Software Requirements Specification (SRS) for Haze Launcher

---

1. Introduction

1.1 Topic Selection and Requirements Elicitation

Problem Description:
Gamers often rely on multiple platforms for game purchases, library management, achievement tracking, and social interaction, leading to fragmented experiences. **Haze Launcher** provides an all-in-one solution to streamline these activities by integrating a game store, library management, achievements, and a social network into a single platform.

1.2 Functional Requirements

1. **User Registration:** Users can create accounts with unique credentials.
2. **User Login:** Users can log in to access their accounts and libraries.
3. **Browse Games:** Users can view the game catalog.
4. **Game Purchase:** Users can buy games and update their libraries.
5. **Manage Friends:** Users can add, remove, and view friends.
6. **Track Achievements:** Users can view game achievements.

1.3 Non-Functional Requirements

1. **Data Storage:** Persist critical data across sessions.
2. **Performance:** Support 50 concurrent users with ≤3s response time.
3. **Security:** Encrypt sensitive user data.
4. **Scalability:** Adapt to new features and user growth.
5. **Availability:** Ensure 99.9% uptime.

---

2. Use Case Models

2.1 Use Case Diagram

Actors:

- **User:** Anyone who creates an account or logs into the system to interact with the platform.
- **System:** The backend of the application that manages data persistence, purchase transactions, etc.
- **Admin (optional):** Manages the game catalog, user accounts, and any administrative tasks.

Use Cases:

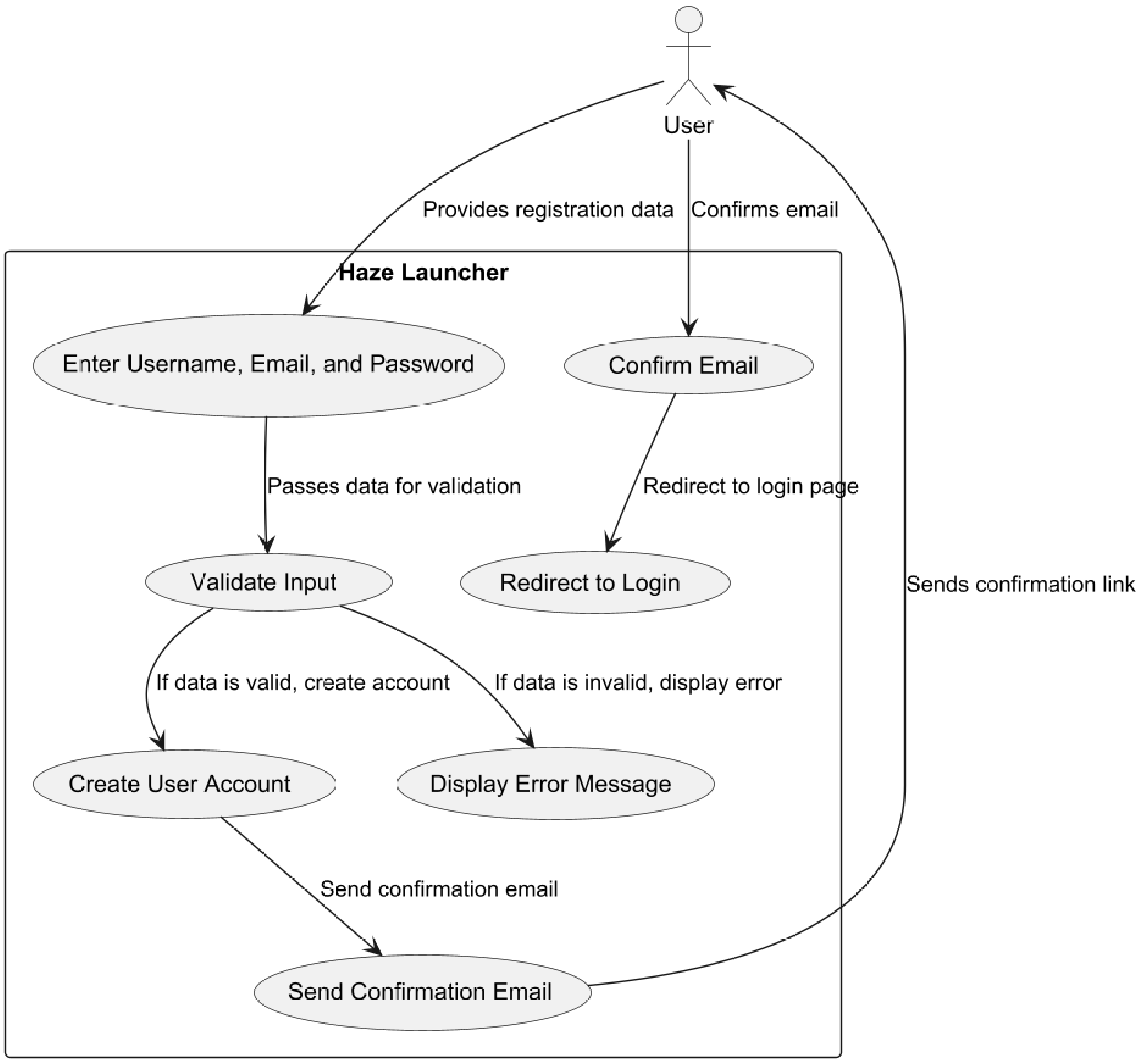
1. **User Registration**
2. **User Login**
3. **Browse Games**

4. **Game Purchase**
5. **Manage Friends**
6. **Track Achievements**

2.2 Detailed Use Case Descriptions

1. User Registration

Diagram:



Actor(s): User
Main Flow:

1. The user navigates to the registration page.
2. The user enters their username, email, and password.
3. The system validates the input (e.g., checks for a valid email format, password strength).
4. If valid, the system creates a new user account.

5. The system stores the user details in the database.
6. The system sends a confirmation message to the user's email.
7. The user verifies their account and is redirected to the login page.

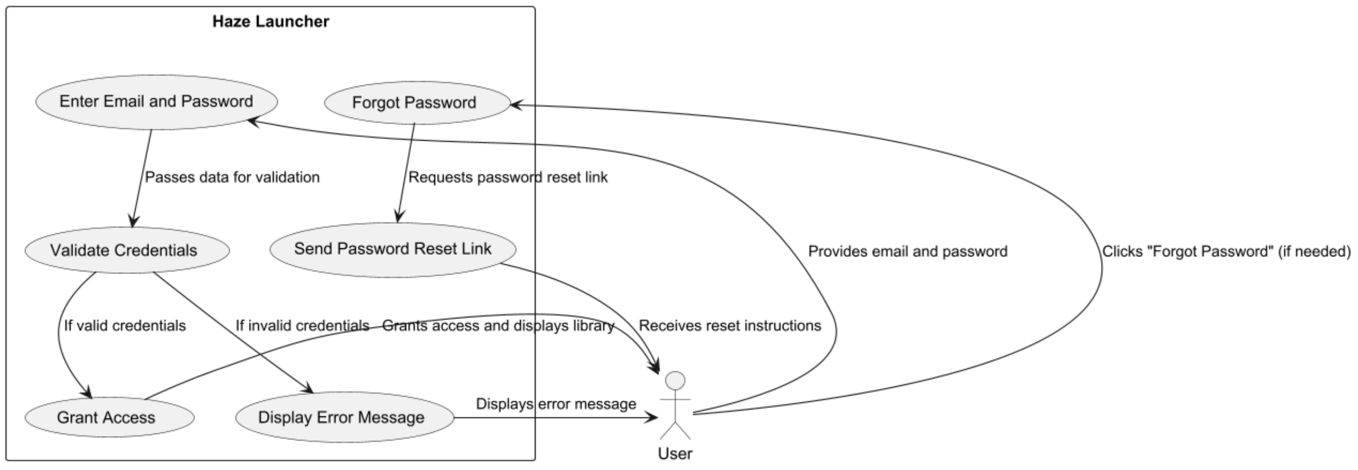
Alternative Flow:

- **A1:** If the email is already in use, the system displays an error message and prompts the user to choose a different one.
- **A2:** If the password does not meet security requirements, the system displays an error message.

---

2. User Login

Diagram:



Actor(s): User
Main Flow:

1. The user enters their email and password.
2. The system checks the credentials.
3. If correct, the system logs the user in and displays their library.
4. If incorrect, the system displays an error message.

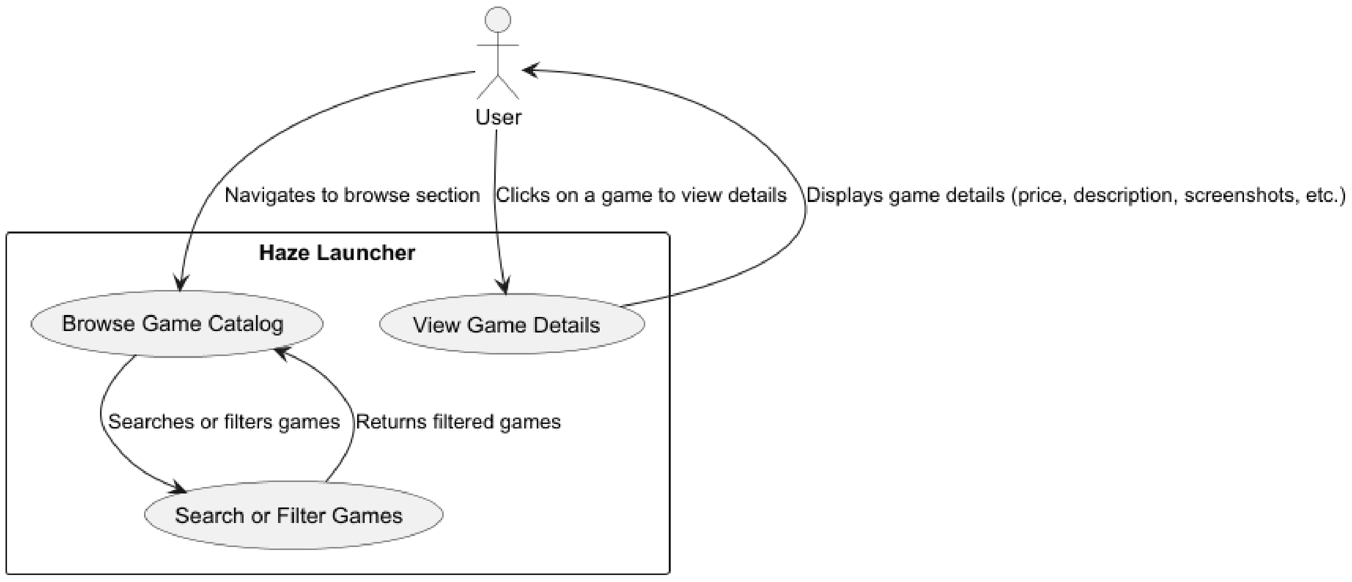
Alternative Flow:

- **A1:** If the credentials are incorrect, the system displays an error message.
- **A2:** If the user forgets their password, they can reset it via email.

---

3. Browse Games

Diagram:


User
Navigates to browse section
Clicks on a game to view details
Displays game details (price, description, screenshots, etc.)
Haze Launcher
Browse Game Catalog
View Game Details
Searches or filters games
Returns filtered games
Search or Filter Games

Actor(s): User
Main Flow:

1. The user navigates to the "Browse Games" section.
2. The system displays available games.
3. The user can filter and search for games.
4. The user clicks on a game to view more details.

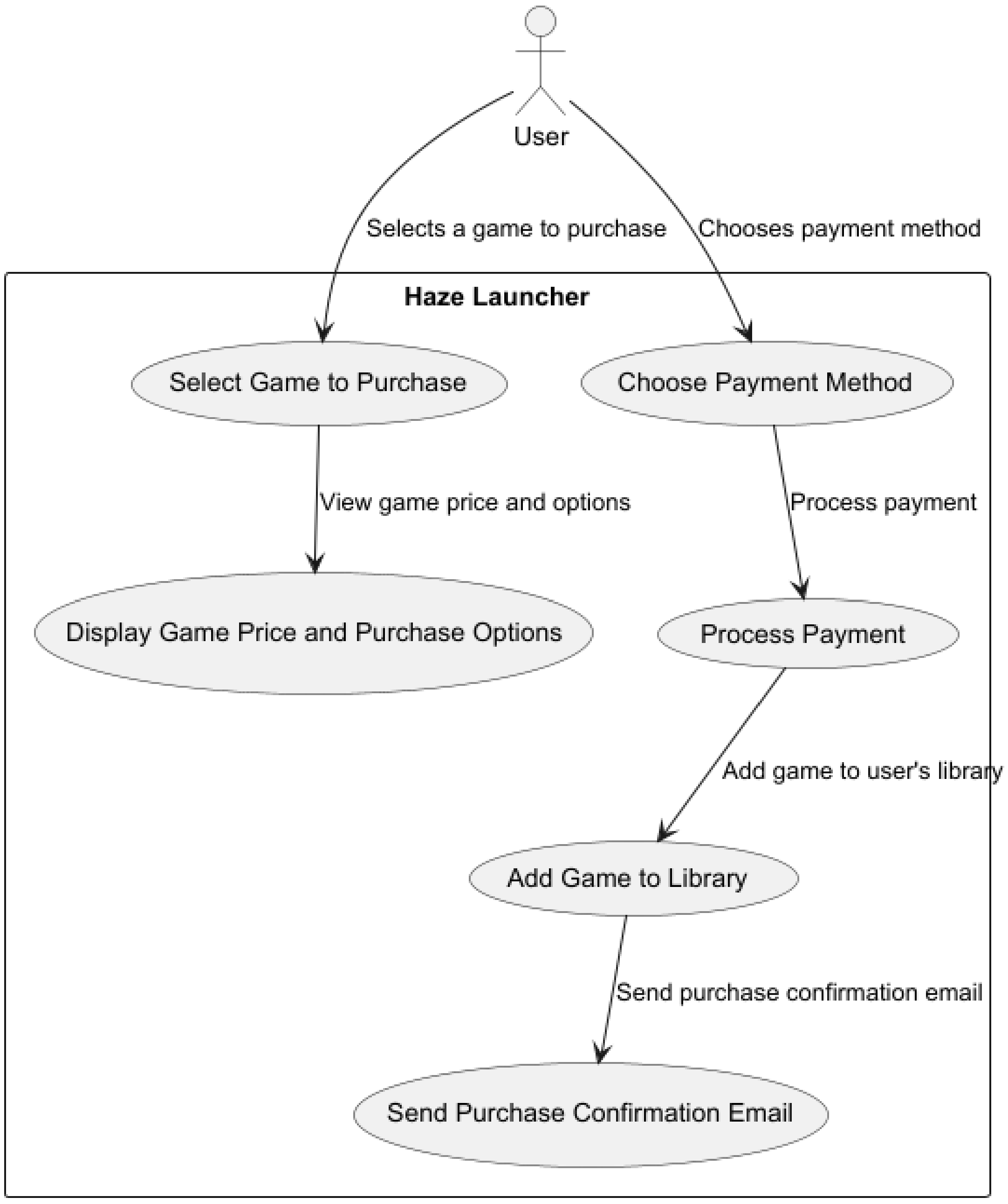
Alternative Flow:

- **A1:** If no games are available, the system displays a message saying "No games available."

---

4. Game Purchase

Diagram:


User
Selects a game to purchase
Chooses payment method
Haze Launcher
Select Game to Purchase
Choose Payment Method
View game price and options
Process payment
Display Game Price and Purchase Options
Process Payment
Add game to user's library
Add Game to Library
Send purchase confirmation email
Send Purchase Confirmation Email

Actor(s): User
Main Flow:

1. The user selects a game to purchase.
2. The system displays the price and purchase options.
3. The user clicks "Buy Now" and selects a payment method.
4. The system processes the payment and adds the game to the library.
5. The user receives a confirmation email.

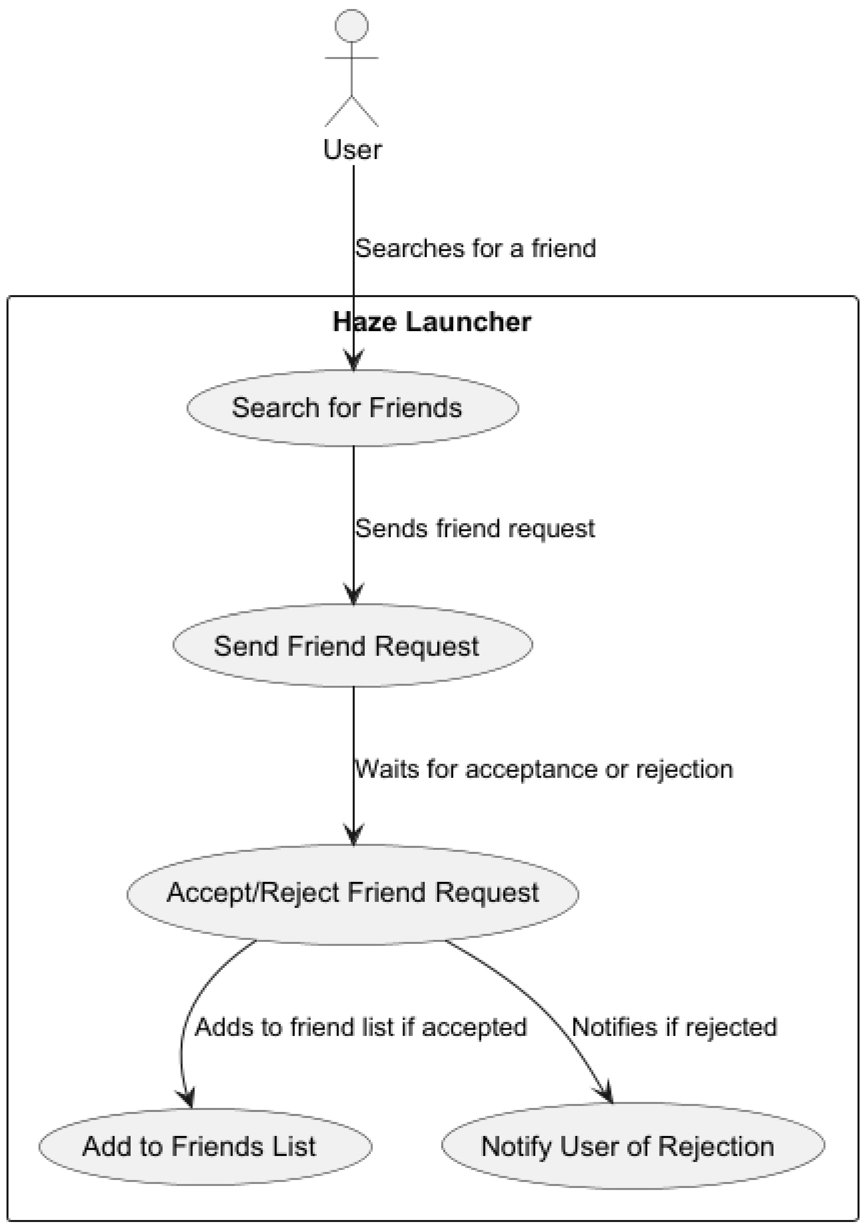
Alternative Flow:

- **A1:** If the payment fails, the system displays an error message and prompts for re-entry.

---

5. Manage Friends

Diagram:



Actor(s): User
Main Flow:

1. The user navigates to the "Friends" section.
2. The user searches for and sends friend requests.
3. The recipient accepts or rejects the request.
4. If accepted, both users are added to each other's friends list.

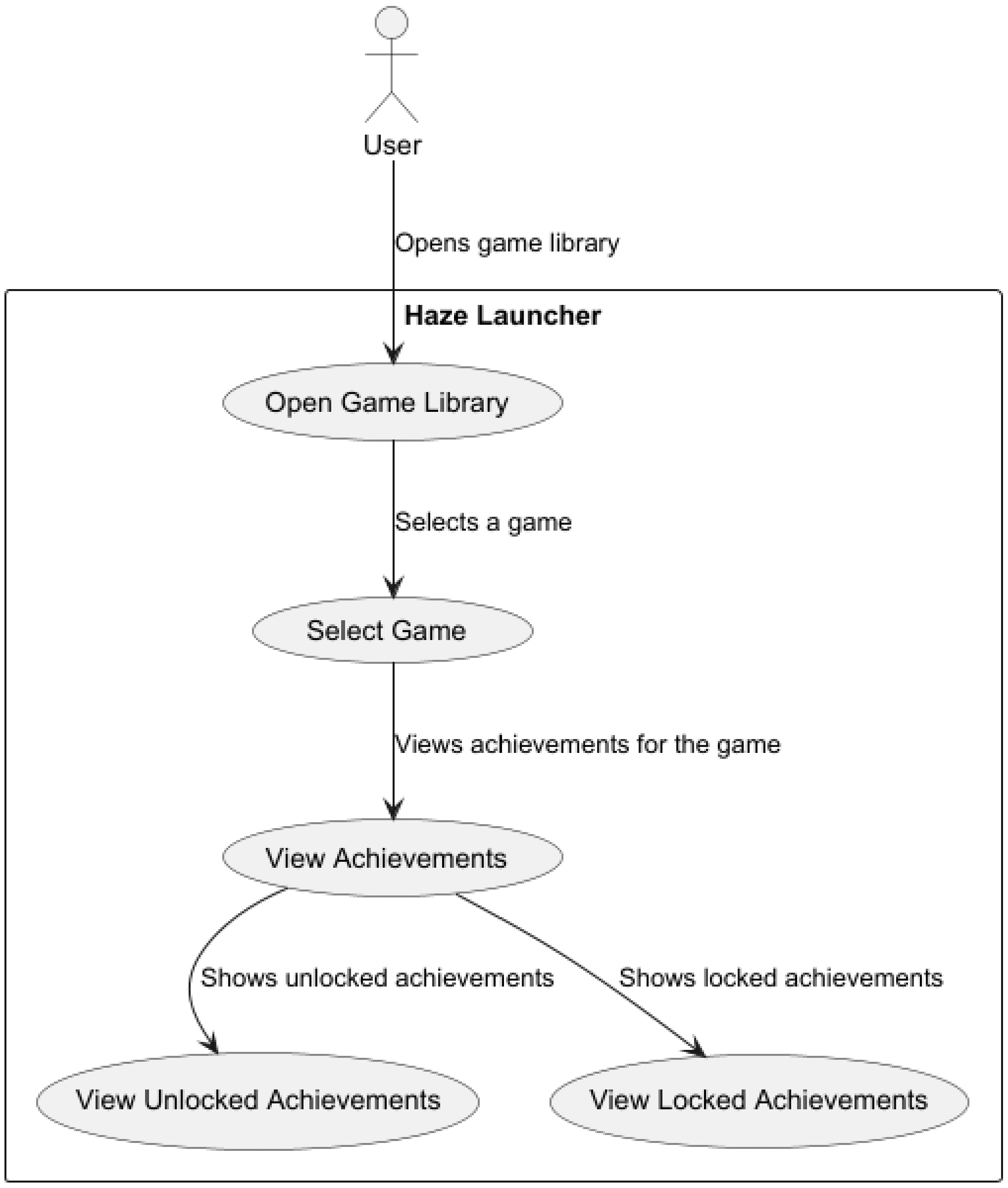
Alternative Flow:

- **A1:** If the user is already friends with the recipient, the system notifies them.

---

6. Track Achievements

Diagram:



Actor(s): User
Main Flow:

1. The user opens their game library.
2. The user selects a game to view its achievements.
3. The system shows the achievements, tracking progress.
4. The user can view unlocked achievements.

Alternative Flow:

- **A1:** If no achievements are unlocked, the system displays a message saying "No achievements unlocked."

---

3. Domain Model

3.1 Key Entities and Relationships:

1. **User**: Represents a registered user of the system.

- Attributes: `userID`, `username`, `email`, `password`, `library`, `friendsList`, `achievements`
- Relationships:
  - A user has a **Library** (1-to-1 relationship).
  - A user has many **Friends** (1-to-many relationship).
  - A user has many **Achievements** (1-to-many relationship).

2. **Game**: Represents a game in the store or the user's library.
   - Attributes: `gameID`, `name`, `description`, `price`, `genre`
   - Relationships:
     - A game can belong to many **Users** (many-to-many relationship) through the **Library**.

3. **Library**: Represents the collection of games a user has purchased or owns.
   - Attributes: `userID`, `games[]` (a collection of Game objects)
   - Relationships:
     - A user has a **Library** (1-to-1 relationship).

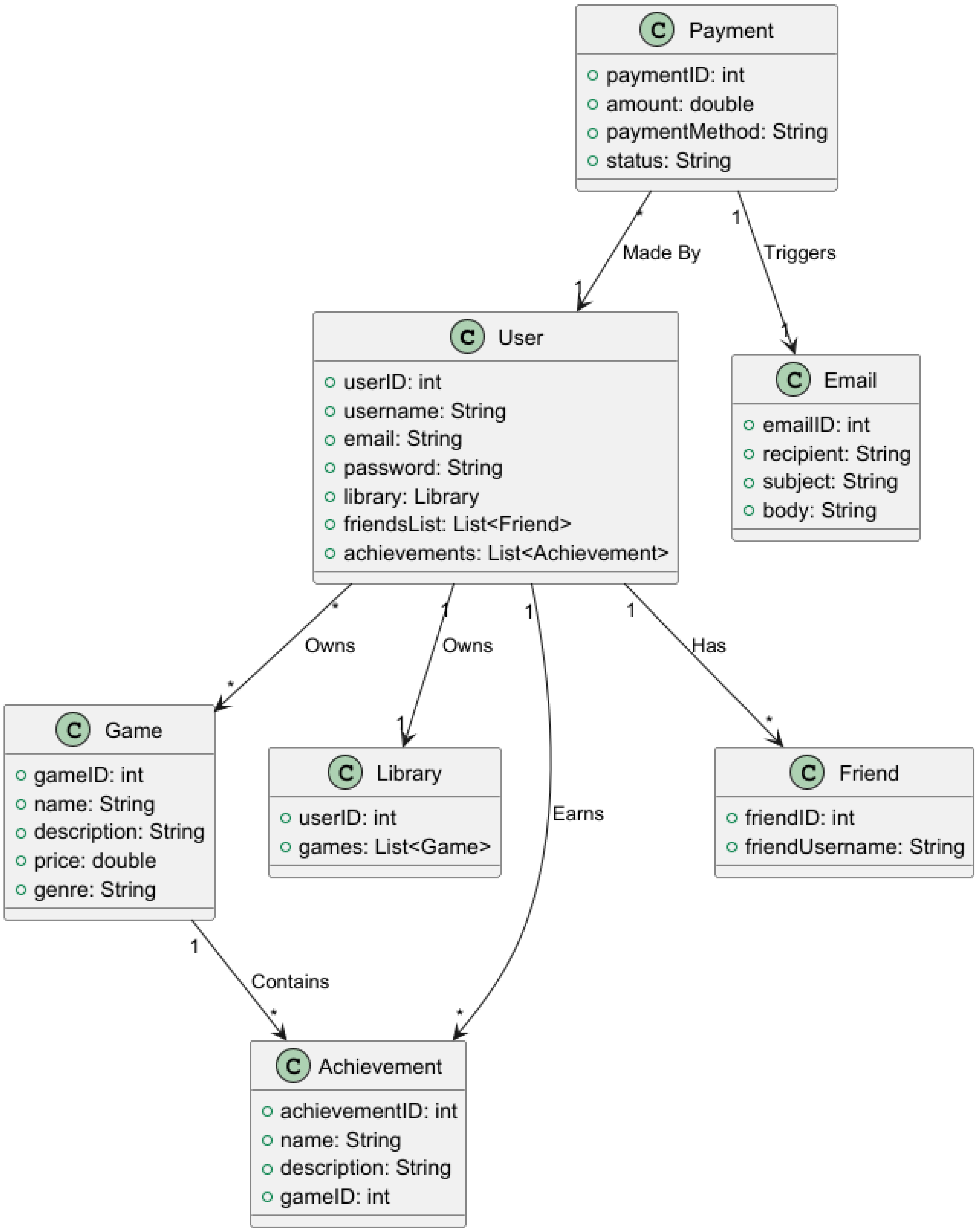
4. **Achievement**: Represents an achievement in a game.
   - Attributes: `achievementID`, `name`, `description`, `gameID`
   - Relationships:
     - An achievement is associated with a **Game** (1-to-many relationship).

5. **Friend**: Represents a user's friend in the system.
   - Attributes: `friendID`, `friendUsername`
   - Relationships:
     - A user has many **Friends** (many-to-many relationship).

6. **Payment**: Represents the payment information for a game purchase.
   - Attributes: `paymentID`, `amount`, `paymentMethod`, `status`
   - Relationships:
     - A user makes many **Payments** (1-to-many relationship).

7. **Email**: Represents the email service used for sending notifications.
   - Attributes: `emailID`, `recipient`, `subject`, `body`
   - Relationships:
     - A **Payment** triggers an **Email** for confirmation (1-to-1 relationship).

3.2 Domain Model Diagram



3.3 Explanation of the Diagram:

- **User**: The user is at the center of the system and has attributes like `userID`, `username`, `email`, and more. They have many relationships, such as owning a **Library**, having **Friends**, and earning **Achievements**.
- **Game**: Games are associated with users via the **Library** and have attributes like `gameID`, `name`, and `price`. A game contains many **Achievements**.
- **Library**: A **Library** is owned by the user and contains a list of **Games**.
- **Achievement**: Achievements are linked to specific **Games** and are earned by users.

- **Friend**: Users can have many **Friends**, and the relationship is many-to-many.
- **Payment**: A user makes payments for games, and each payment triggers an **Email** notification.

---

4. System Sequence Diagrams (SSDs)

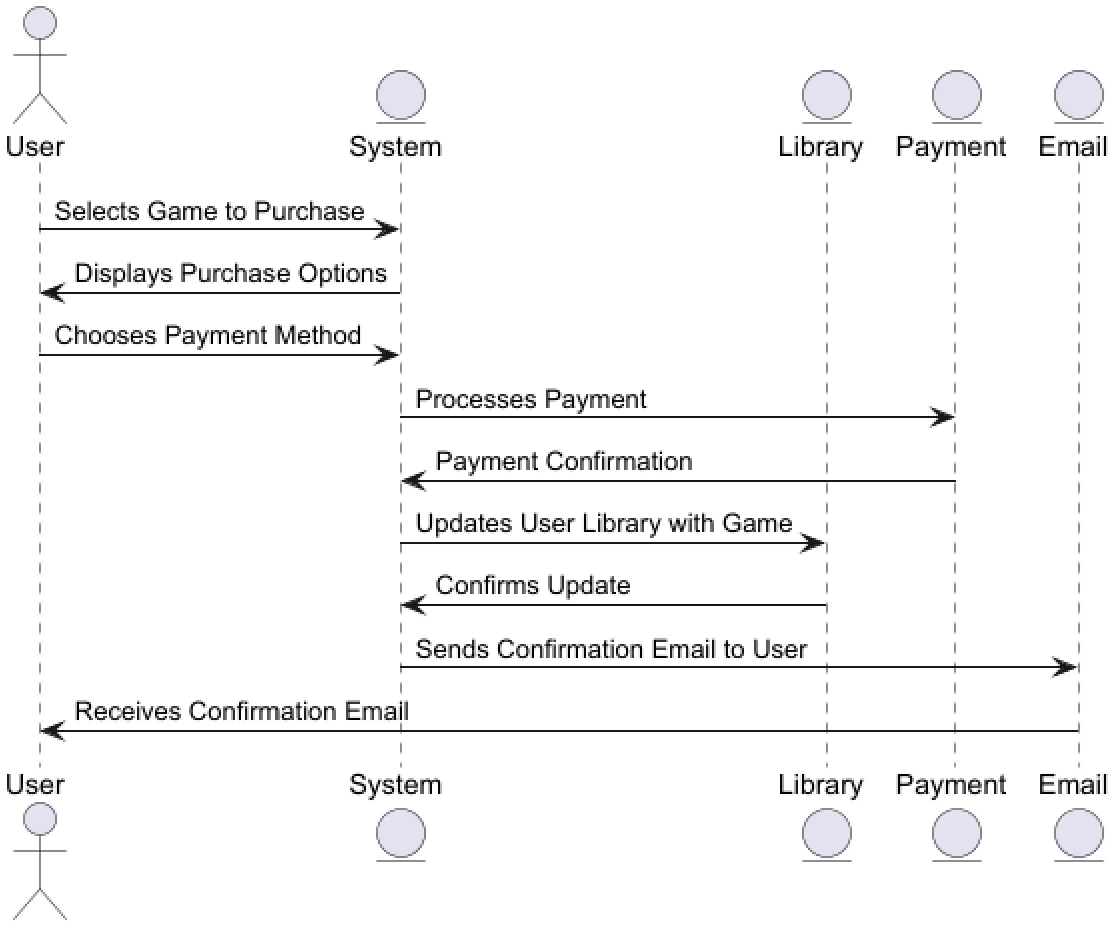
4.1 Game Purchase Sequence

Actors: **User**, **System**

Flow:

1. The **User** selects a **Game** to purchase.
2. The **System** displays purchase options.
3. The **User** selects a payment method (e.g., credit card).
4. The **System** processes the payment via the **Payment** entity.
5. The **System** updates the **User**'s **Library** to include the purchased **Game**.
6. The **System** triggers the **Email** entity to send a purchase confirmation email.

Diagram:



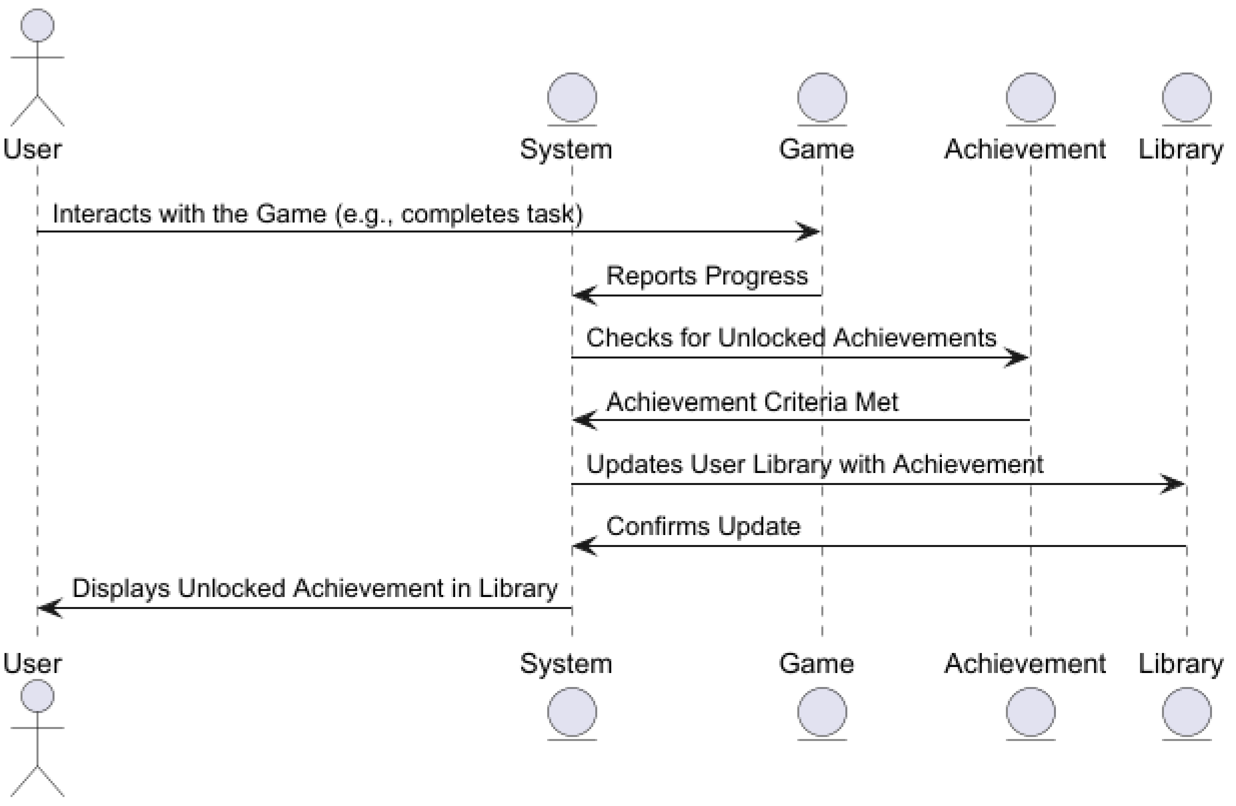
4.2 Unlock Achievement Sequence

Actors: **User**, **System**, **Game**

Flow:

1. **User** interacts with the **Game** (e.g., completes a level, gains points).
2. The **Game** reports the progress back to the **System**.
3. The **System** checks whether the **User** has unlocked any achievements based on the **Game's** progress report.
4. If the **User** meets the achievement criteria, the **System** updates the **Library** with the new achievement.
5. The **User** views the unlocked achievement in the **Library**.

Diagram:


User
System
Game
Achievement
Library
Interacts with the Game (e.g., completes task)
Reports Progress
Checks for Unlocked Achievements
Achievement Criteria Met
Updates User Library with Achievement
Confirms Update
Displays Unlocked Achievement in Library
