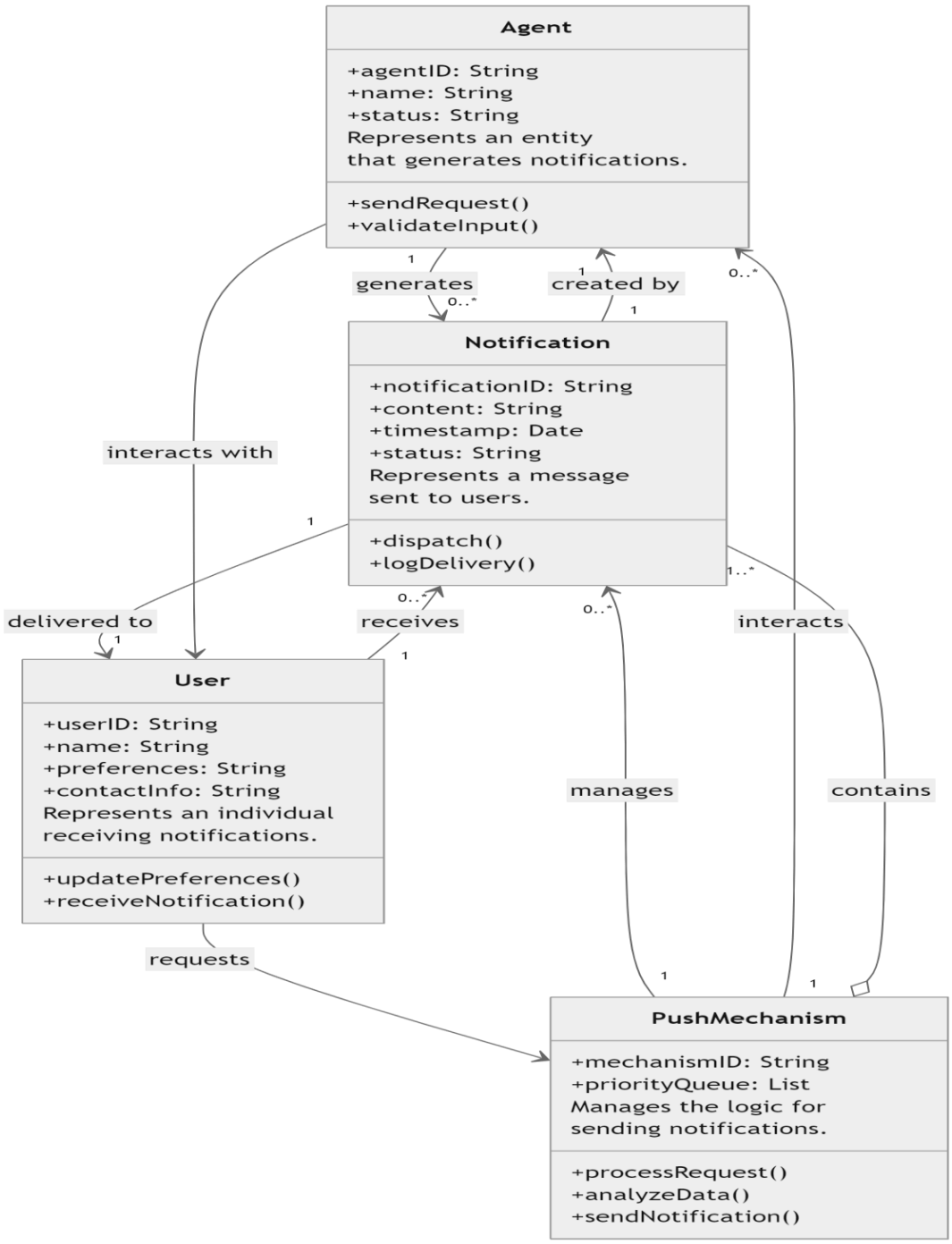


Figure 5.4.1 – Class Diagram for Agent Based Push Mechanism

5.5 Use-Case Model

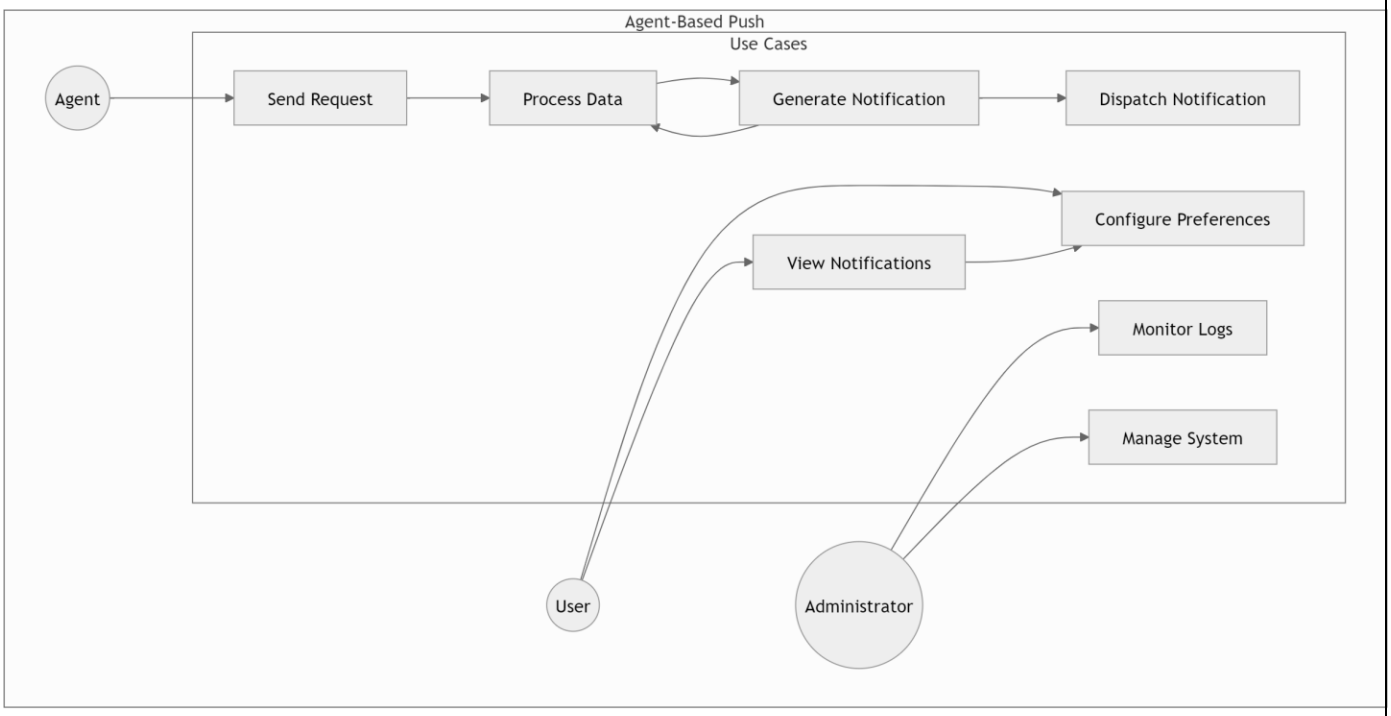


Figure 5.5.1 – Use case diagram for Agent Based Push Mechanism

5.6 Behaviors Diagrams

5.6.1 Sequence Diagram

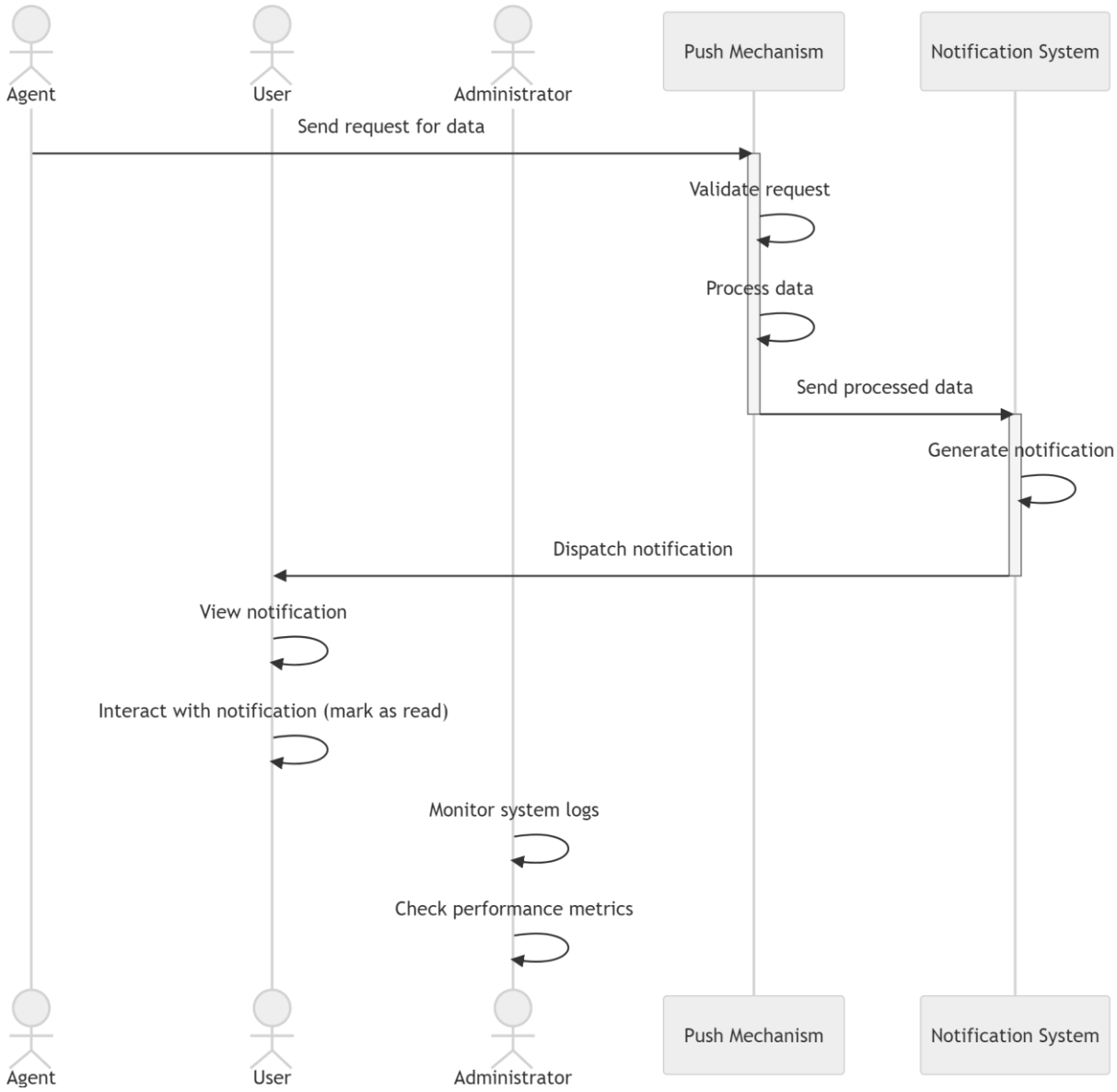


Figure 5.6.1 – Sequence Diagram for E – learning resource management

5.7 Assumptions and Dependencies

- The system assumes that all core technologies—**Node.js**, **Python**, **MySQL**, and machine learning frameworks (such as **Keras** and **PyTorch**)—are compatible, available, and supported for use throughout development and deployment.
- Third-Party Services: Assumes successful integration with external services like payment gateways, cloud storage, and external APIs for additional functionality.
- Internet Connectivity: Requires stable internet connectivity for cloud-based storage and real-time updates.
- User Base: Assumes a diverse user base with varying technical skill levels and access to modern devices (laptops, tablets, smartphones).
- Scalability: The system's infrastructure must support scaling based on user demand, particularly during peak usage times, without performance degradation.
- Legal Compliance: Assumes adherence to relevant data protection regulations, such as GDPR for user data privacy.

6. Supporting Information

6.1 List Of Figures

7 Conclusion and Future Scope

1. Conclusion

The **Agent-Based Push Mechanism (ABPM)** plays a crucial role in addressing the dynamic educational needs of students, educators, and institutions. By streamlining the organization, delivery, and tracking of learning resources, it significantly enhances **accessibility** and **personalized learning** experiences. This system promotes efficient utilization of resources and supports diverse learning styles, empowering educators with **data-driven insights** to make

informed decisions. Moreover, ABPM reduces logistical challenges often associated with traditional educational methods. By integrating advanced technologies such as **cloud computing**, **artificial intelligence (AI)**, and **analytics**, it is reshaping the education sector, ensuring that **quality education** becomes accessible to a broader and more diverse audience.

1. **Future Scope**

The future of **Agent-Based Push Mechanism (ABPM)** is promising, driven by continuous technological advancements and the growing demand for digital education solutions. Key future directions include:

2. **AI-Driven Personalization**: Leveraging artificial intelligence to deliver hyper-personalized learning paths based on individual preferences, performance, and learning speed.
3. **Gamification and Interactivity**: Increasing engagement through gamified learning experiences and interactive content delivery.
4. **Blockchain for Certification**: Implementing blockchain technology to secure and authenticate learner credentials and certificates.
5. **Global Accessibility**: Expanding access to e-learning resources in remote and underprivileged areas by overcoming infrastructure limitations.
6. **Integration with Emerging Technologies**: Incorporating virtual reality (VR) and augmented reality (AR) for immersive learning experiences.
7. **Advanced Analytics**: Utilizing big data and machine learning for predictive analytics to identify trends, improve resource allocation, and enhance learning outcomes.
8. **Scalable Systems**: Building systems that can seamlessly handle the demands of growing user bases, especially in regions with increasing digital literacy.

8 Concerns / Queries / Doubts if any:

1. **Data Privacy and Security**:
    - How will sensitive student and teacher information be protected?
    - What encryption methods will be used for secure data transmission?
    - How will we ensure compliance with data protection regulations like GDPR or CCPA?
2. **Scalability**:
    - Will the system handle an increasing number of users effectively?
    - Are there provisions to scale up storage and computational resources as user demands grow?
3. **User Accessibility**:
    - Is the platform designed to be inclusive and accessible for users with disabilities (e.g., support for screen readers or keyboard navigation)?
    - How will we handle varying levels of digital literacy among users?
4. **AI and Recommendation Systems**:

- How effective will the recommendation algorithms be in personalizing content for learners?
- What if the recommendations are biased or irrelevant?