CHAPTER 3

REQUIREMENT AND ANALYSIS

# Problem Definition

The primary challenge is creating a seamless and engaging AI Companion platform where users can create, interact with, and customize AI personas modeled after celebrities, historical figures, or

fictional characters. The system must support secure, scalable, and personalized interactions while providing real-time conversational capabilities.

**Common Issues with Existing Systems:**

* + - Limited Personalization: Existing platforms often have insufficient customization, reducing user satisfaction.
    - Slow Response Times: Many applications suffer from latency due to suboptimal back-end performance.
    - Inconsistent Personalities : AI personas fail to maintain consistent behaviour or across conversations, breaking immersion.
    - Security Concerns : Insufficient authentication measures, leading to user data breaches or leaks in chat history.

# Requirement Specification

## Functional Requirements

Functional requirements are a set of specifications that outline the functions a system or component must perform. They're like a checklist for a system and are essential for developers to implement so that users can accomplish their tasks

1. AI Companion Creation : Users can choose from predefined templates like celebrities or historical figures and personalize the appearance, voice, and personality. The backend will store these configurations and ensure the AI responds accordingly based on these personalized attributes.
   * Example : A user can choose to chat with a historical figure like Napoleon Bonaparte, tweak his personality to be more humorous, and give him a modern voice.
2. Dynamic Conversations: Users can interact with their companions via text (and possibly voice in future expansions). The AI uses NLP (Natural Language Processing) to analyze user queries and respond appropriately.
   * Example : If a user asks, “Tell me something about physics,” and their companion is modeled after Albert Einstein, the system will access preloaded knowledge or external resources to respond accurately.
3. User Profile Management : Users should have control over their profiles, view their chat history, manage subscription preferences, and customize their companion's appearance.
4. Payment Handling : Through Stripe integration, the platform must support payment options like credit cards, subscription plans, and one-time payments for premium features or advanced AI companion customization.
5. Secure Authentication: Using Clerk , secure user sessions, registration, and login will be managed with support for OAuth or third-party logins like Google or Facebook.

## Non-Functional Requirements

Non-functional requirements define how the system performs certain functions rather than what it does.

* + - * **Performance** : The platform should be able to support at least 1,000 concurrent users initially, with sub-500 ms latency per user interaction.
      * **Scalability**: Built using Vercel and cloud-hosted MySQL , the system should dynamically scale to accommodate growing users.
      * **Security** : Secure user data (chats, profiles, payment info) using encryption (AES-256 for data-at-rest, HTTPS/SSL for data-in-transit).
      * **Usability** : Ensure a responsive, easy-to-navigate UI so users can create and interact with companions without technical difficulties.
      * **Availability**: Target a 99.9% uptime using high-availability infrastructure, backup databases, and redundant server setups.
      * **Maintainability** : The codebase should be modular and use industry-standard patterns to allow future updates, like adding new AI companions, without disrupting existing

functionality.

# Planning and Scheduling

## Milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start Date** | **End Date** |
| **Selection of Project** |  |  |  |
| **Requirement**  **Gathering** |  |  |  |
| **System Analysis** |  |  |  |
| **Survey of**  **Technology** |  |  |  |
| **Designing of UML Diagram** |  |  |  |
| **Chapter No 1,2,3,4**  **[Documentation]** |  |  |  |
| **Coding and**  **Implementation** |  |  |  |
| **Testing** |  |  |  |
| **Chapter No 5,6,7**  **[Documentation]** |  |  |  |
| **Final Deployment** |  |  |  |
| **Final Blackbook**  **Submission** |  |  |  |

* + 1. **Number of Person**

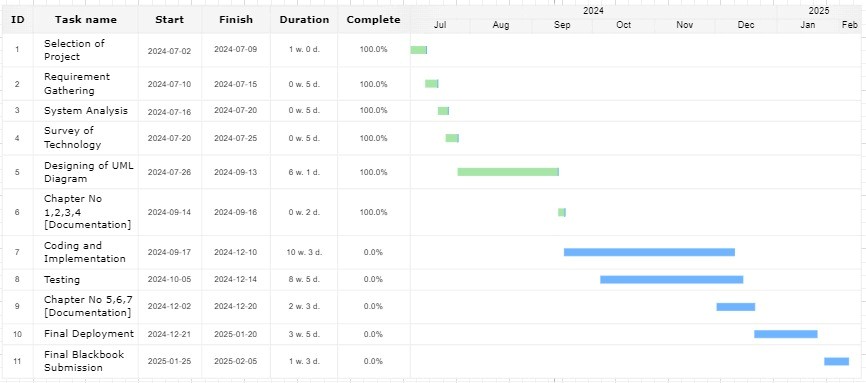
The project consist only one person- Anjali Suthar Schedule Start Day : 02/07/2024

Schedule End Day : 05/02/2025

## Gantt Chart

A Gantt Chart is a visual project plan that lists tasks and milestones on the vertical axis with time

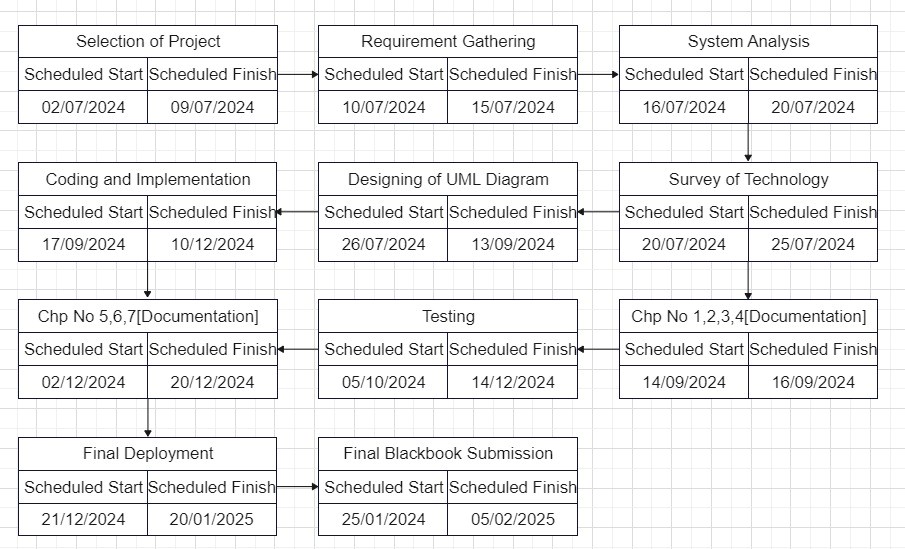
plotted on the horizontal axis. Gantt Charts are used in project management to schedule, track, and communicate deliverables, deadlines, dependencies, and resource assignments.



## Pert Chart

A PERT chart is a visual project management tool used to map out and track the tasks and timelies. The name PERT is an acronym for Project (or Program) Evaluation and Review Technique.PERT charts are similar to Gantt charts in that they offer a graphical view of a project’s tasks, schedule, and

timelines.



# Software and Hardware Requirements

## Software Requirements

* **Designing** : Figma or Adobe XD for UI/UX design.

### Development :

* + Languages : JavaScript/TypeScript
  + Frameworks : Next.js (for front-end), Prisma (ORM for MySQL), React.js
  + Libraries : Clerk (for authentication), Stripe (for payments)
* **Documentation** : GitHub or Confluence for documenting API references, user guides, and architecture.
* **Operational** : Vercel for hosting, MySQL for database, CI/CD pipeline with GitHub Actions.

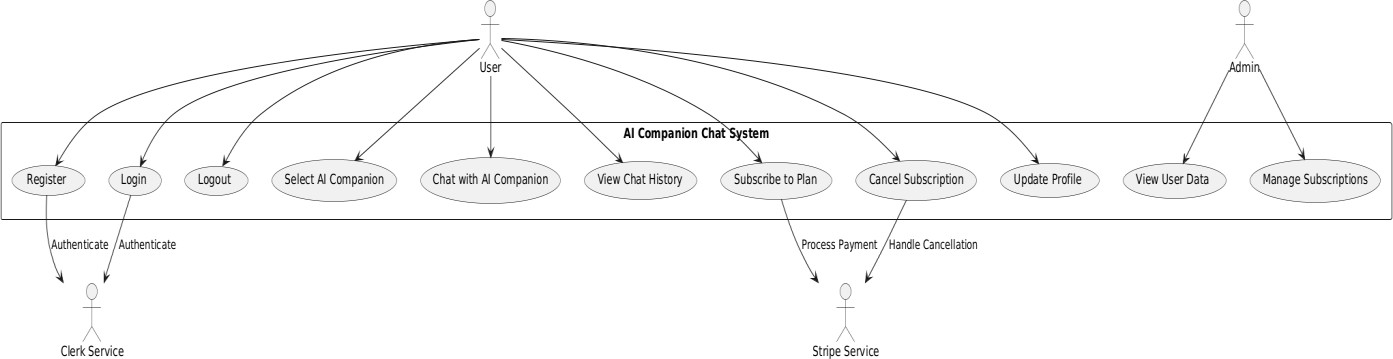
## Hardware Requirements

### Development Machines :

* + **RAM**: Minimum 16GB
  + **SSD**: Minimum 512GB
* **Server** Requirements :
  + **RAM**: 32GB+ (depending on the scale)
  + **SSD:** 1TB+ (for fast database and chat log access)

# Conceptual Models

## Use Case Diagram

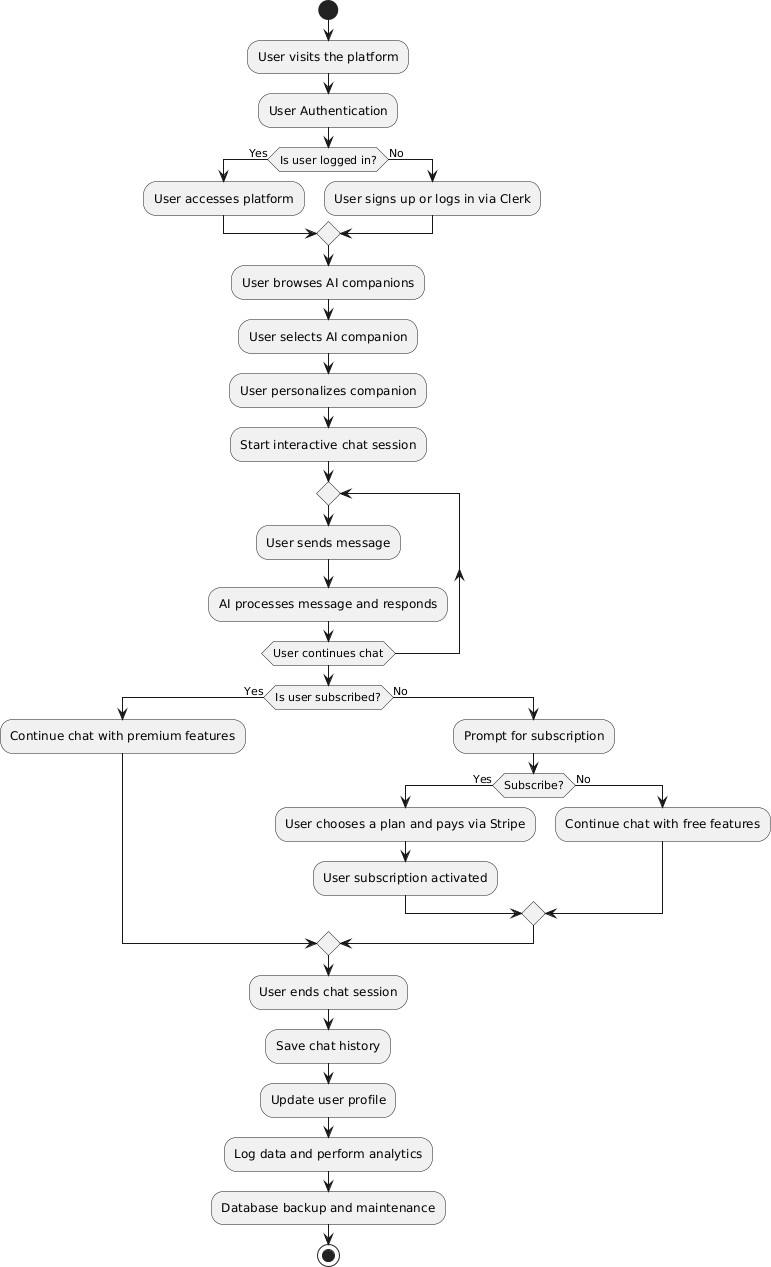


* + 1. Entity Relationship Diagram

## DFD (Data Flow Diagram)

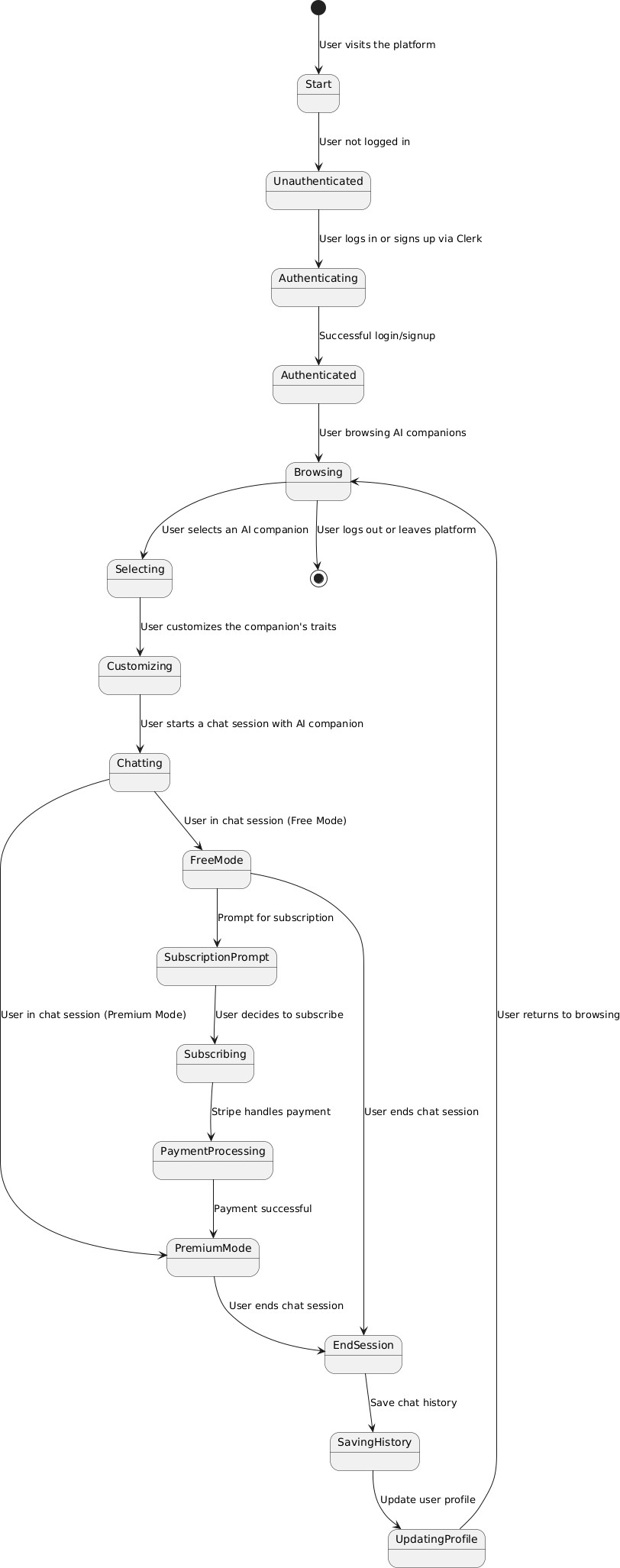
* Level 0 DFD : Shows basic flows like user requests, server responses, and data updates.
* Level 1 DFD : Detailed breakdown showing how AI Companion creation, chat interactions, and profile management interact with databases and external services.

## Activity Diagram

* Depicts how users interact with the system to create companions, customize them, and initiate chats.

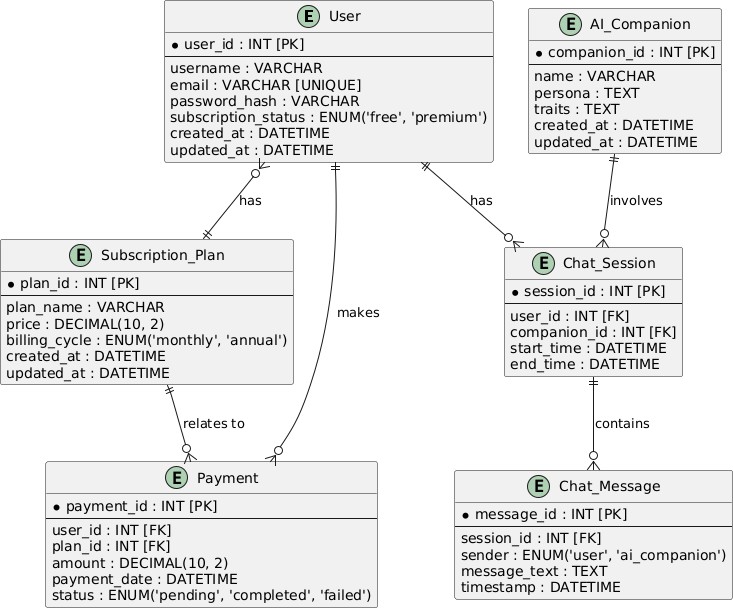
## State Transition Diagram

* Shows the transition states of AI companions (e.g., creation, customization, active interaction, idle, etc.).



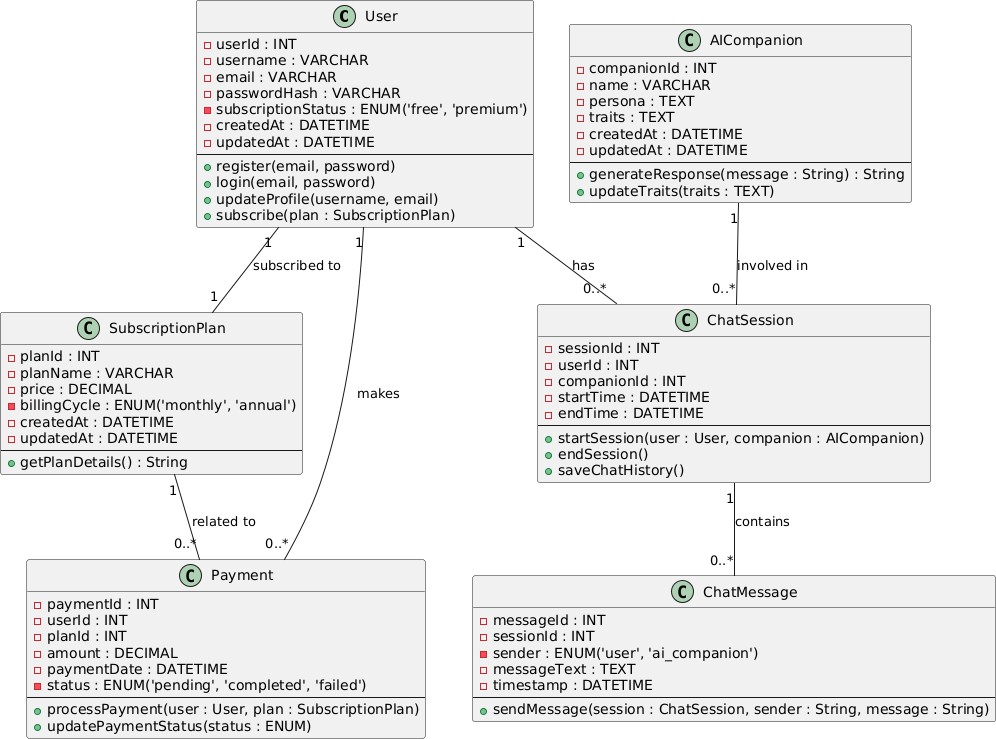
## ER Diagram

* Models the relationships between users, AI companions, chat logs, and transactions within the database.



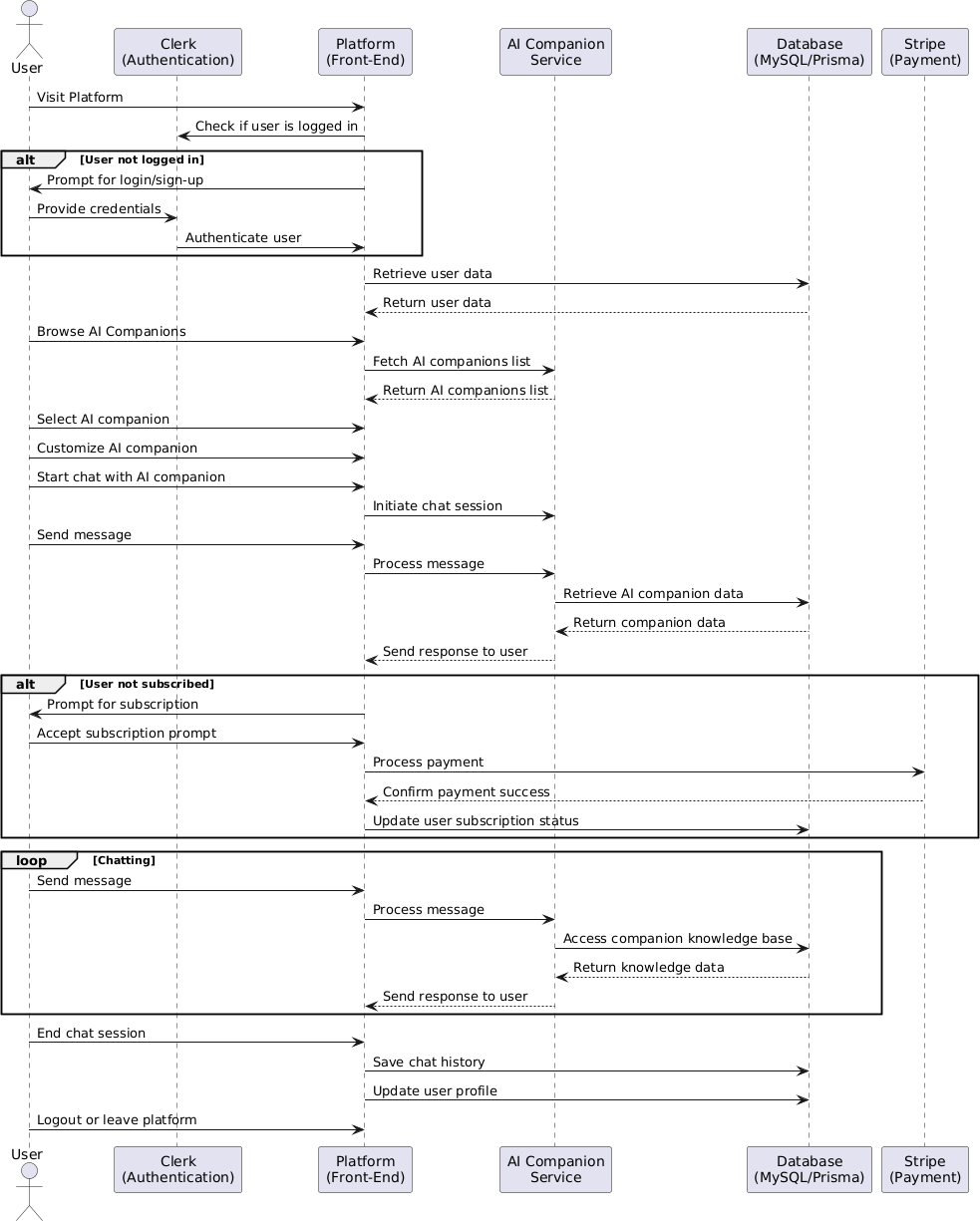
## Class Diagram

* Depicts classes for User, Companion, Subscription, Payment, and their relationships.



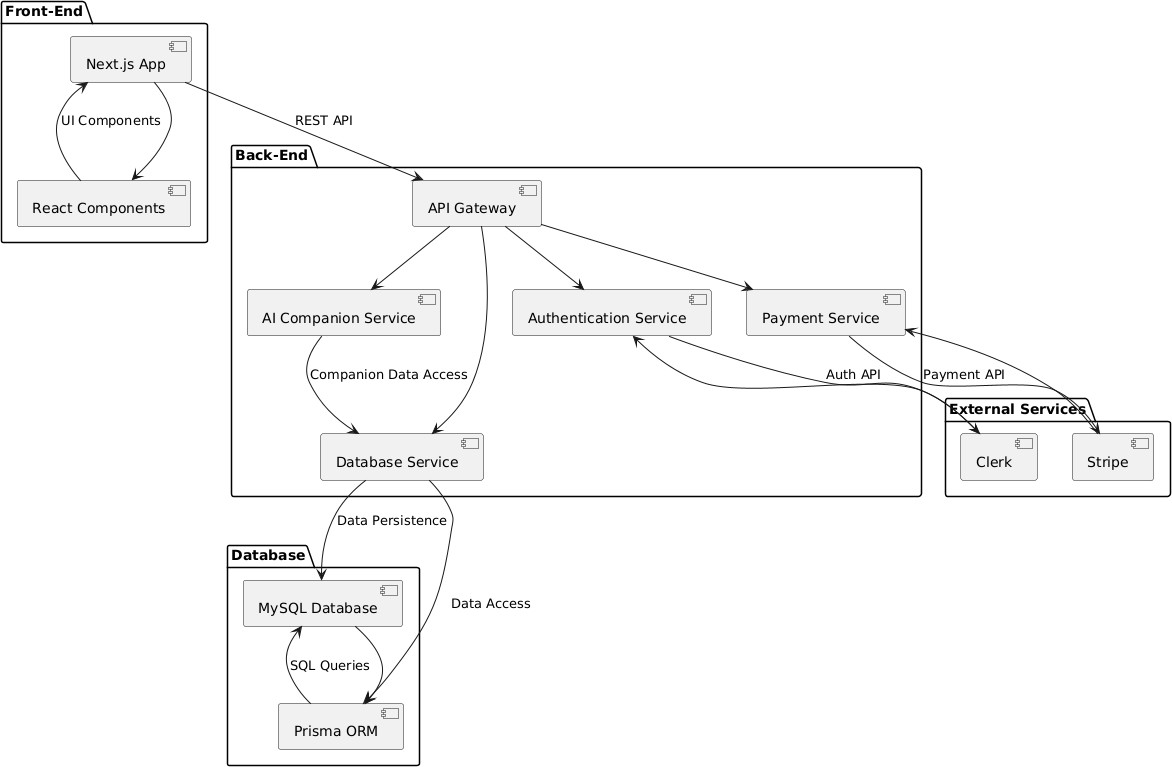
## Sequence Diagram

* Demonstrates interaction between users, AI services, payment systems (Stripe), and databases during a complete user session.



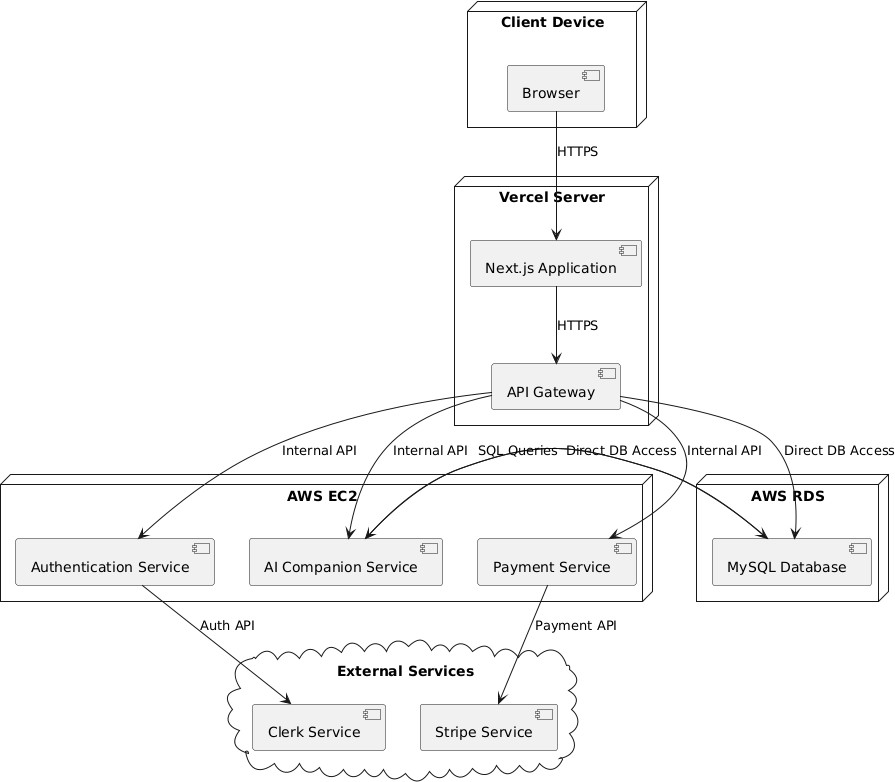
## Component Diagram

* Illustrates components such as front-end, back-end, database, payment gateway, and authentication service.



## Deployment Diagram

* Depicts deployment architecture on Vercel, integrating the database, user authentication, and payment systems.



## Event Table

* Maps out events like user login, AI companion creation, conversation start, and payment transactions with triggers and responses.