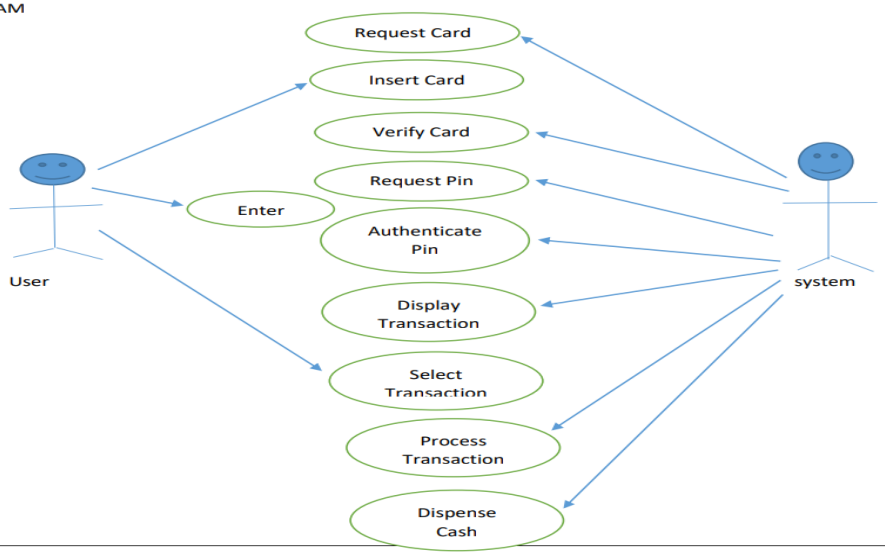
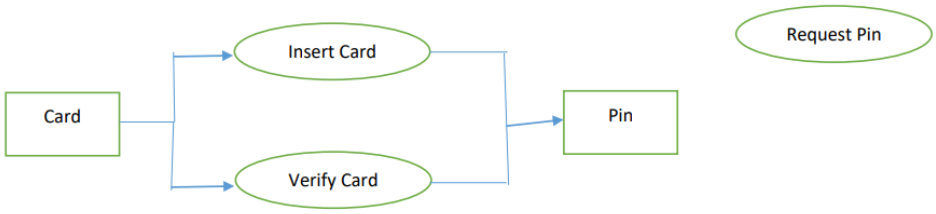
EXAMPLE SOFTWARE PROJECT: ATM SOFTWARE SYSTEM						
SOFTWARE DIVERSITY (SOFTWARE TYPE)	SOFTWARE PROCESS METHODS	SOFTWARE PROCESS MODELS	FOUR FUNDAMENTAL ACTIVITIES	REQUIREMENTS/SPECIFICATIONS		
1. Stand-Alone Apps 2. Interactive Transaction Based Apps 3. Embedded System 4. Data Collection Systems 5. Systems for Modelling and Simulations 6. Batch Processing System 7. Entertainment System 8. Information System 9. System of Systems	AGILE PLAN-DRIVEN	WATERFALL (STRICT sequential model) INCREMENTAL (RAPID APPLICATION DEVELOPMENT) SPIRAL RE-USE ORIENTED	REQUIREMENTS/ SPECIFICATIONS DEVELOPMENT (DESIGN & CODING) TESTING (VERIFICATION & VALIDATION) EVOLUTION/ MAINTENANACE	FUNCTIONAL REQUIREMENTS . Withdraw Cash .Transfer Cash .Deposit Cash .Check Balance .Print Statement .Change PIN .Pay Bills .Loan Request	FUNCTIONAL USER REQUIREMENTS . Users should be able to withdraw cash .System should enable users to transfer money .Users should be able to deposit cash Etc.	FUNCTIONAL SYSTEM REQUIREMENTS . System shall request card .User insert card to system .System shall verify card .System shall request pin . System authenticate user pin .System display transaction types on screen . User select transaction type . System process transaction . System returns card . System dispense cash . System gives receipt
				NON-FUNCTIONAL REQUIREMENTS . Dependability		

USE-CASE DIAGRAM



ACTIVITY DIAGRAM



USE CASE DESCRIPTIONS

USE CASE
Request Card: kdjkfjkfkjfjkfdkjkjf

Insert Card: jdfkjfdkjkjfdjkfkdfd

ACTORS:
System: jhsdjhdshjdsdshjdsjhfdjhfd

User: jhfjhfdfdkjfdkfd

What is the most important difference between generic software product development and custom software development? What might this mean **in practice for users of generic software products?**

The essential difference is that in generic software product development, the specification is owned by the product developer. For custom product development, the specification is owned and controlled by the customer. The implications of this are significant – the developer can quickly decide to change the specification in response to some external change (e.g. a competing product) but, when the customer owns the specification, changes have to be negotiated between thecustomer and the developer and may have contractual implications. For users of generic products, this means they have no control over the software specification so cannot control the evolution of the product. The developer may decide to include/exclude features and change the user interface. This could have implications for the user's business processes and add extra training costs when new versions of the system are installed. It also may limit the customer's flexibility to change their own business processes.

1.3 What are the four important attributes that all professional software should have? Suggest four other attributes that may sometimes be significant.

Four important attributes are maintainability, dependability, performance and usability. Other attributes that may be significant could be reusability (can it be reused in other applications), distributability (can it be distributed over a network of processors), portability (can it operate on multiple platforms e.g laptop and mobile platforms) and inter-operability (can it work with a wide range of other software systems). *Decompositions of the 4 key attributes e.g. dependability decomposes to security, safety, availability, etc. is also a valid answer to this question.*

Problems and challenges for software engineering

There are many possible challenges that could be identified. These include:

1. Developing systems that are energy-efficient. This makes them more usable on low power mobile devices and helps reduce the overall carbon footprint of IT equipment.
2. Developing validation techniques for simulation systems (which will be essential in predicting the extent and planning for climate change).
3. Developing systems for multicultural use
4. Developing systems that can be adapted quickly to new business needs
5. **Designing systems for outsourced development**
6. Developing systems that are resistant to attack
7. Developing systems that can be adapted and configured by end-users
8. Finding ways of testing, validating and maintaining end-user developed Systems

Advantages of certification

• Certification is a signal to employers of some minimum level of competence.
• Certification improves the public image of the profession.
• Certification generally means establishing and checking educational standards and therefore a mechanism for ensuring course quality.
• Certification implies responsibility in the event of disputes. Certifying body is likely to be accepted at a national and international level as 'speaking for the profession'.
• Certification may increase the status of software engineers and attract particularly able people into the profession.

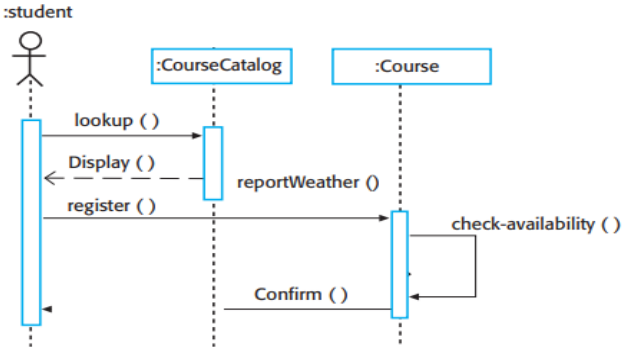
Disadvantages of certification

• Certification tends to lead to protectionism where certified members tend not to protect others from criticism.
• Certification does not guarantee competence merely that a minimum standard was reached at the time of certification.
• Certification is expensive and will increase costs to individuals and organisations.
• Certification tends to stultify change. This is a particular problem in an area where technology developments are very rapid.

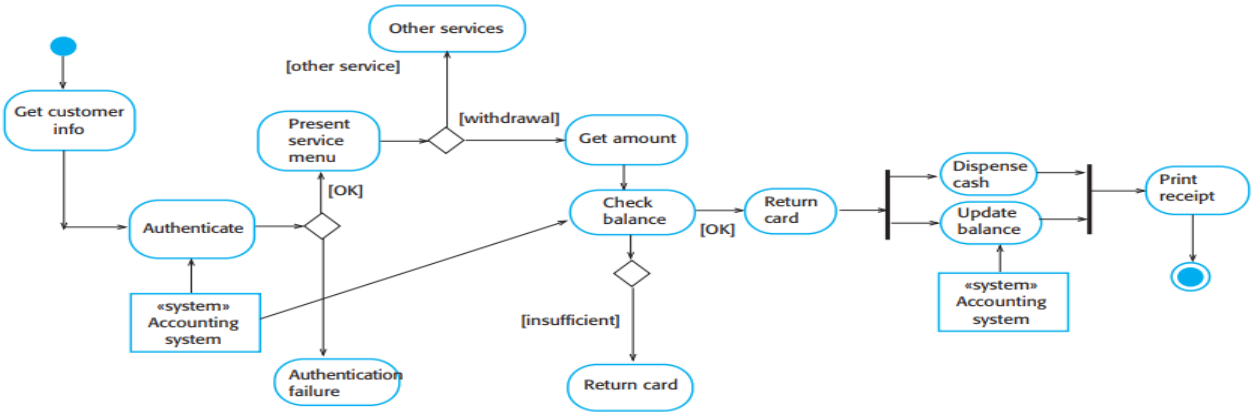
Giving reasons for your answer based on the type of system being developed, suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following systems:

• A system to control anti-lock braking in a car • A virtual reality system to support software maintenance • A university accounting system that replaces an existing system • An interactive travel planning system that helps users plan journeys with the lowest environmental impact 1. Anti-lock braking system This is a safety-critical system so requires a lot of up-front analysis before implementation. It certainly needs a plan-driven approach to development with the requirements carefully analysed. A waterfall model is therefore the most appropriate approach to use, perhaps with formal transformations between the different development stages. 2. Virtual reality system This is a system where the requirements will change and there will be an extensive user interface components. Incremental development with, perhaps, some UI prototyping is the most appropriate model. An agile process may be used. 3. University accounting system This is a system whose requirements are fairly well-known and which will be used in an environment in conjunction with lots of other systems such as a research grant management system. Therefore, a reuse-based approach is likely to be appropriate for this. 4. Interactive travel planning system System with a complex user interface but which must be stable and reliable. An incremental development approach is the most appropriate as the system requirements will change as real user experience with the system is gained.

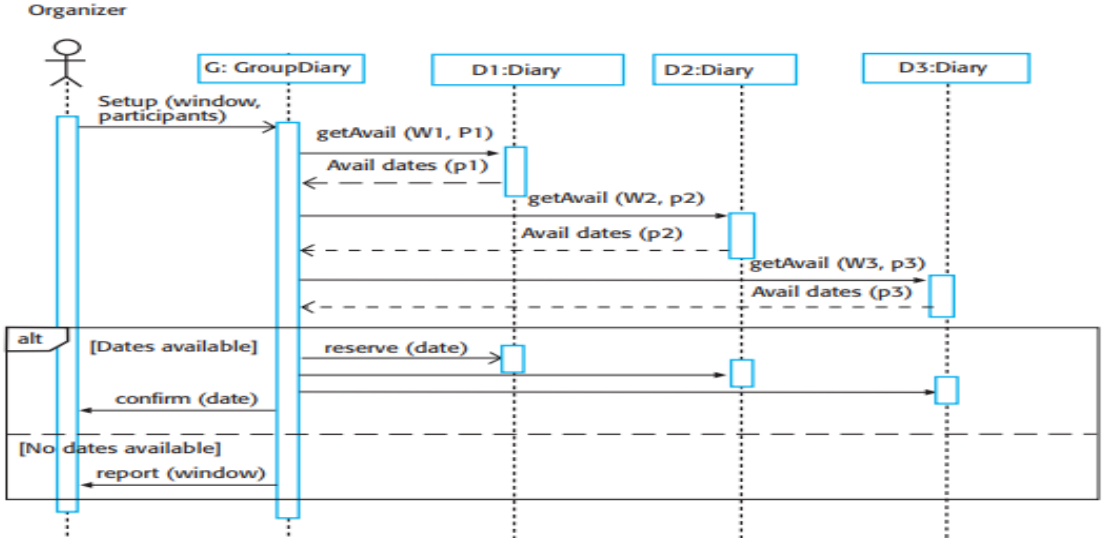
Develop a sequence diagram showing the interactions involved when a student registers for a course in a university. Courses may have limited enrolment, so the registration process must include checks that places are available. Assume that the student accesses an electronic course catalog to find out about available courses.



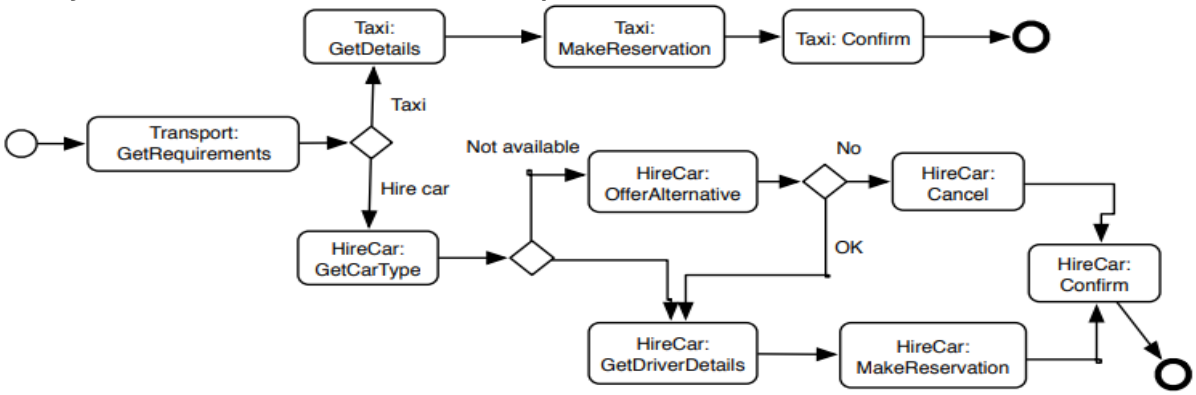
Based on your experience with a bank ATM, draw an activity diagram that models the data processing involved when a customer withdraws cash from the machine



Draw a sequence diagram showing the interactions of objects in a group diary system, when a group of people are arranging a meeting.



For the example of the vacation package reservation service, design a workflow that will book ground transportation for a group of passengers arriving at an airport. They should be given the option of booking either a taxi or a hire car. You may assume that the taxi and car hire companies offer web services to make a reservation.



QUESTION :discuss the 3 key characteristics of the engineering of web based software engineering

1. Distributed Nature: Web-based software operates across distributed environments, involving multiple servers, databases, and clients. Engineers must design systems that can handle concurrent users and large volumes of data while maintaining consistent functionality and responsiveness.
2. Interoperability: Web-based applications integrate various components, services, and technologies. Engineers ensure that these applications can communicate seamlessly with different platforms, devices, and protocols using standardized techniques like HTTP, RESTful APIs, or web services.
3. Security: Security is a critical concern due to the risks associated with operating over the internet. Engineers implement robust security measures to protect sensitive data, authenticate users, and mitigate threats such as unauthorized access, data breaches, and injection attacks.

QUESTION: explain why it is important to make distinction between developing the user requirements and developing the system requirements in the requirements engineer process

distinguishing between user requirements and system requirements is essential for understanding user needs, aligning with business objectives, defining project scope, designing user-centric systems, mitigating risks, facilitating communication and collaboration, and ensuring traceability and validation throughout the development process.

QUESTION: explain why change is inevitable in the development of complex systems and discuss 2 ways of dealing with change....

Change is inevitable in the development of complex systems due to various factors such as evolving user needs, technological advancements, market dynamics, and organizational requirements. Here are two ways of dealing with change:

1. Agile Methodology: Embraces change as a natural part of development. It emphasizes iterative, incremental work with frequent feedback loops. Agile teams prioritize collaboration, communication, and adaptation to changing requirements throughout the project.
2. Change Management Processes: Involves systematic assessment of proposed changes, evaluating their impact on scope, schedule, budget, and resources. Change management ensures changes are properly documented, communicated, and integrated

QUESTION: explain why design conflicts might arise when designing an architecture for which both availability and security requirements are the most important non functional requirements

When designing an architecture where both availability and security are crucial:

1. Availability: Ensures continuous access and operation, typically through redundancy and fault tolerance mechanisms.
2. Security: Protects against unauthorized access and data breaches using encryption, authentication, and access controls.

Conflicts may arise due to:

1. Trade-offs: Increasing redundancy for availability may heighten vulnerability to security breaches.
2. Performance Impact: Stringent security measures may degrade performance, affecting availability.
3. User Experience vs. Access Control: Strict security measures might hinder user access and satisfaction.
4. Scalability Challenges: Scaling to meet demand while maintaining security requires careful consideration.

QUESTION: what is meant by software process model ? Briefly explain 3 examples of software processing models indicating when each might be used

Software process models are structured approaches to software development. Here are three examples:

a software process model is a simplified representation of a software process

1. Waterfall Model: A sequential approach with phases progressing linearly. Suitable for projects with stable and well-defined requirements. Used in small-scale or straightforward projects like simple mobile apps or websites.
2. Agile Model: An iterative and incremental approach emphasizing flexibility and collaboration. Breaks development into small iterations with continuous feedback. Ideal for dynamic projects with evolving requirements, such as software products with frequent updates or enhancements.
3. Spiral Model: Combines elements of waterfall and iterative approaches, focusing on risk management and iteration. Involves multiple cycles, addressing high-risk areas early. Suitable for large-scale projects with complex requirements and uncertainties, such as mission-critical systems or projects involving cutting-edge technologies.

Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.

Project Topic: Student Academia Copilot (SAC_AI).

Introduction:

The AI Academia Chatbot project proposes the development of an innovative AI-powered platform tailored for academia. This platform will utilize advanced natural language processing (NLP) techniques to create a dynamic chatbot capable of engaging with users and providing personalized learning experiences based on academic course materials.

Statement of Problem:
Traditional methods of studying often lack interactivity and fail to adapt to individual learning needs. Students struggle to efficiently access and engage with vast amounts of academic content available across various disciplines.

Project Aim:
The aim of this project is to create an AI-powered chatbot that serves as an interactive study companion for students, leveraging AI algorithms to process and understand academic course materials effectively.

Specific Objectives:
1. Implement RAG (Red, Amber, Green) patterns for tracking learning progress and adapting study recommendations.
2. Develop an AI agent capable of comprehending and responding to user queries related to academic topics.
3. Incorporate memory and retrieval patterns to facilitate efficient knowledge recall during interactions.
4. Utilize prompt engineering techniques to generate contextually relevant responses and engage users effectively.

Deliverables:
1. Functional AI Academia Chatbot platform accessible via web and mobile devices.
2. Documentation detailing chatbot functionalities, including user guides and technical specifications.
3. Comprehensive testing reports demonstrating the effectiveness and reliability of the chatbot's capabilities.
4. Training materials and resources for administrators to further enhance the chatbot's performance and expand its knowledge base.

Justification for Project:
The project addresses the pressing need for personalized and interactive learning solutions in academia. By harnessing the power of AI, we can revolutionize the way students engage with course materials, leading to improved learning outcomes and retention rates.

Motivation for Project:
The motivation behind this project lies in empowering students with accessible and engaging learning tools that adapt to their individual needs and preferences. By leveraging AI technology, we can democratize access to quality education and foster a culture of lifelong learning.

Project Scope:

The project will focus on developing the core functionalities of the AI Academia Chatbot, including RAG pattern integration, AI agent implementation, memory and retrieval mechanisms, and prompt engineering techniques. Initial training will involve feeding the chatbot with various academic course materials to ensure comprehensive knowledge coverage.

Project Method:

The project will employ an agile development approach, with iterative cycles of design, implementation, testing, and feedback incorporation. Development tools such as Python programming language, TensorFlow, and OpenAI GPT will be utilized to build and train the chatbot model. Continuous evaluation and refinement will be conducted to optimize the chatbot's performance and user experience. Additionally, user feedback loops will be established to gather insights and prioritize feature enhancements throughout the development process.

Functional User Requirements:

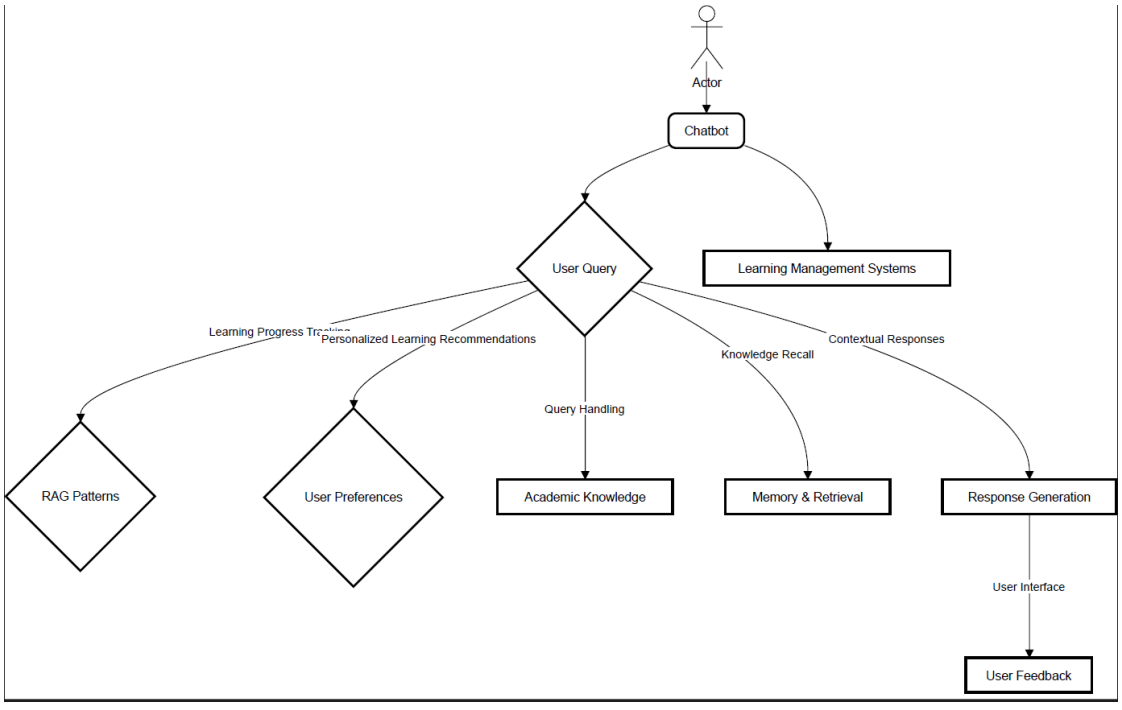
- **Learning Progress Tracking:** The chatbot should track the user's learning progress using RAG (Red, Amber, Green) patterns.
- **Personalized Learning Recommendations:** The chatbot should provide personalized study recommendations b ased on the user's learning progress and preferences.
- **Query Handling:** The chatbot should be able to comprehend and respond to user queries related to academic top ics.
- **Knowledge Recall:** The chatbot should be able to efficiently recall relevant knowledge to answer user queries.
- **Contextual Responses:** The chatbot should generate contextually relevant responses that engage and assist the user.
- **User Interface:** The chatbot should have a user-friendly interface that is easy to navigate and use.

Functional System Requirements

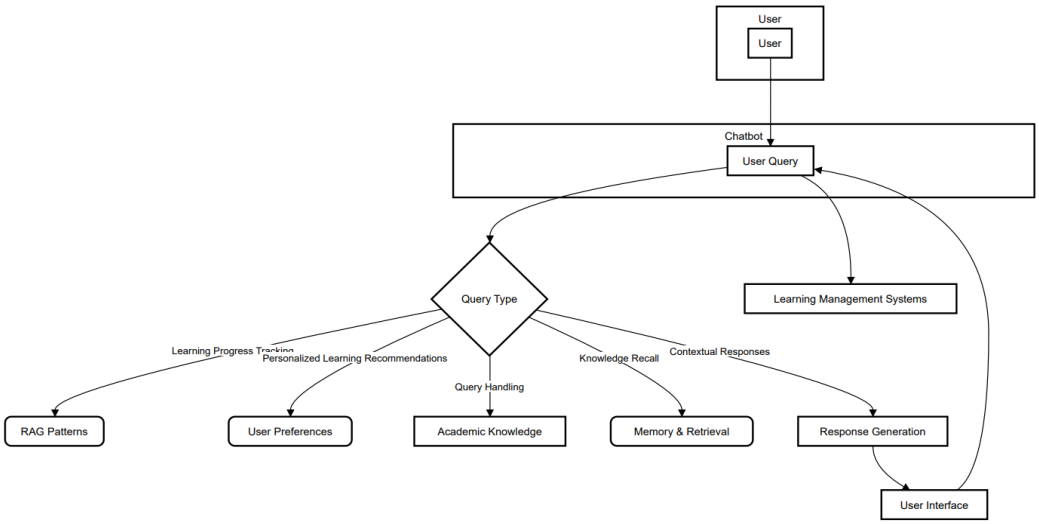
- **Natural Language Processing:** The chatbot should utilize advanced NLP techniques to process and understand u ser queries.
- **Knowledge Base:** The chatbot should have a comprehensive knowledge base covering various academic discipli nes.
- **AI Agent:** The chatbot should be equipped with an AI agent capable of learning and adapting to the user's needs.
- **Memory and Retrieval Mechanisms:** The chatbot should incorporate memory and retrieval patterns to facilitate efficient knowledge recall.
- **Prompt Engineering:** The chatbot should leverage prompt engineering techniques to generate contextually releva nt responses.
- **Scalability:** The chatbot should be scalable to handle a large number of users and interactions.
- **Security:** The chatbot should meet industry standards for data security and privacy.
- **Integration:** The chatbot should be easily integrated with existing learning management systems.

UML DIAGRAMS FOR THIS PROJECT

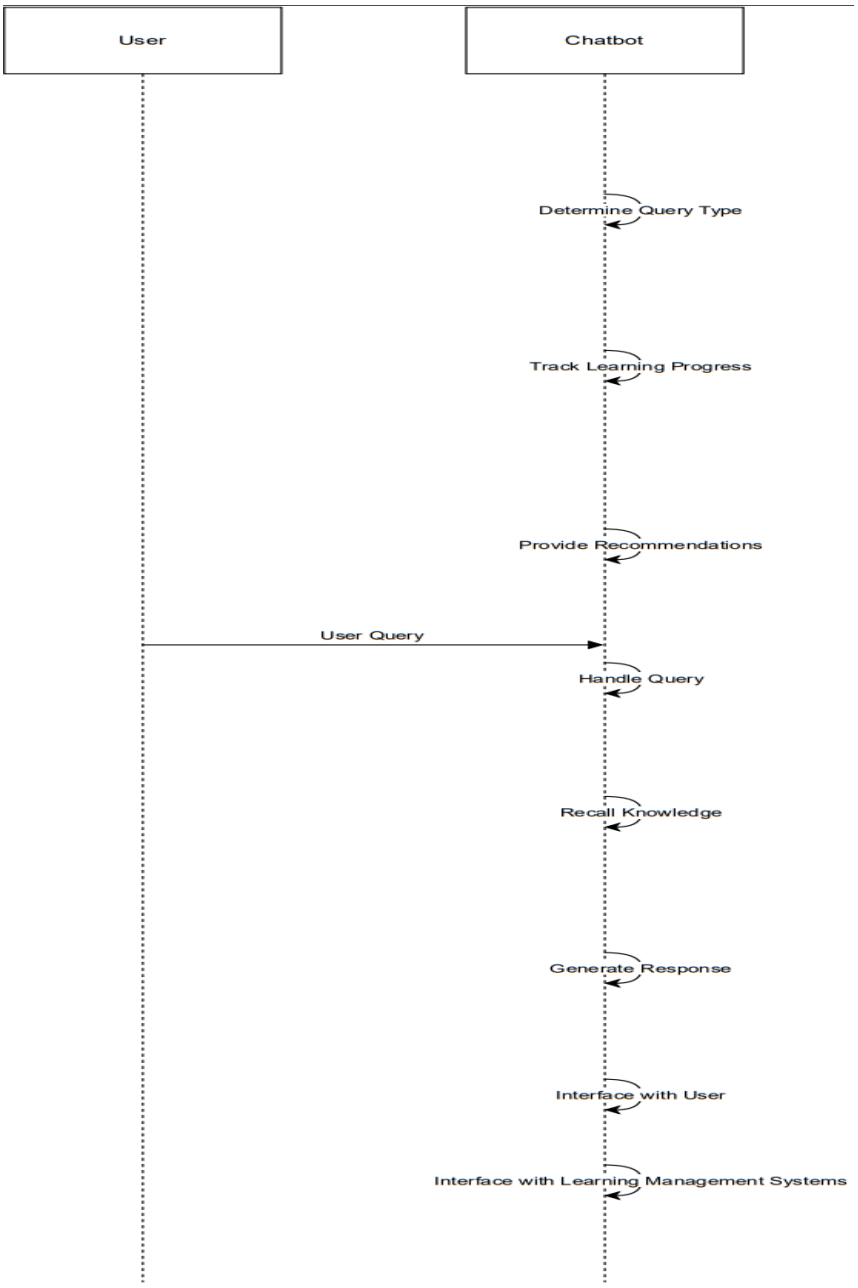
USE CASE DIAGRAM:



ACTIVITY DIADRAM:



SEQUENCE DIAGRAM:



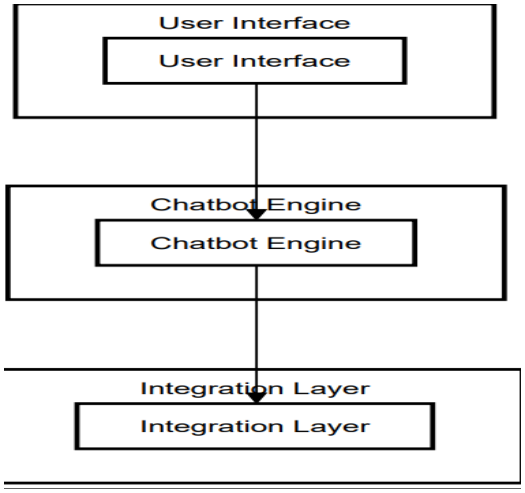
ARCHITECTURAL DESIGN:

The architectural design of the AI Academia Chatbot comprises three main components: the User Interface, the Chatbot Engine, and the Integration Layer.

1. User Interface: This component serves as the interface between the user and the chatbot. It provides an intuitive and user-friendly platform for users to interact with the chatbot, receive responses, and provide feedback.

2. Chatbot Engine: The core of the system, the Chatbot Engine, is responsible for processing user queries, implementing learning progress tracking, generating personalized recommendations, handling queries related to academic topics, recalling relevant knowledge, and generating contextually relevant responses using NLP techniques.

3. Integration Layer: This component facilitates integration with external systems, particularly learning management systems, to enhance the chatbot's functionality and scalability.



COMPONENT DESIGN:

The component design of the AI Academia Chatbot comprises several modular components, each responsible for specific functionalities within the system.

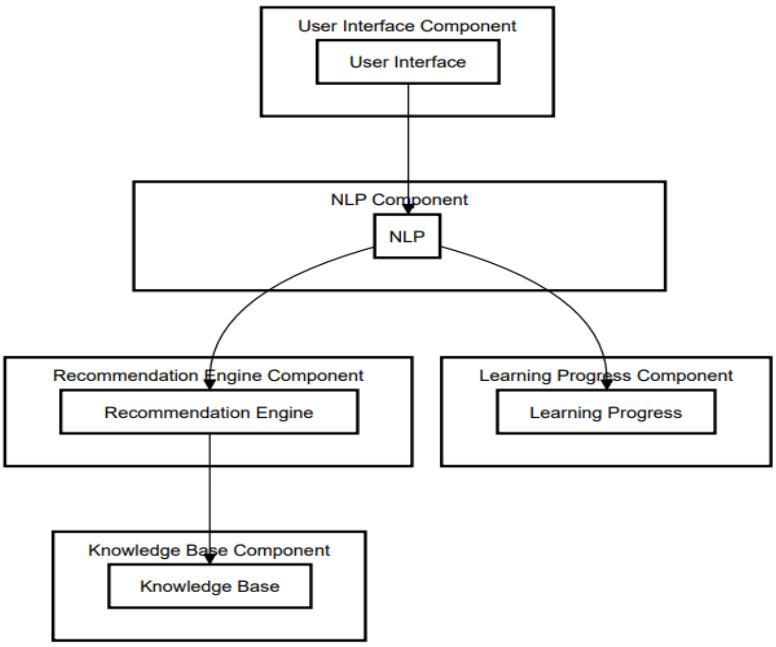
1. User Interface Component: This component handles user interaction, providing an intuitive interface for users to communicate with the chatbot and receive responses.

2. NLP Component: The Natural Language Processing (NLP) component processes user queries, understands their intent, and extracts relevant information to generate appropriate responses.

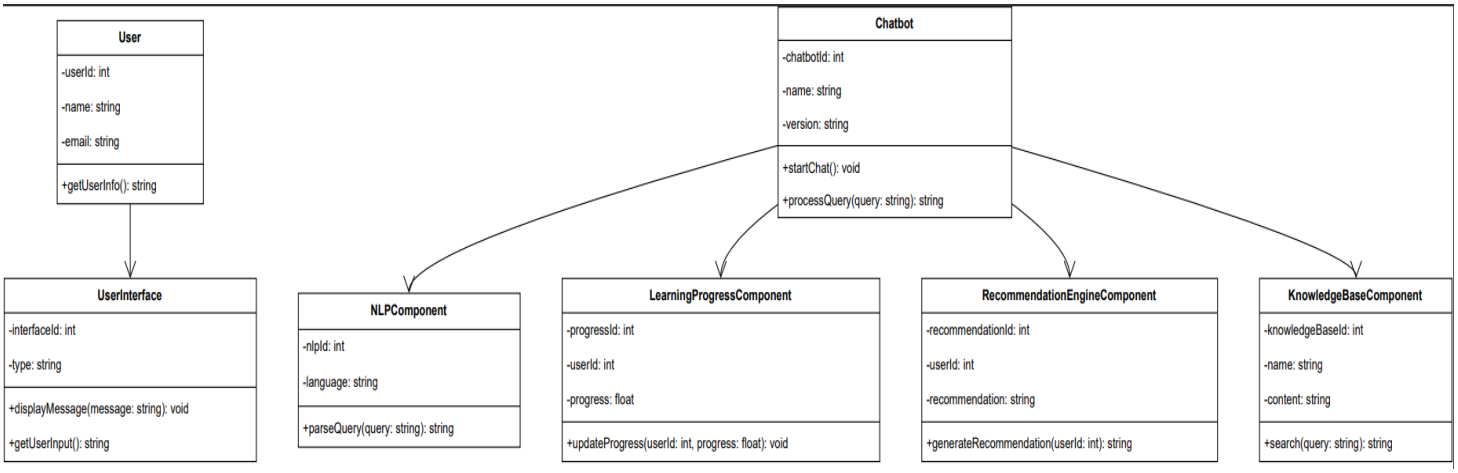
3. Learning Progress Component: Responsible for tracking user learning progress using RAG patterns, this component analyzes user interactions and adjusts recommendations accordingly.

4. Recommendation Engine Component: This component generates personalized study recommendations based on user preferences, learning progress, and the content of academic materials.

5. Knowledge Base Component: Stores and manages the repository of academic content and knowledge used by the chatbot to respond to user queries effectively.



CLASS DIAGAM:



USER INTERFACE DESIGN: