# Project Overview Statement

The Game Store project is an online platform which aims to create a digital platform similar to Steam or Xbox Game Center, designed to enable users to browse / search / subscript / like(collect) / buy the video games. We Name it as GameOne Store.

This platform caters to a wide range of users including gamers, developers, and administrators. The goal is to provide a seamless, user-friendly experience where users can browse games, make purchases, and leave reviews. At the same time, we’d like to enrich the user experience on our platform and enhance the user stickiness of users. Such as post / comment / share game experiences or insights.

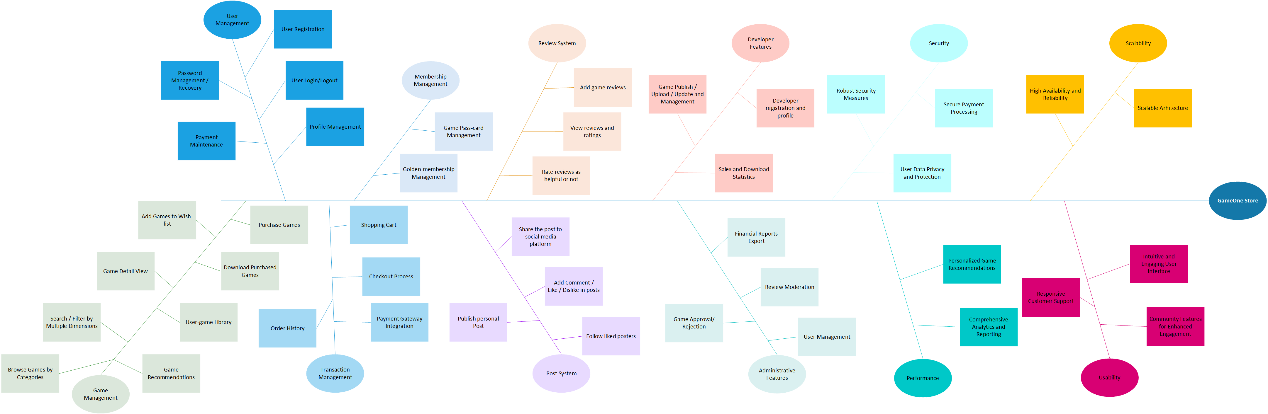
For Game Developers, they can upload and manage their games, while administrators oversee platform activities to ensure smooth operations. The objective is to offer a seamless and engaging user experience for gamers while ensuring robust backend support for developers / publishers to manage their game listings.

# Requirements/Feature List

|  |  |
| --- | --- |
| Functional Requirements | |
| User Management | User Registration |
| User Login/Logout |
| Profile Management |
| Password Management / Recovery |
| Payment Maintenance |
| Game Management | Browse Games by Categories |
| Search / Filter by Multiple Dimensions(category, title, genre, developer, etc) |
| Game Detail View (View detailed information about games, including trailers, screenshots, and descriptions) |
| Add Games to Wish-list |
| Purchase Games |
| Download Purchased Games |
| User-game Library |
| Game Recommendations (Personalized game recommendations based on user preferences and purchase history) |
| Membership Management | Game Pass-card Management |
| Golden-membership Management |
| Transaction Management | Shopping Cart |
| Checkout Process |
| Payment Gateway Integration |
| Order History |
| Review System | Add game reviews |
| View reviews and ratings |
| Rate reviews as helpful or not |
| Post System | Publish personal Post about Games/Review/Experiences |
| Follow liked posters |
| Add Comment / Like / Dislike in posts |
| Share the post to social media platform |
| Developer Features | Developer registration and profile |
| Game Publish / Upload / Update and Management |
| Sales and Download Statistics |
| Administrative Features | User Management |
| Game Approval/Rejection |
| Review Moderation |
| Financial Reports Export |
| Additional Features | Notifications (email, in-app) |
| Promotional offers and discounts |
| Multilingual support |

|  |  |
| --- | --- |
| Non-functional Requirements | |
| Security | Robust Security Measures |
| Secure Payment Processing |
| User Data Privacy and Protection |
| Scalability | Scalable Arhitecture |
| High Availability and Reliability |
| Performance | Personalized Game Recommendations |
| Comprehensive Analytics and Reporting |
| Usability | Responsive Customer Support |
| Intuitive and Engaging User Interface |
| Community Features for Enhanced Engagement |

# Feature Tree Diagram



# Domain Dictionary

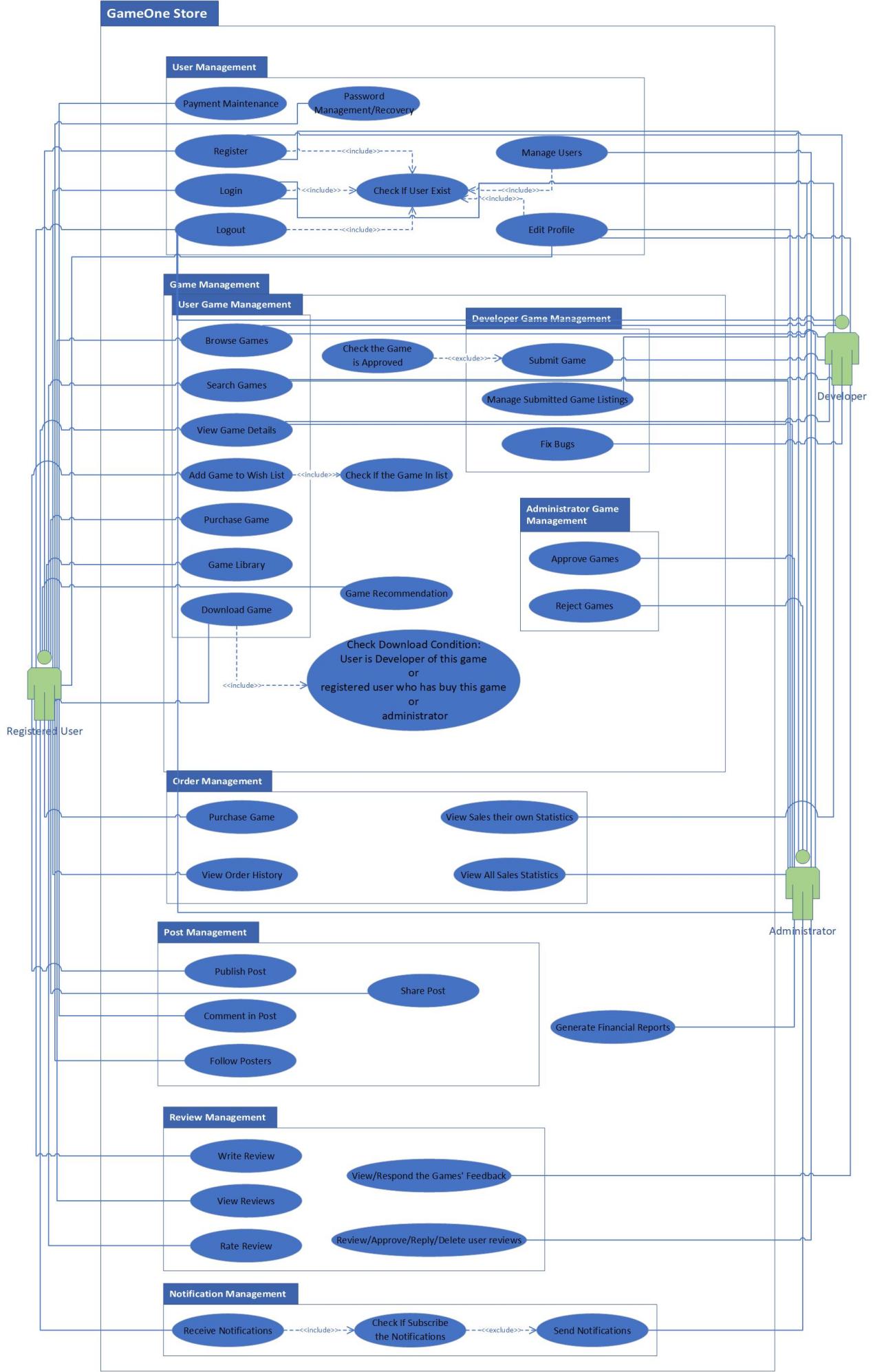
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| User | Role | An individual who uses the Game Store platform. |
| Developer | Role | A user who submits and manages games on the platform. |
| Administrator | Role | A user with elevated privileges for managing the platform. |
| Game | Object | A digital product available for purchase or download. |
| Review | Process | Feedback provided by users about a game. |
| Browse / Search | Process | The process of searching a game |
| Subscript / Like(Collect) | Process | Th process of Subscribe / Like(collect) a game, after this operation it will be added to Wish List |
| Transaction | Process | The process of purchasing a game. |
| Post | Process | The Process of posting a post about games, including the experience / rates. |
| Wish List | Object | A list of games a user is interested in purchasing later. |
| Notification | Object | Messages sent to users about updates, offers, or other information. |
| Shopping Cart | Object | A virtual cart where users can add games before purchasing. |
| Payment Gateway | Process | A service that processes credit card payments for transactions. |
| Library | Object | The collection of games owned by a user |

# Actor Dictionary

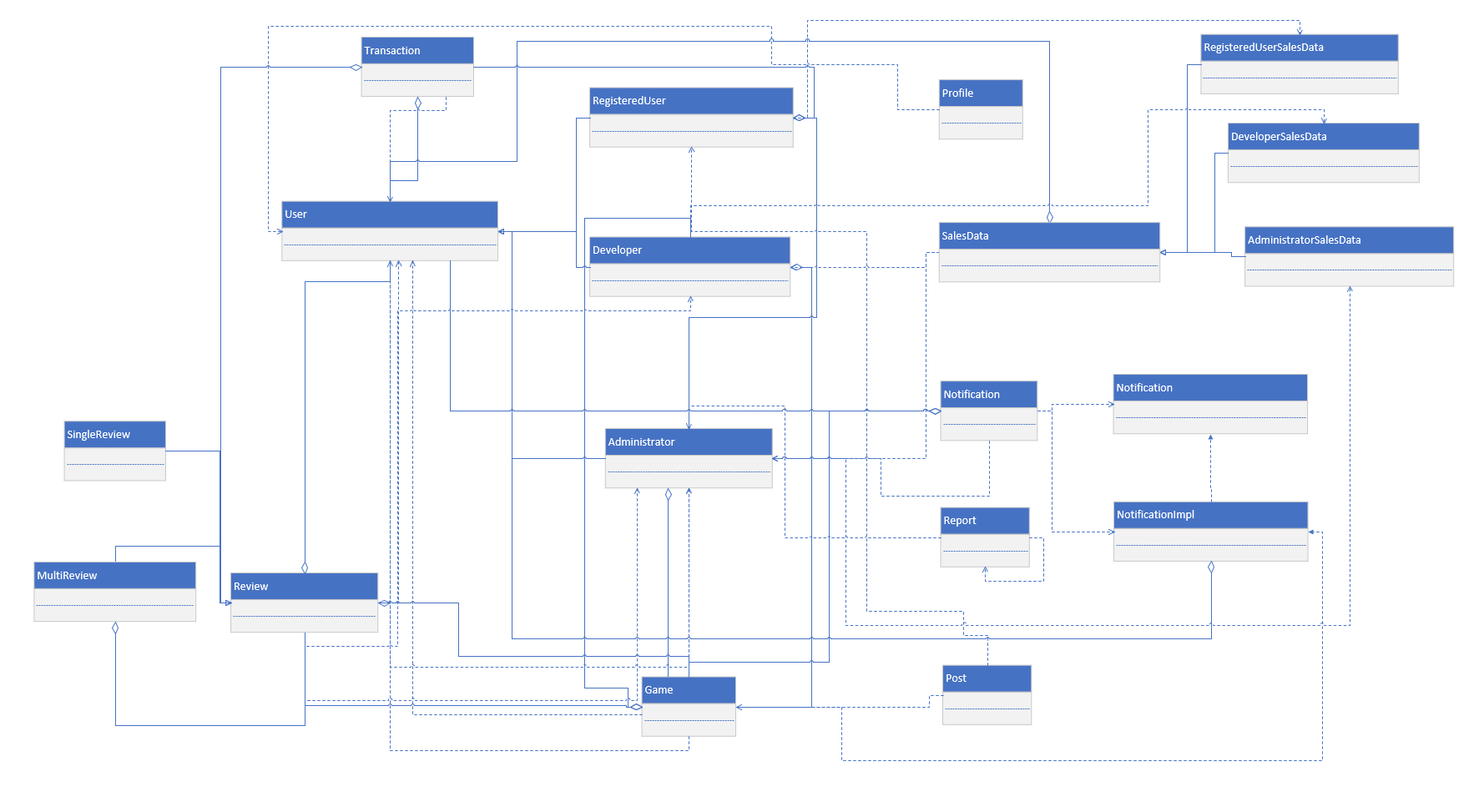
|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | **Description** | **Abstraction** | **Use case(s)** |
| User | Any visitor to the Game Store who can browse and purchase games. | Yes | 1. Register 2. Login 3. Logout 4. Browse Games 5. Search Games 6. View Game Details |
| Registered User | A user who has created an account and can access additional features like the wish list and review system. | No(inherit from the User) | 1. Edit Profile 2. Add Game to Wish List 3. Check If the Game in List 4. Purchase Game 5. Download Game 6. Check Download Condition 7. Get Game Library 8. View Order History 9. Write Review 10. View Review 11. Rate Review 12. Recommend Game 13. Receive Notifications 14. Subscribe/Follow Game 15. Publish Post 16. Share Game 17. Share Post 18. Comment Post 19. Follow Posters 20. Maintain payment 21. Manage/Recover password 22. Check Subscribe |
| Developer | A registered user who submits and manages their games. | No(inherit from the User) | 1. Submit Game 2. Manage Submitted Game Listings 3. View Sales Statistics from themselves 4. View/Respond the Games’ Feedback 5. Fix Bugs |
| Administrator | A registered user with permissions to manage the platform, including user management and content moderation. | No(inherit from the User) | 1. Manage Users 2. Approve Games 3. Reject Games 4. Review/Approve/Reply/Delete user reviews 5. View All Sales Statistics 6. Generate Financial Reports 7. Send Notifications |

# Use Case Diagram

1. Register
2. Login
3. Logout
4. Browse Games
5. Search Games
6. View Game Details
7. Edit Profile
8. Add Game to Wish List
9. Check If the Game in List
10. Purchase Game
11. Download Game
12. Check Download Condition
13. Get Game Library
14. View Order History
15. Write Review
16. View Review
17. Rate Review
18. Recommend Game
19. Receive Notifications
20. Subscribe/Follow Game
21. Publish Post
22. Share Game
23. Share Post
24. Comment Post
25. Follow Posters
26. Maintain payment
27. Manage/Recover password
28. Check Subscribe
29. Submit Game
30. Manage Submitted Game Listings
31. View Sales Statistics from themselves
32. View/Respond the Games’ Feedback
33. Fix Bugs
34. Manage Users
35. Approve Games
36. Reject Games
37. Review/Approve/Reply/Delete user reviews
38. View All Sales Statistics
39. Generate Financial Reports
40. Send Notifications

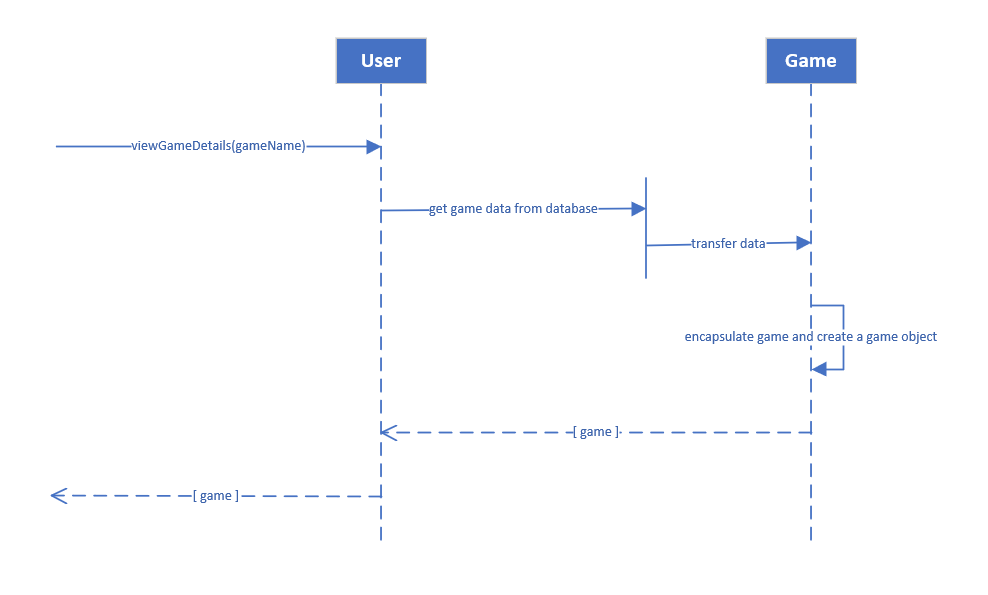


# Analysis Model

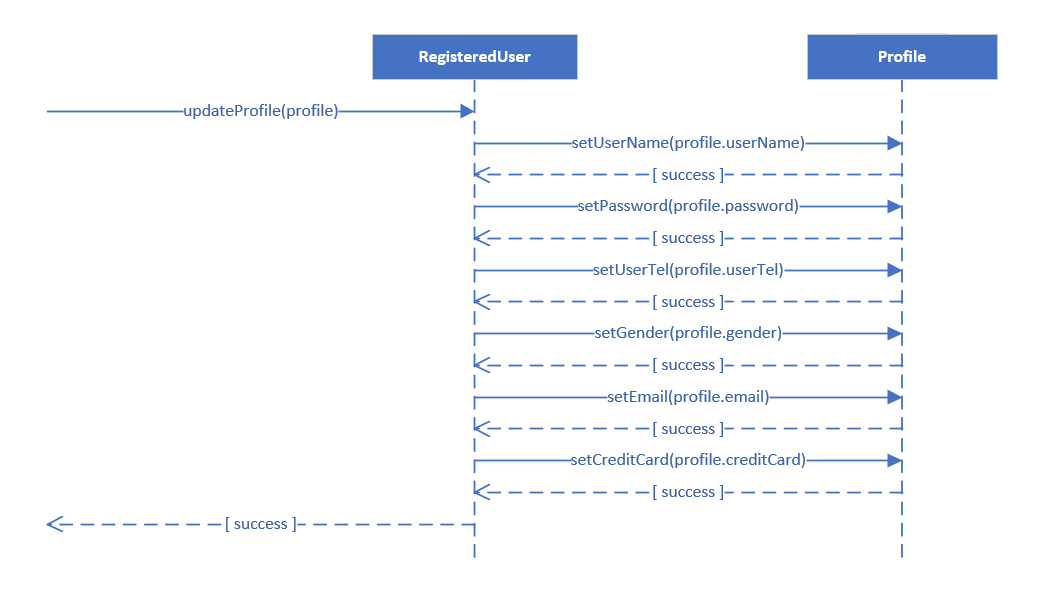


# Interaction Sequence Diagrams

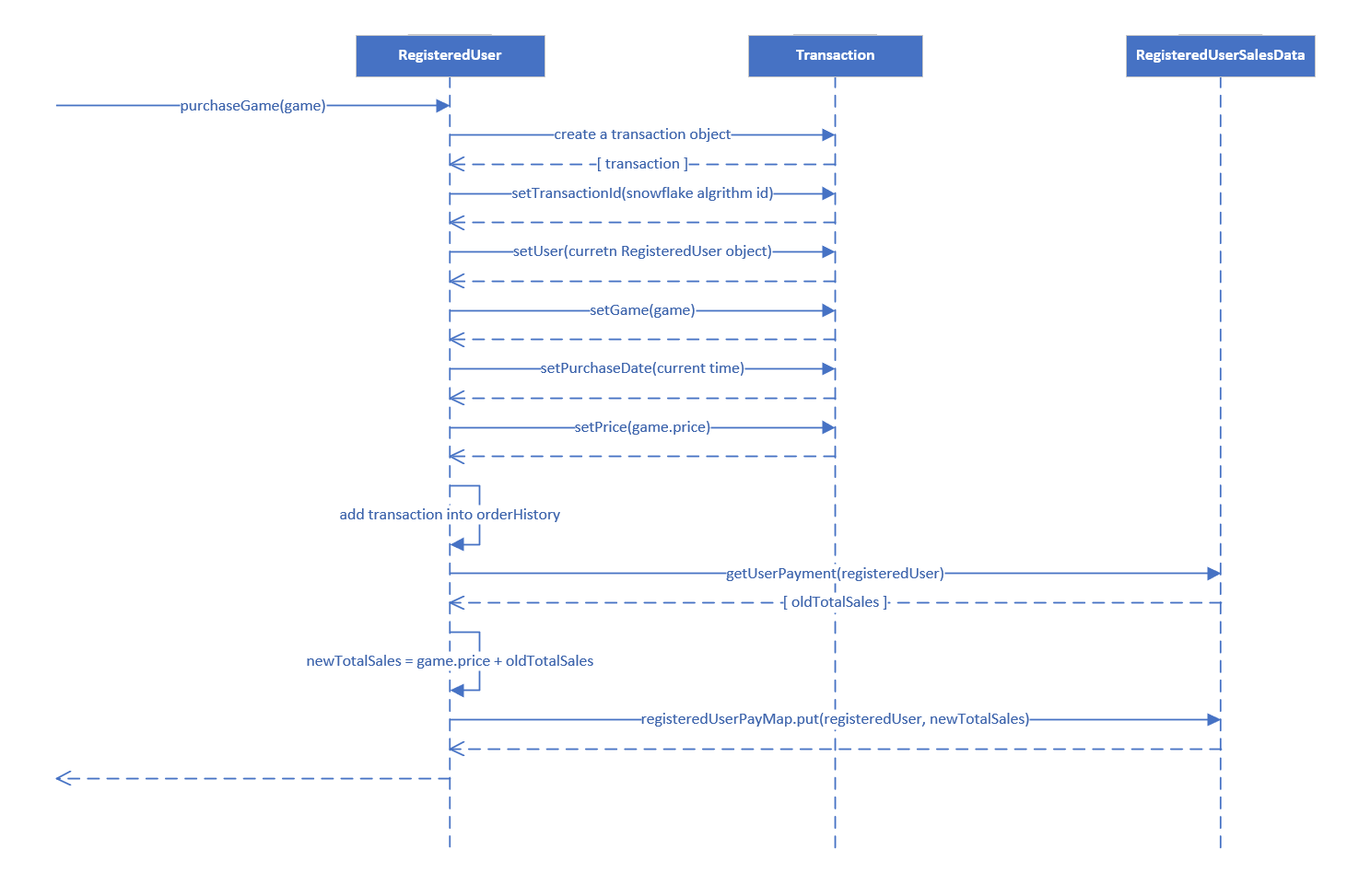
1. viewGameDetails(gameName): “gameName” is the name of the game that the User wants to view in detail.



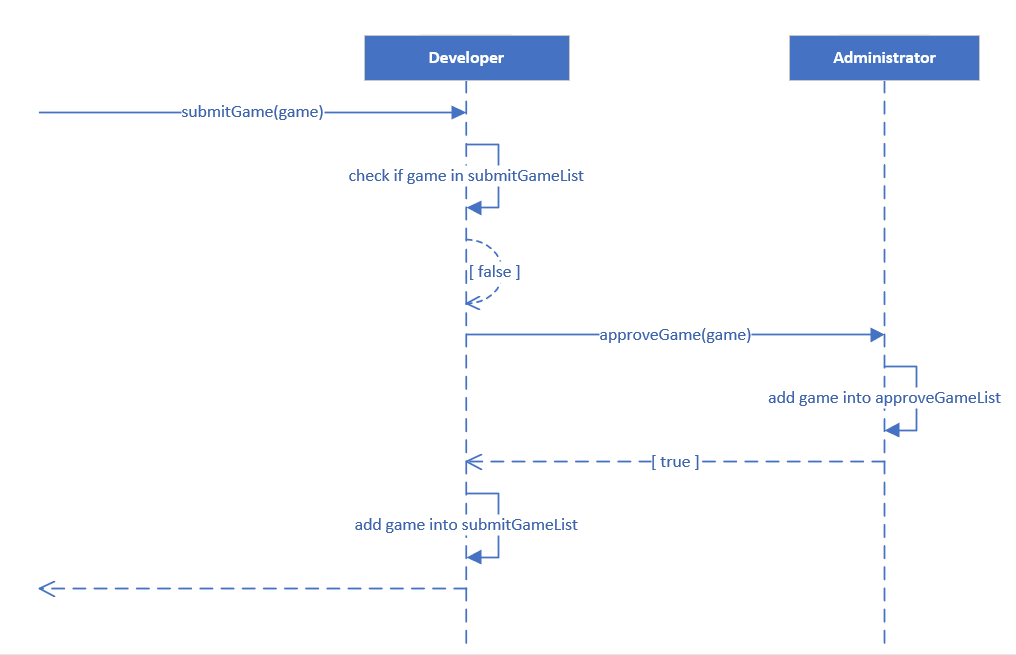
1. updateProfile(profile): “profile” contains the basic information for the RegisteredUser that needs to be updated, such as userName, userTel, creditCard and so on.



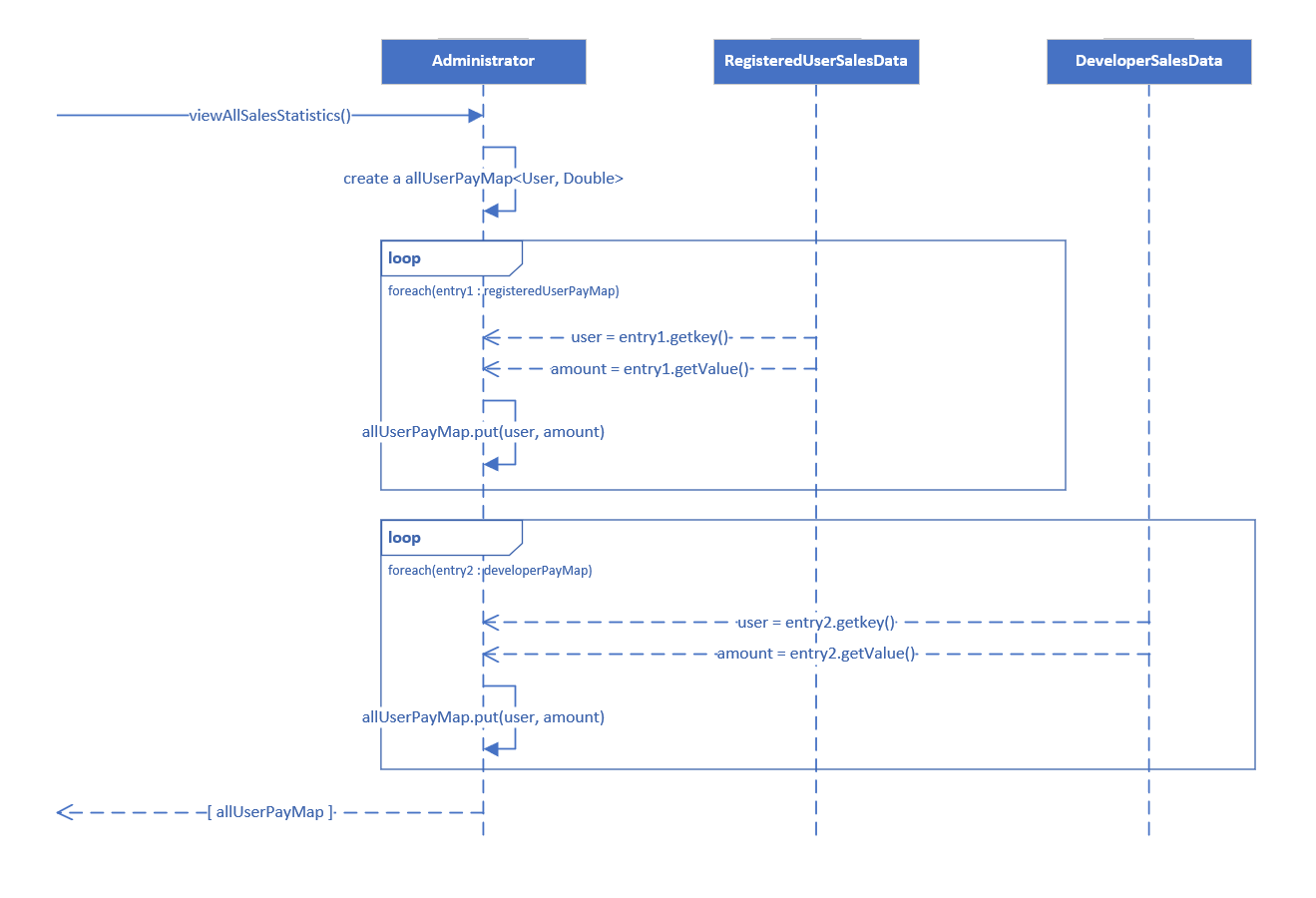
1. purchaseGame(game): “game” is a game object that RegisteredUser wants to buy. (Here, RegisteredUser is an example. The same applies to Developer and Administrator.)



1. submitGame(game): “game” is a game object that Developer wants to submit.

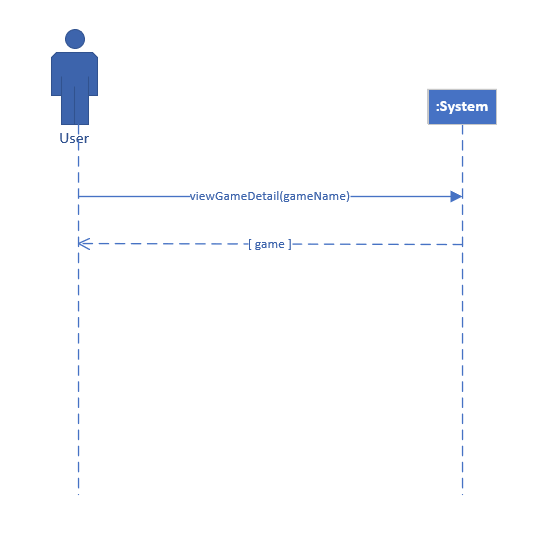


1. viewAllSalesStatistics(): Get the SalesData mapping for each Game.

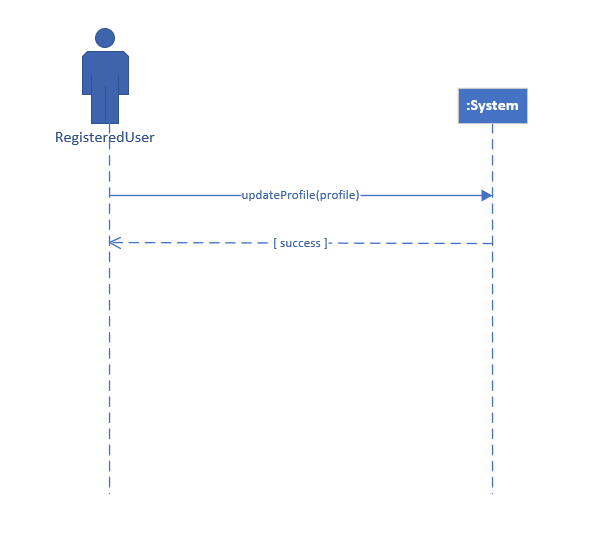


# System Sequence Diagrams

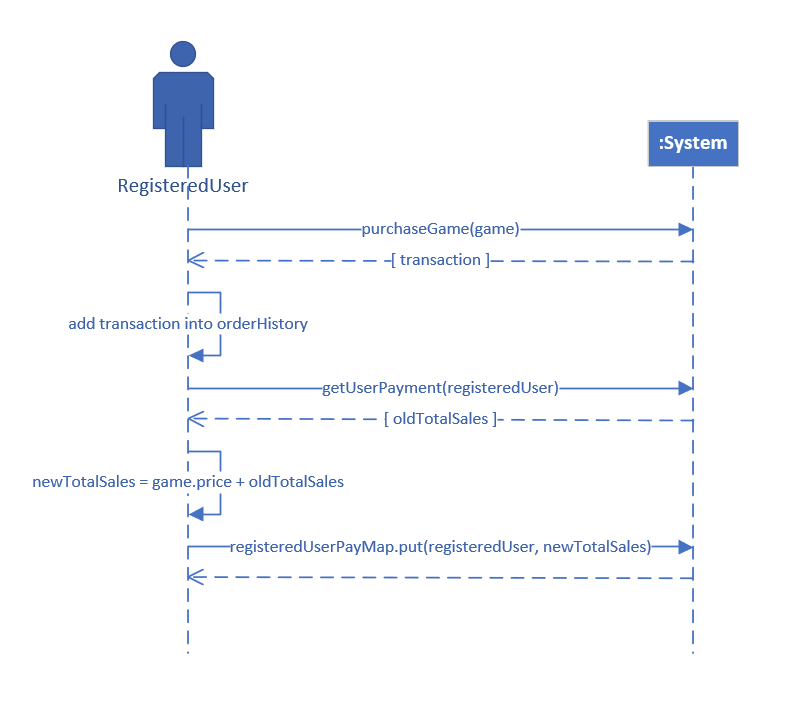
1. viewGameDetails(gameName)



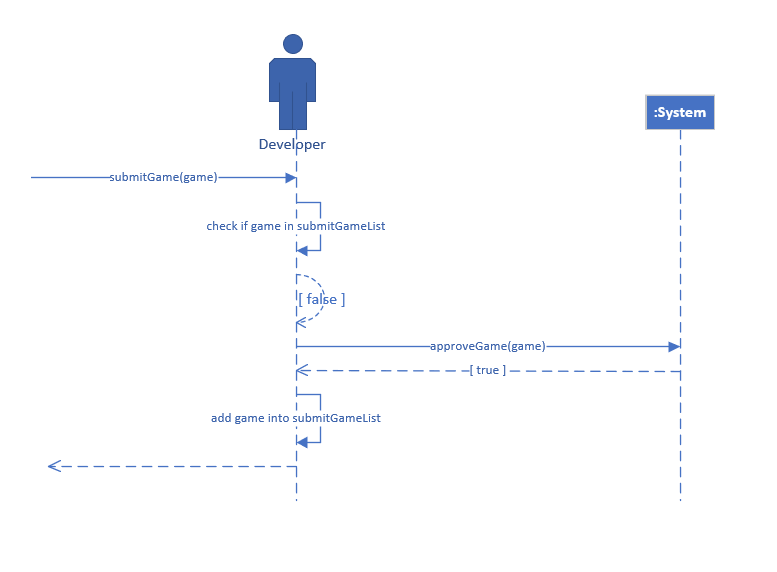
1. updateProfile(profile)



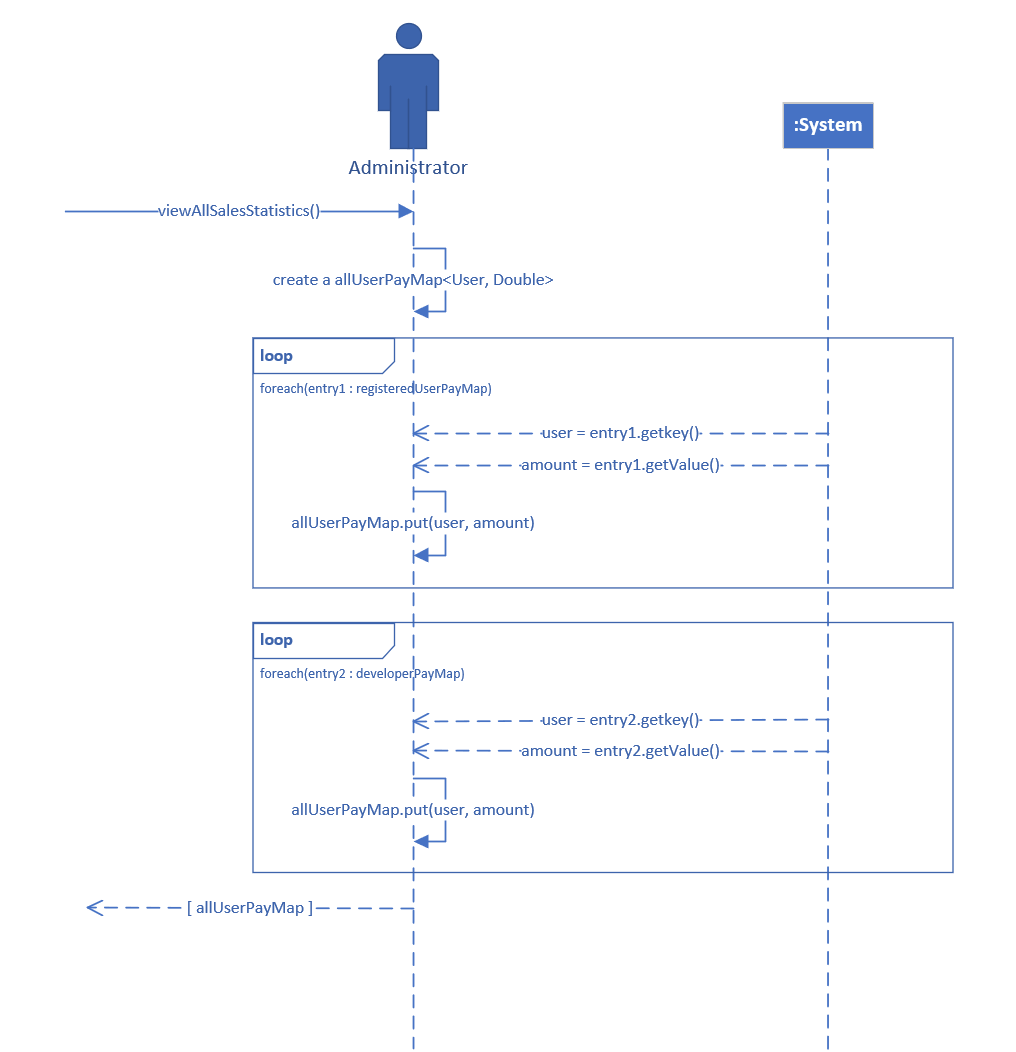
1. purchaseGame(game)



1. submitGame(game)

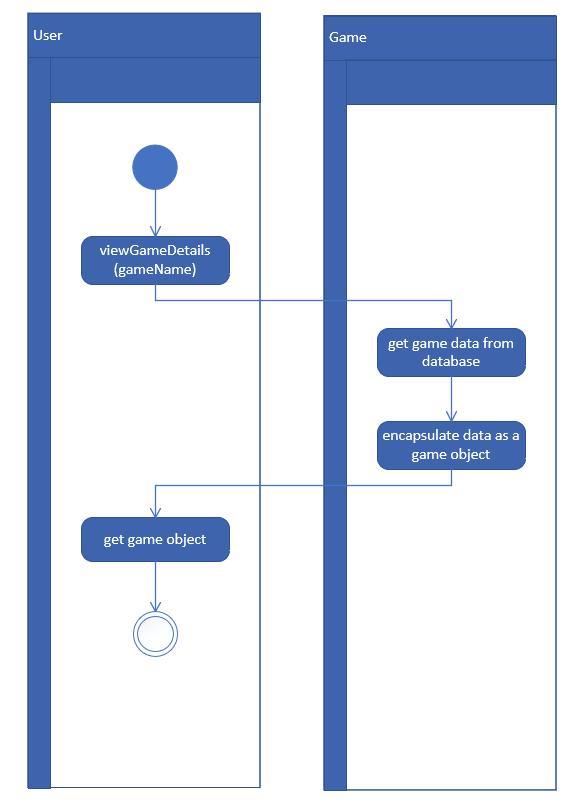


1. viewAllSalesStatistics()

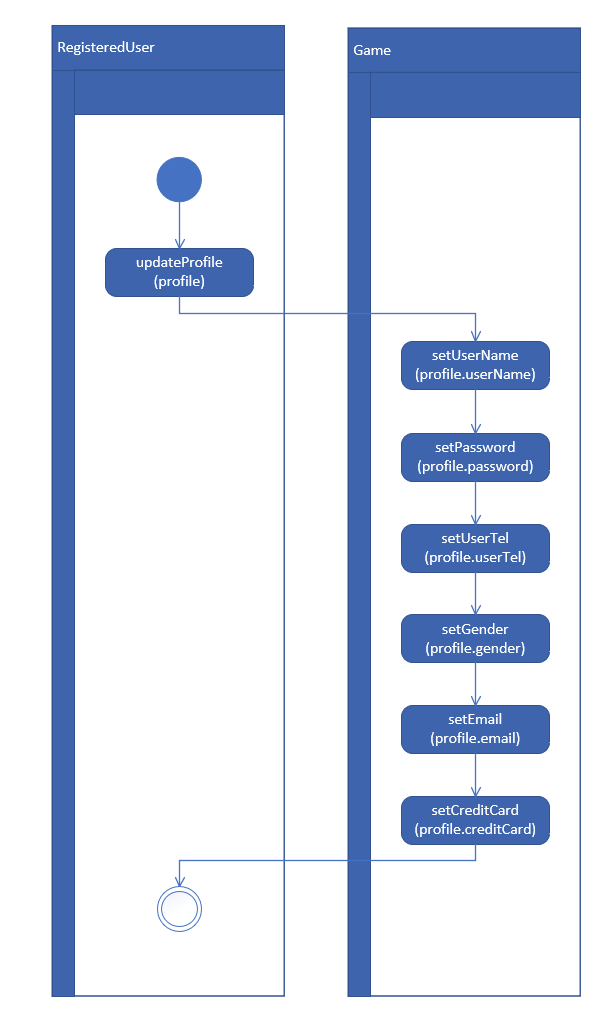


# Activity Diagrams

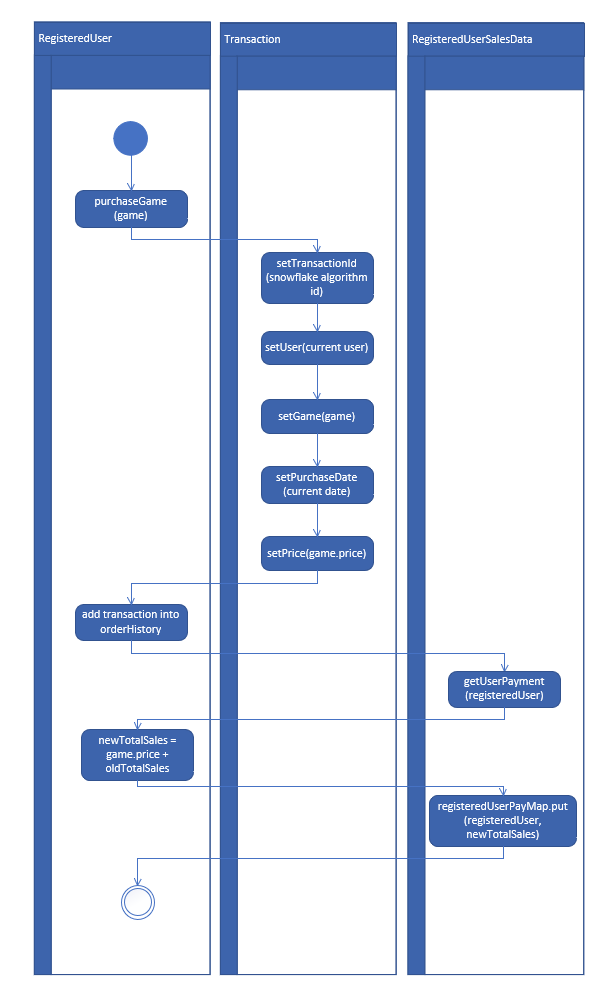
1. viewGameDetails(gameName)



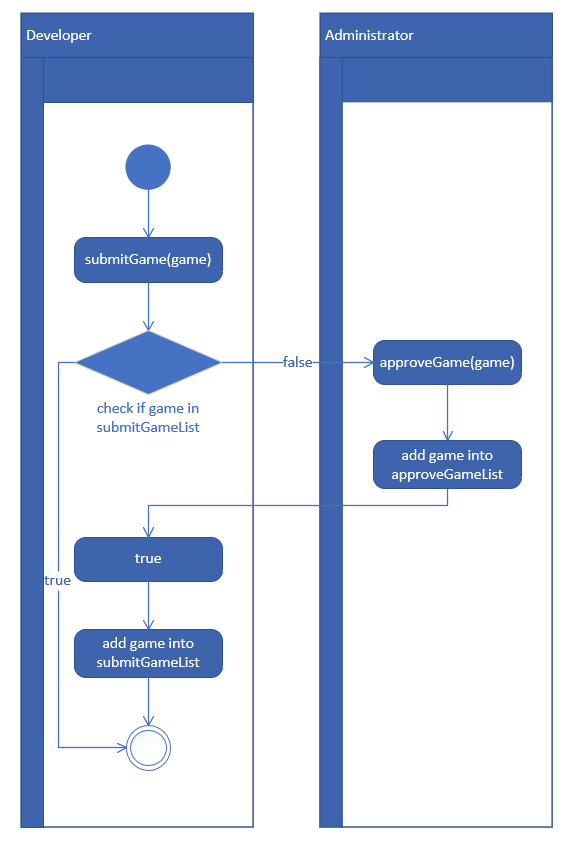
1. updateProfile(profile)



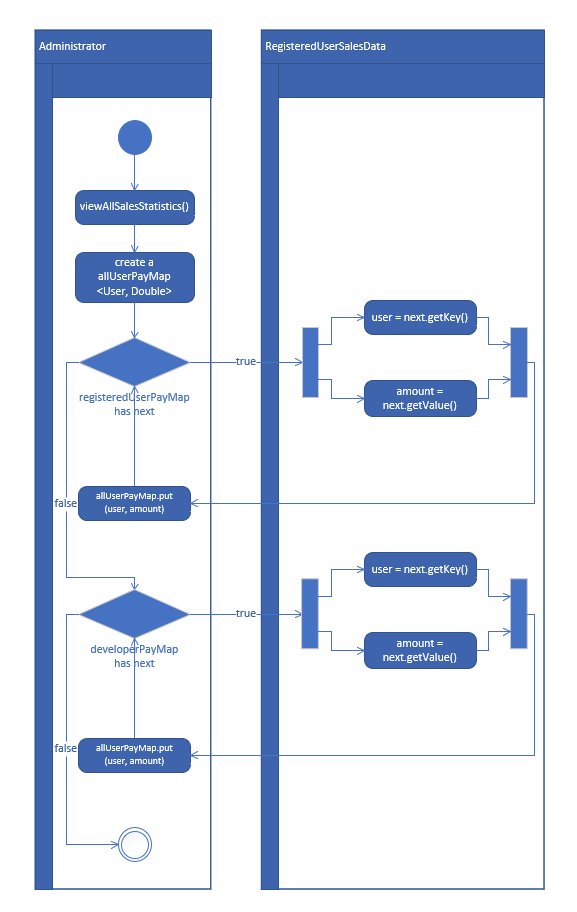
1. purchaseGame(game)



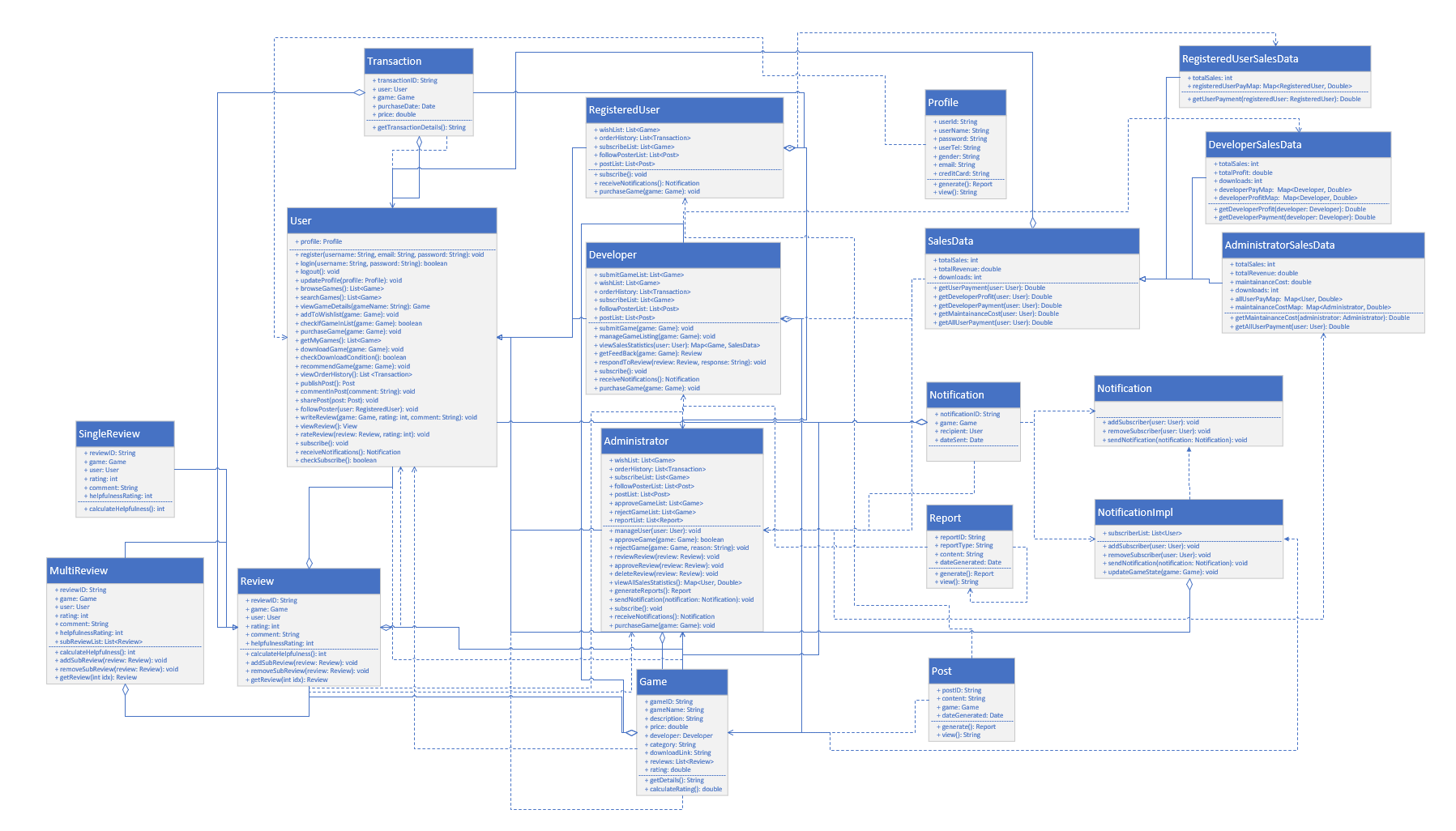
1. submitGame(game)



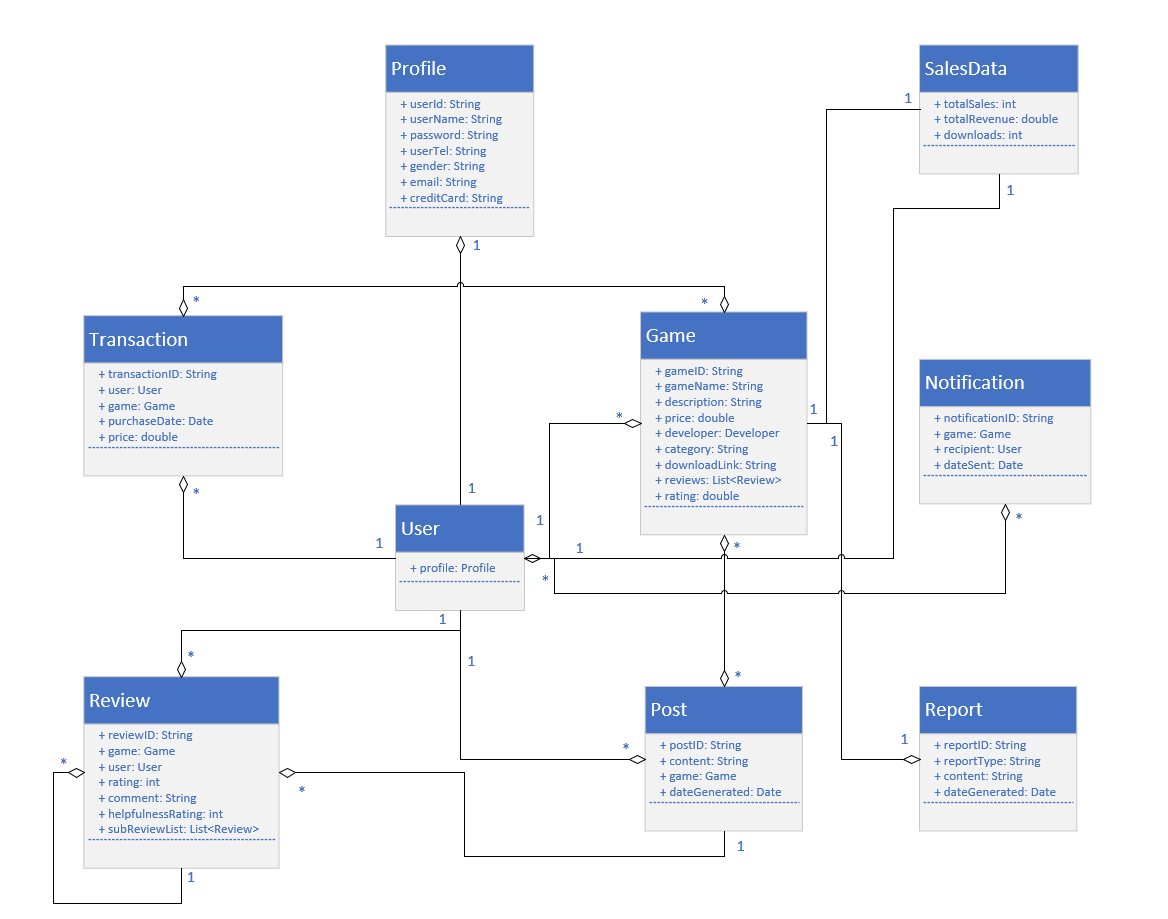
1. viewAllSalesStatistics()



# Design Model

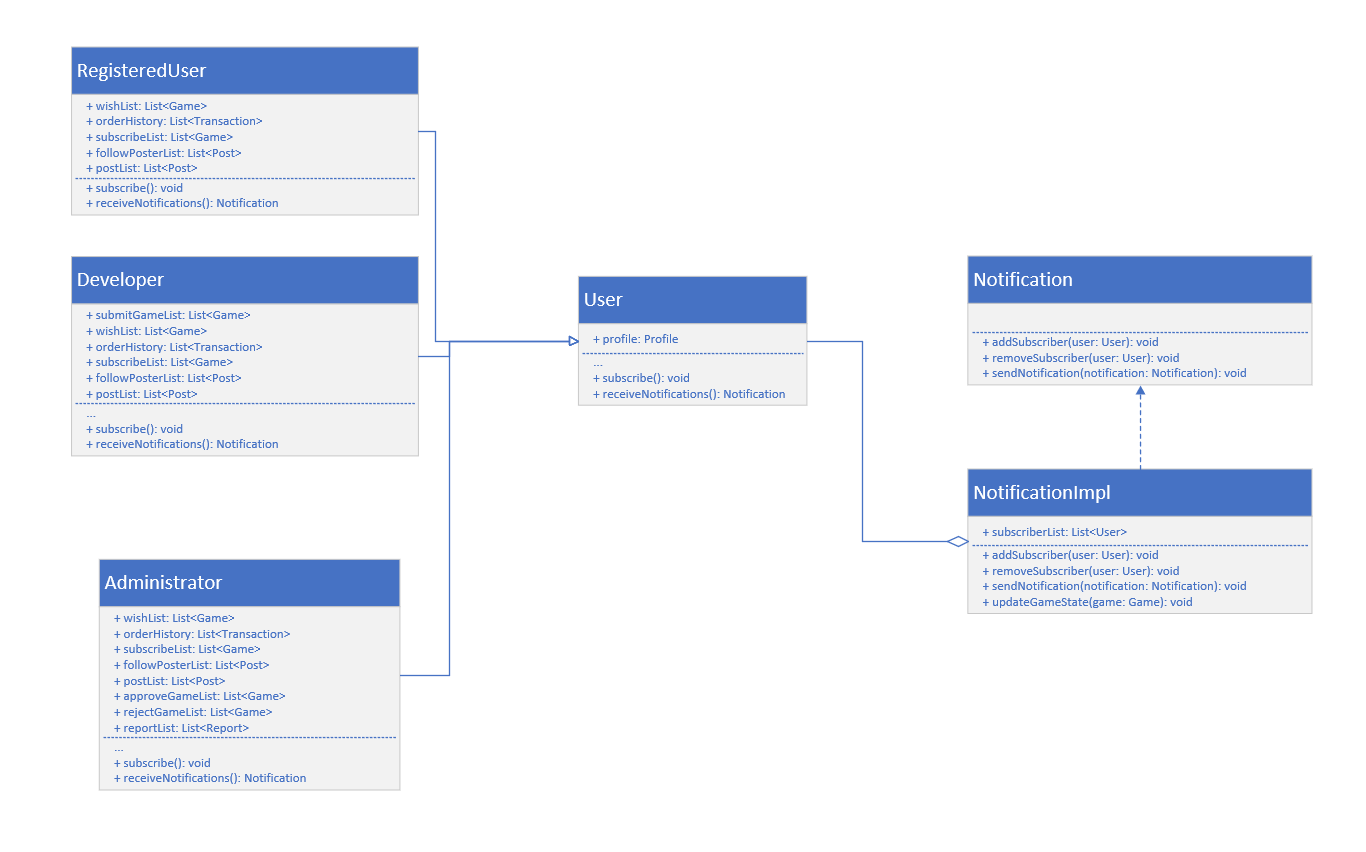


# Object-Relational Model



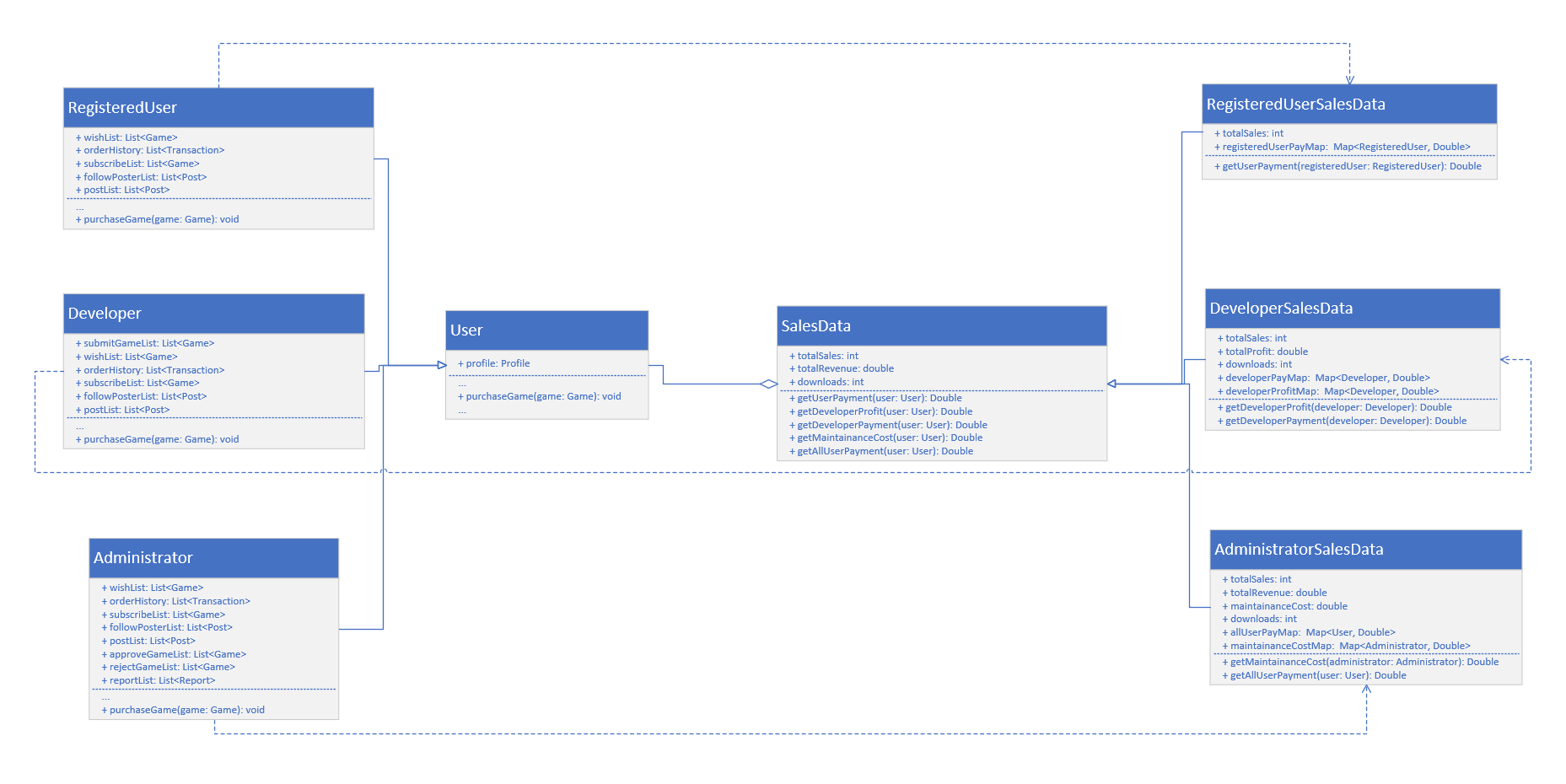
# Documentation and class diagrams for Design Patterns used

1. **Observer Design Pattern**



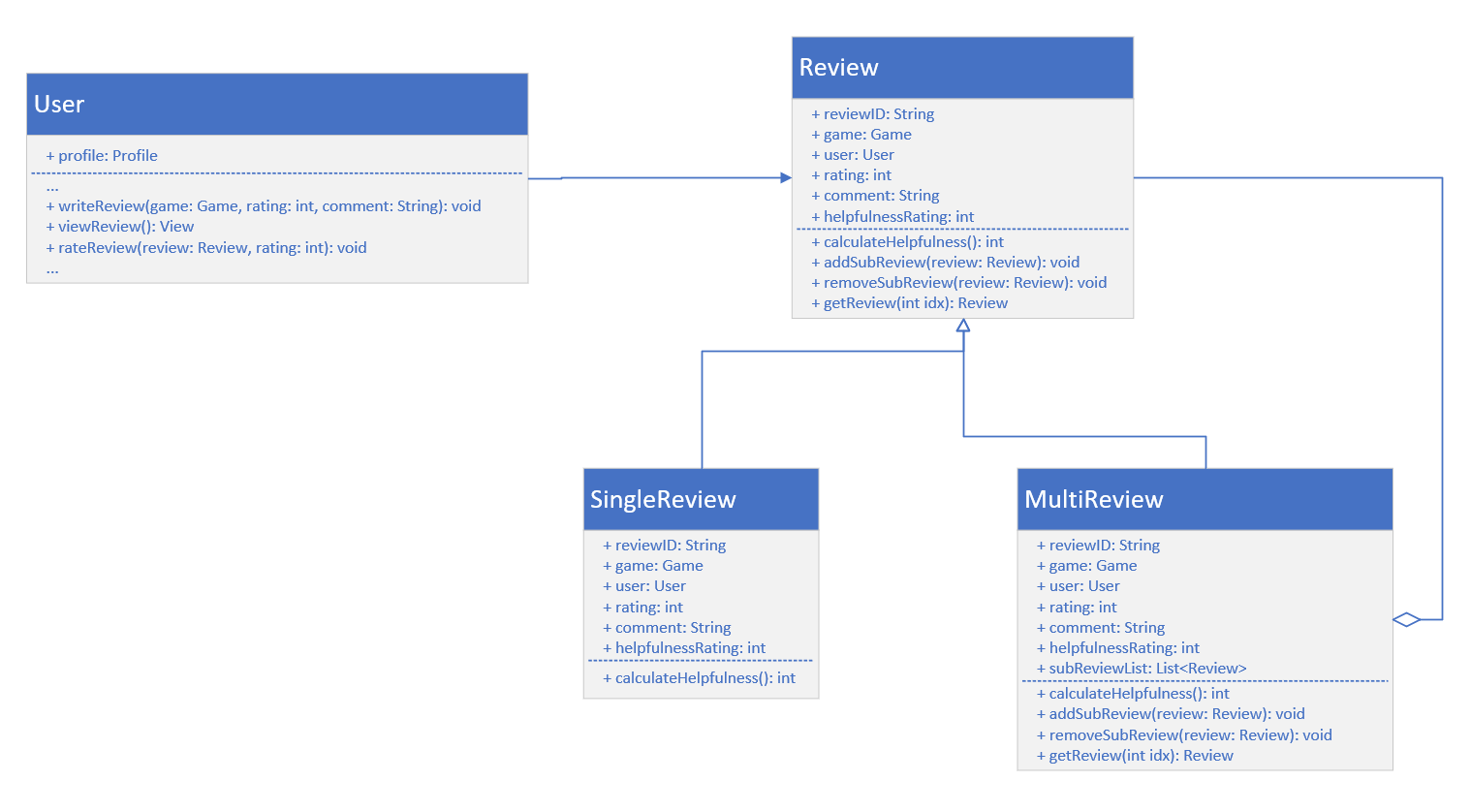
Each time a user subscribes to the Game Update Service, the user is registered to the **NotificationImpl**, which is stored in the subscriberList. When the game is updated, the new game is pushed to every user who subscribes to the game, including regular users, developers, and administrators.

1. **Factory Method Design Pattern**



**User** can be seen as an abstract factory and its sub-classes as concrete factories. **SalesData** can be viewed as an abstract product and its sub-classes as concrete products. When a concrete user is created and a game is purchased, the purchaseGame() method implemented by the concrete user is invoked, where a concrete SalesData subclass is created, which can then perform different logical processing such as the amount of merchandise sold or maintenance costs, and so on.

1. **Composite Design Pattern**



Users can create a single comment, while other users can attach their own comments to the comment. This is a self-associative process that is typical of the Composite design pattern.