Game Shopping System

Đặng Đăng Khôi - ITCSIU22266

Lê Hoài Bảo - ITCSIU22259

Võ Hoàng Hiệp- ITCSIU22261

Overview

1: Introduction

2. Methodology

3. Implementation

4. Conclusion and future work

1. Introduction

Background

Market Shift: Rapid growth in digital game distribution and the transition to online-only platforms.

E-Commerce Essentials: High demand for seamless browsing, personalizations, and digital library management.

1. Introduction

Problem Statement

Missing Features: Lack of personalized recommendations and integrated wishlist/library management.

User References Connection: Lack of seeking for user references to provide suitable games in terms of platform or hardware.

1. Introduction

Objective and Scopes

Full stack Development:

- Database: PostgreSQL
- Frontend: ReactJS
- Backend: ExpressJS/NodeJS

Responsive Design: Optimized for both desktop and mobile devices.

Data Integrity: Optimized database performance and standardized error handling.

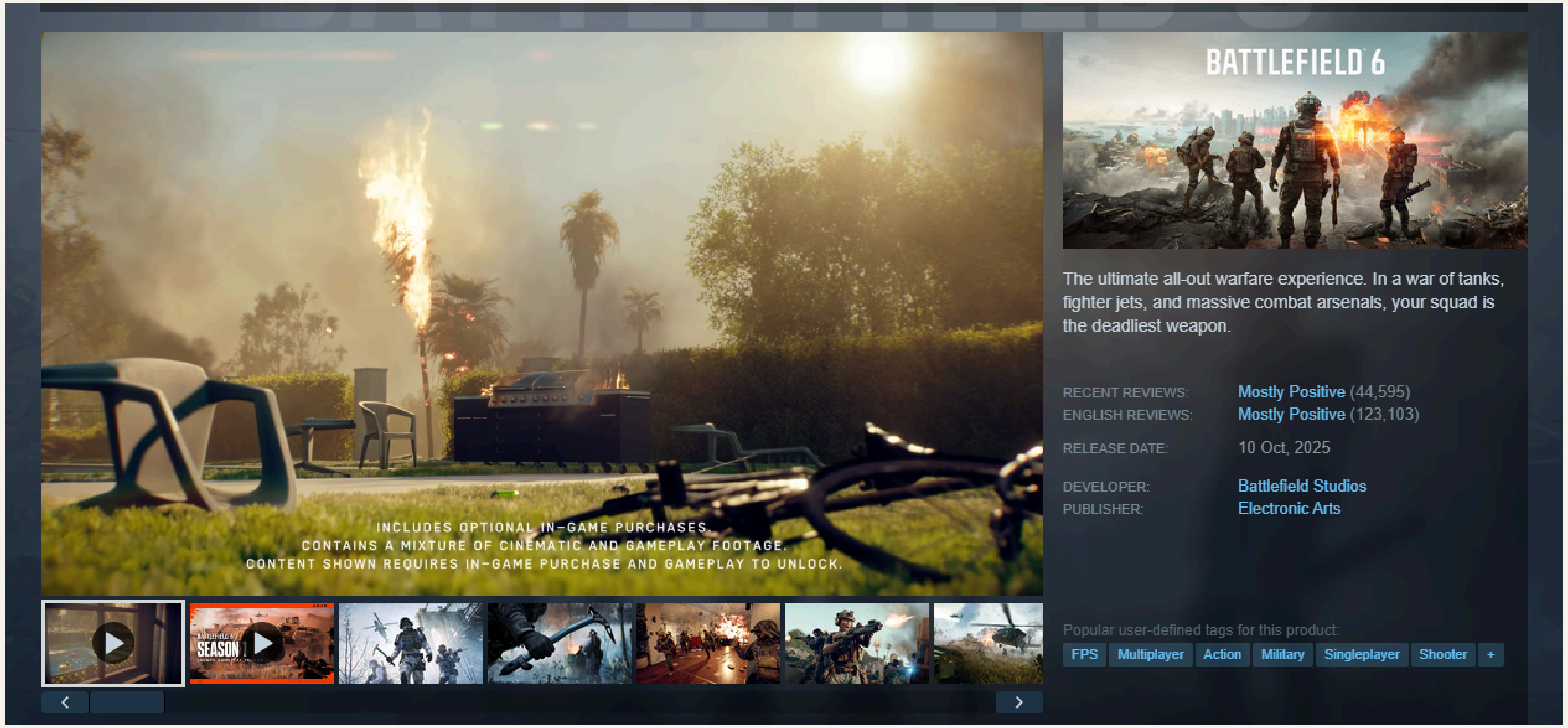
User: Registration, shopping cart, wishlist, order history, and a personal game library.

Admin: Game CRUD operations, payment monitoring, and user activity administration.

2. Methodology

Data Preparation

Use API to crawl data from Steam Application Website. Request API from local computer, then extract data from the returned JSON.



INCLUDES OPTIONAL IN-GAME PURCHASES.
CONTAINS A MIXTURE OF CINEMATIC AND GAMEPLAY FOOTAGE.
CONTENT SHOWN REQUIRES IN-GAME PURCHASE AND GAMEPLAY TO UNLOCK.

The ultimate all-out warfare experience. In a war of tanks, fighter jets, and massive combat arsenals, your squad is the deadliest weapon.

RECENT REVIEWS: Mostly Positive (44,595)
ENGLISH REVIEWS: Mostly Positive (123,103)
RELEASE DATE: 10 Oct, 2025
DEVELOPER: Battlefield Studios
PUBLISHER: Electronic Arts

Popular user-defined tags for this product:

2. Methodology

Requirements

Functional

User:

- User Registration
- User Authentication
- User Profile Management
- Review Management
- Session and Cookie Handling

Admin:

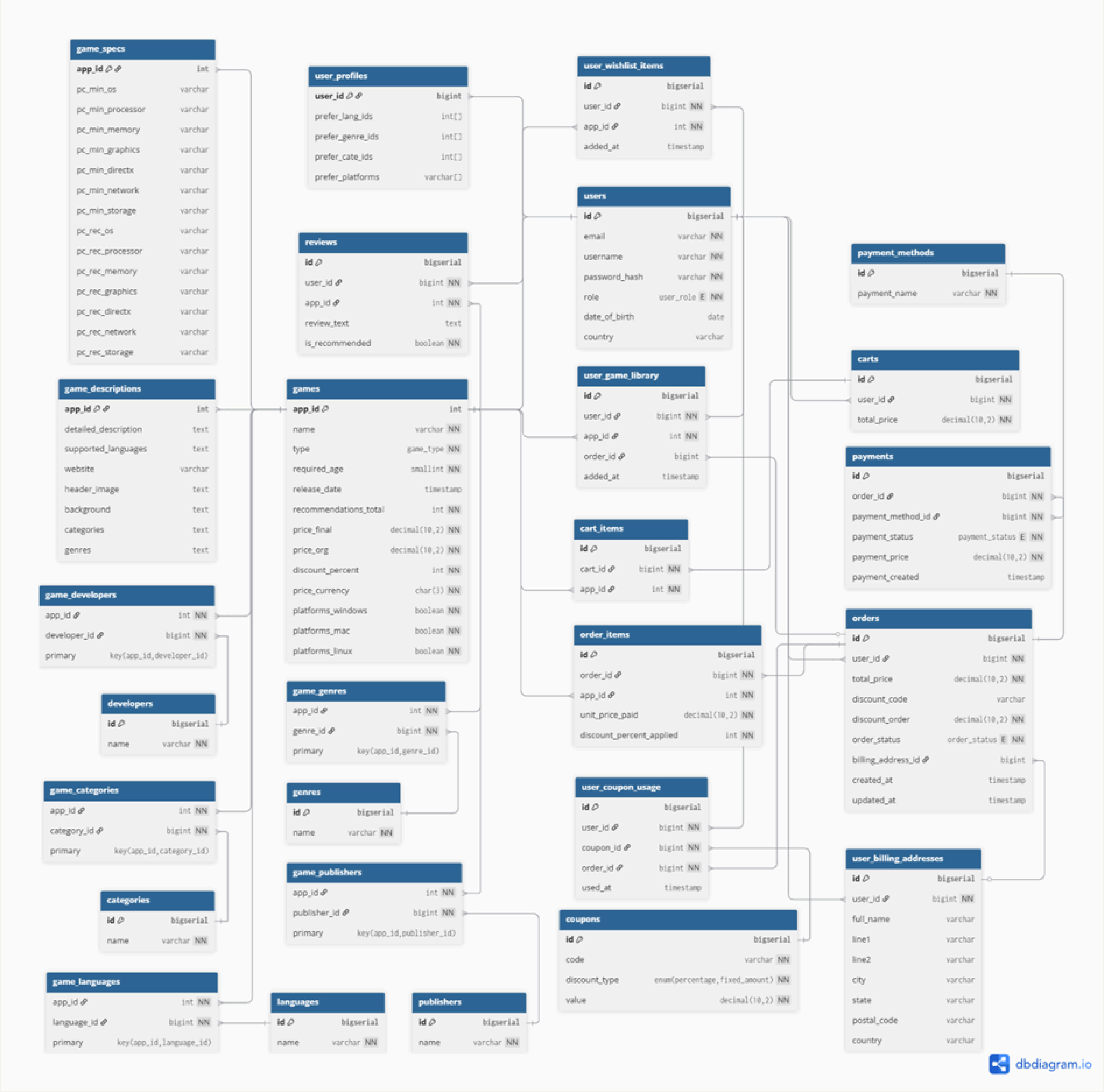
- Admin Authorization
- Admin Dashboard
- Admin Management

Non-Functional

- Performance
- Scalability
- Usability
- Security
- Compatibility
- Reliability
- Maintainability

2. Methodology

System Design -ERD



game_specs		
app_id	int	
pc_min_os	varchar	
pc_min_processor	varchar	
pc_min_memory	varchar	
pc_min_graphics	varchar	
pc_min_directx	varchar	
pc_min_network	varchar	
pc_min_storage	varchar	
pc_rec_os	varchar	
pc_rec_processor	varchar	
pc_rec_memory	varchar	
pc_rec_graphics	varchar	
pc_rec_directx	varchar	
pc_rec_network	varchar	
pc_rec_storage	varchar	

game_descriptions		
app_id	int	
detailed_description	text	
supported_languages	text	
website	varchar	
header_image	text	
background	text	
categories	text	
genres	text	

game_developers		
app_id	int	NN
developer_id	bigint	NN
primary	key(app_id,developer_id)	

developers		
id	bigserial	
name	varchar	NN

game_categories		
app_id	int	NN
category_id	bigint	NN
primary	key(app_id,category_id)	

categories		
id	bigserial	
name	varchar	NN

game_languages		
app_id	int	NN
language_id	bigint	NN
primary	key(app_id,language_id)	

user_profiles		
user_id	bigint	
prefer_lang_ids	int[]	
prefer_genre_ids	int[]	
prefer_cate_ids	int[]	
prefer_platforms	varchar[]	

reviews		
id	bigserial	
user_id	bigint	NN
app_id	int	NN
review_text	text	
is_recommended	boolean	NN

games		
app_id	int	
name	varchar	NN
type	game_type	NN
required_age	smallint	NN
release_date	timestamp	
recommendations_total	int	NN
price_final	decimal(10,2)	NN
price_org	decimal(10,2)	NN
discount_percent	int	NN
price_currency	char(3)	NN
platforms_windows	boolean	NN
platforms_mac	boolean	NN
platforms_linux	boolean	NN

game_genres		
app_id	int	NN
genre_id	bigint	NN
primary	key(app_id,genre_id)	

genres		
id	bigserial	
name	varchar	NN

game_publishers		
app_id	int	NN
publisher_id	bigint	NN
primary	key(app_id,publisher_id)	

languages		
id	bigserial	
name	varchar	NN

publishers		
id	bigserial	
name	varchar	NN

user_wishlist_items		
id	bigserial	
user_id	bigint	NN
app_id	int	NN
added_at	timestamp	

users		
id	bigserial	
email	varchar	NN
username	varchar	NN
password_hash	varchar	NN
role	user_role E	NN
date_of_birth	date	
country	varchar	

user_game_library		
id	bigserial	
user_id	bigint	NN
app_id	int	NN
order_id	bigint	
added_at	timestamp	

cart_items		
id	bigserial	
cart_id	bigint	NN
app_id	int	NN

order_items		
id	bigserial	
order_id	bigint	NN
app_id	int	NN
unit_price_paid	decimal(10,2)	NN
discount_percent_applied	int	NN

user_coupon_usage		
id	bigserial	
user_id	bigint	NN
coupon_id	bigint	NN
order_id	bigint	NN
used_at	timestamp	

coupons		
id	bigserial	
code	varchar	NN
discount_type	enum(percentage,fixed_amount)	NN
value	decimal(10,2)	NN

payment_methods		
id	bigserial	
payment_name	varchar	NN

carts		
id	bigserial	
user_id	bigint	NN
total_price	decimal(10,2)	NN

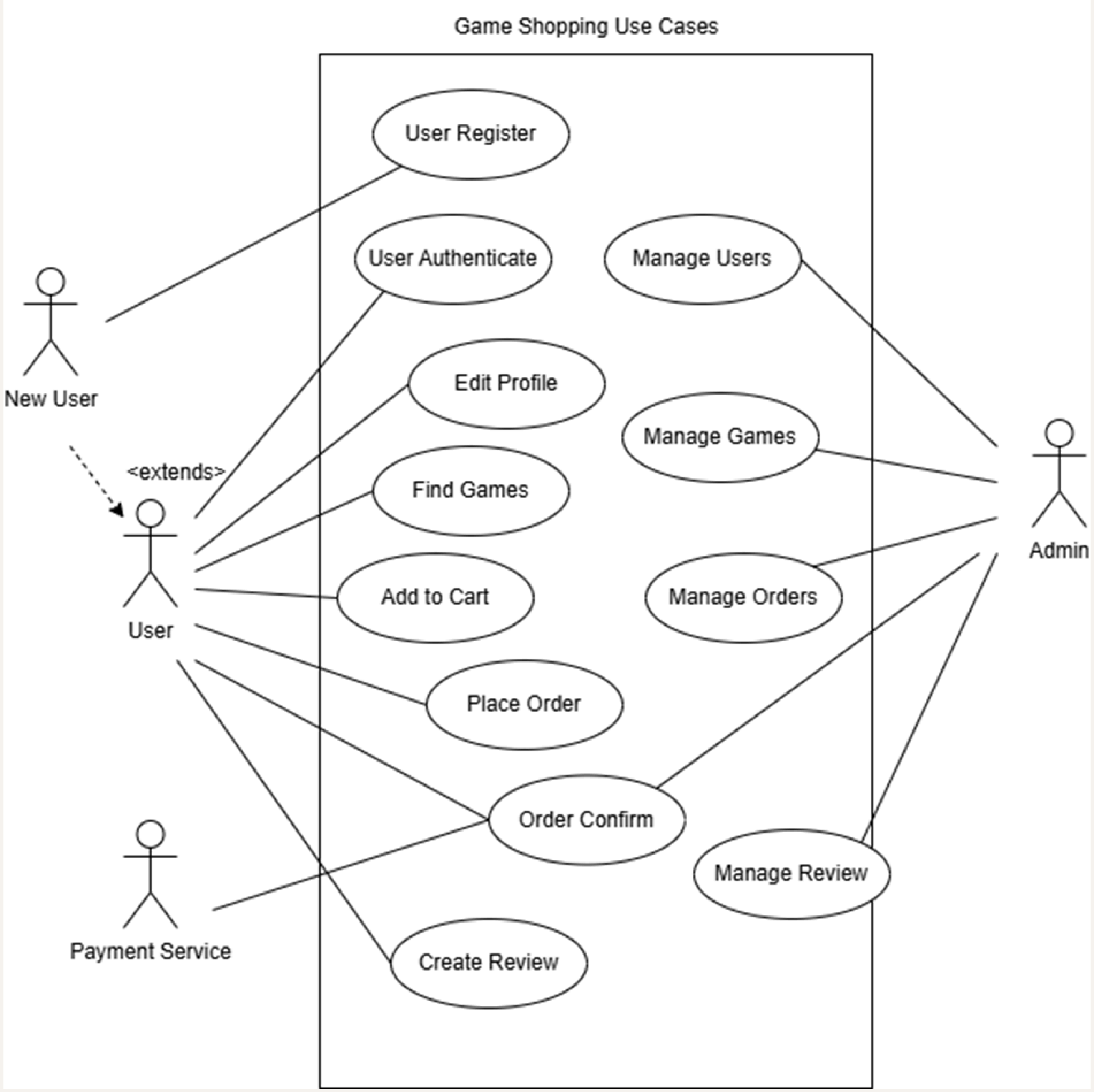
payments		
id	bigserial	
order_id	bigint	NN
payment_method_id	bigint	NN
payment_status	payment_status E	NN
payment_price	decimal(10,2)	NN
payment_created	timestamp	

orders		
id	bigserial	
user_id	bigint	NN
total_price	decimal(10,2)	NN
discount_code	varchar	
discount_order	decimal(10,2)	NN
order_status	order_status E	NN
billing_address_id	bigint	
created_at	timestamp	
updated_at	timestamp	

user_billing_addresses		
id	bigserial	
user_id	bigint	NN
full_name	varchar	
line1	varchar	
line2	varchar	
city	varchar	
state	varchar	
postal_code	varchar	
country	varchar	

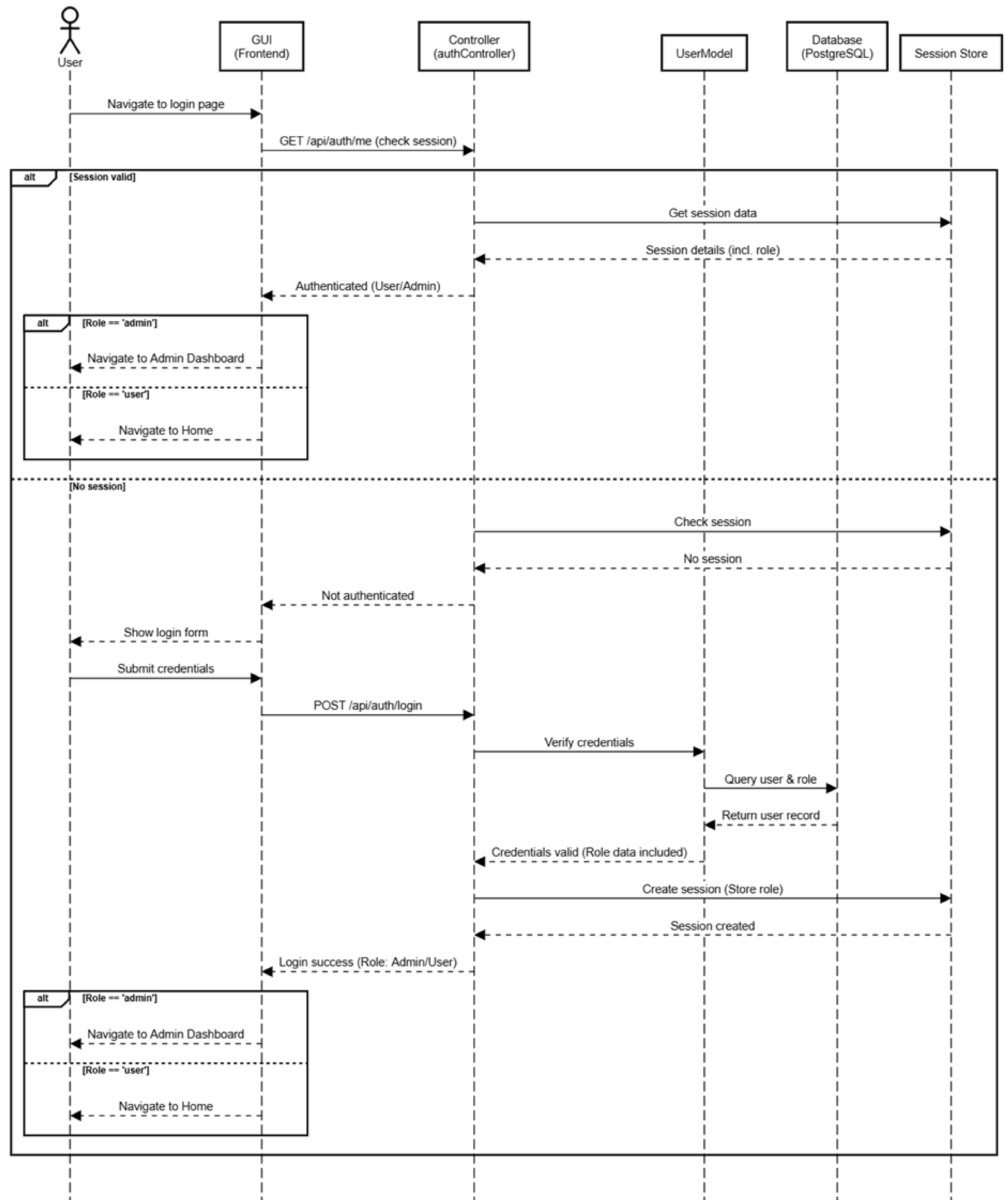
2. Methodology

System Design - Use Case



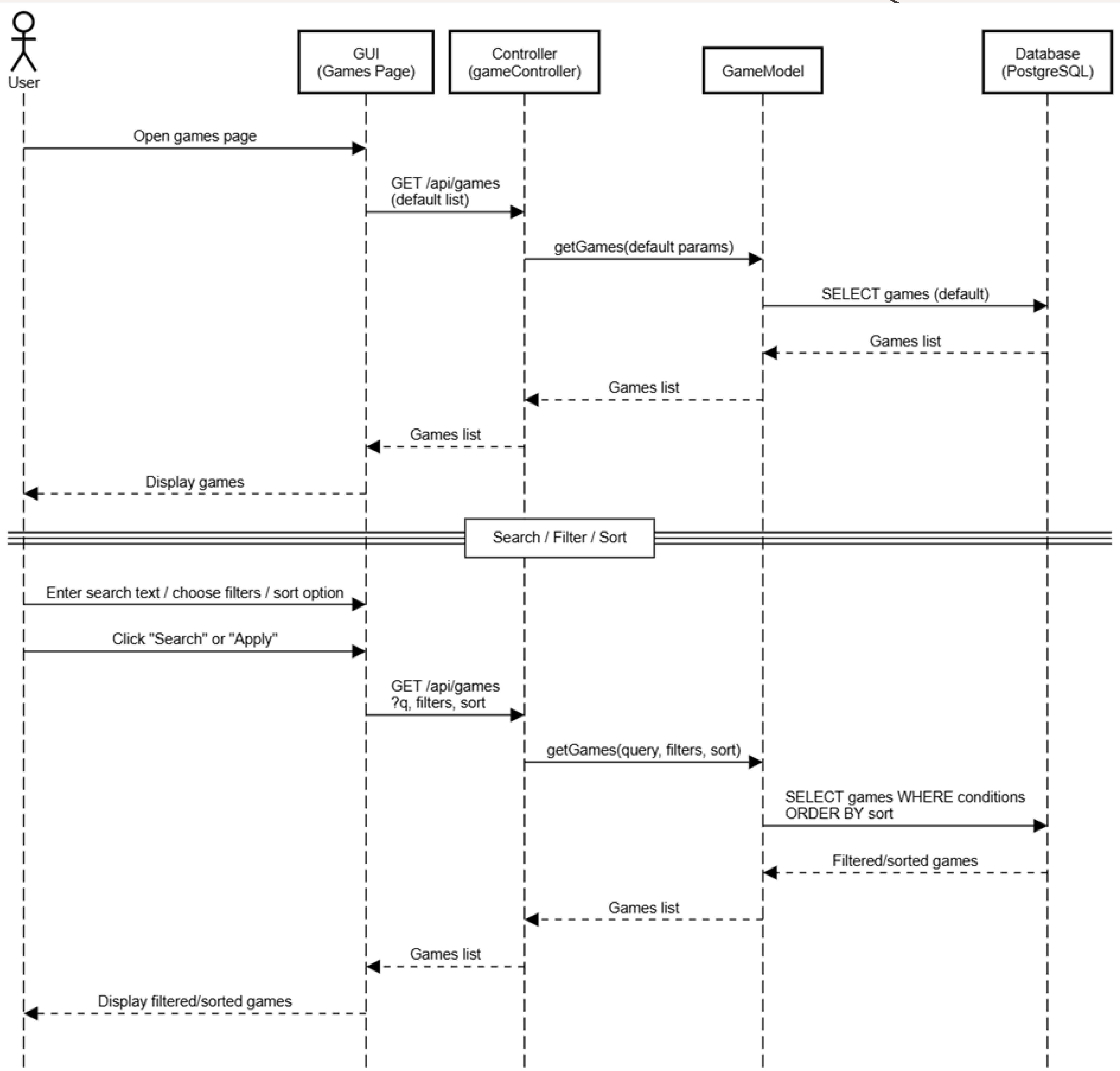
2. Methodology

User Registration



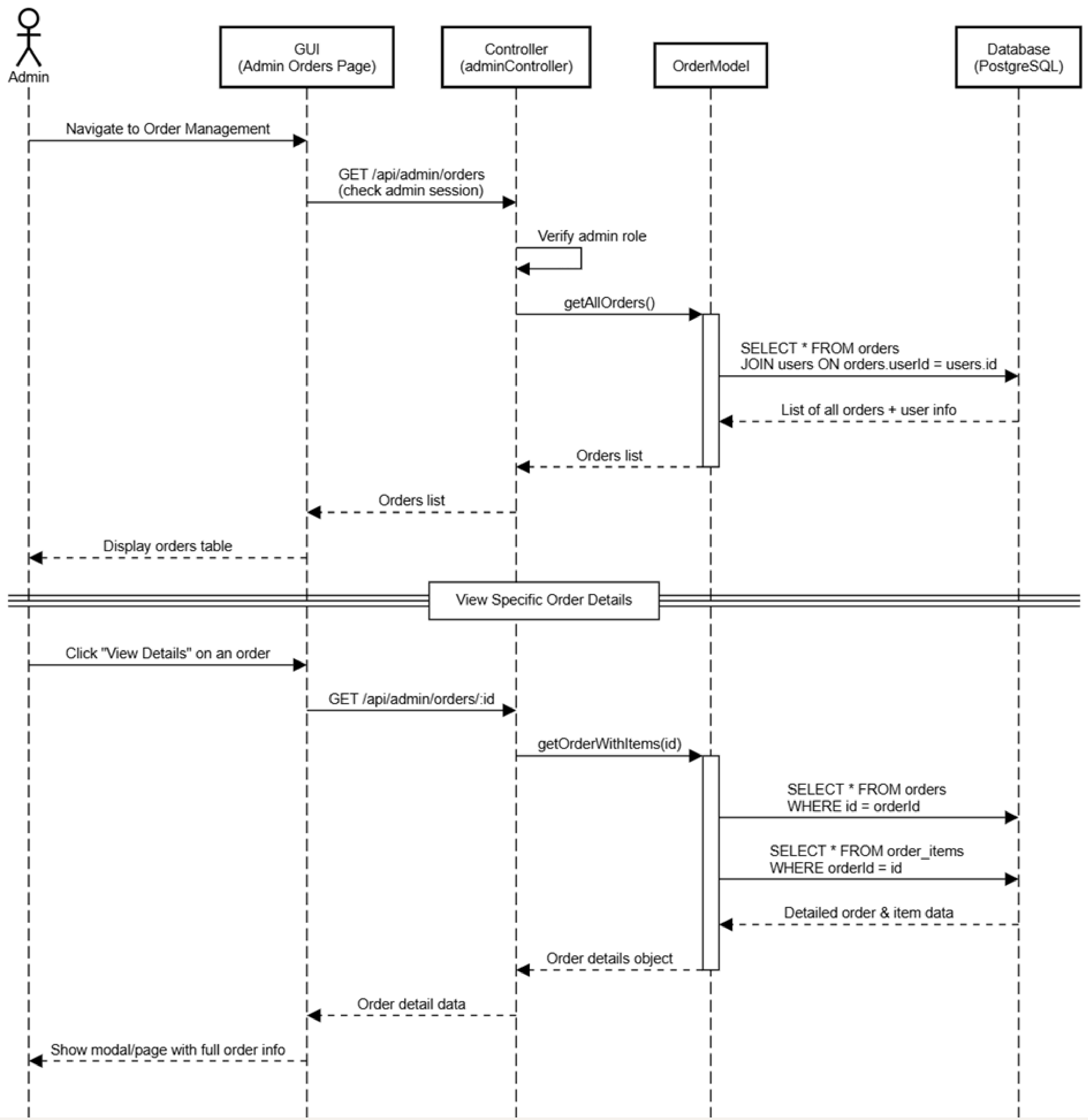
2. Methodology

User - Find Games



2. Methodology

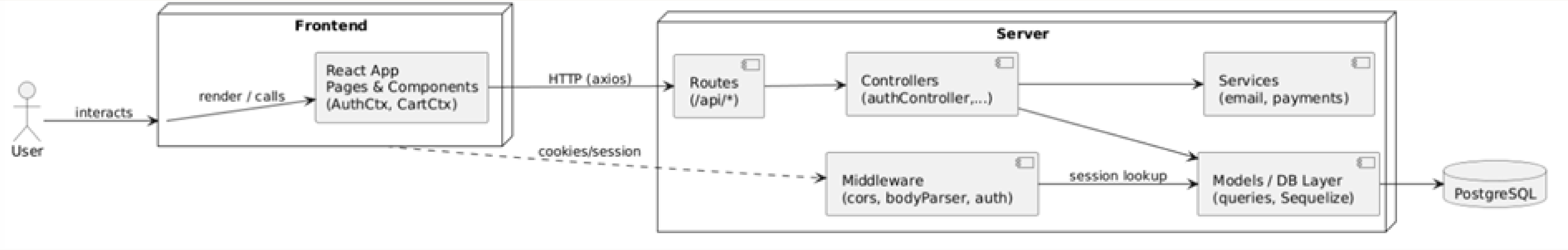
Admin - Manage Order



Admin
GUI (Admin Orders Page)
Controller (adminController)
OrderModel
Database (PostgreSQL)
Navigate to Order Management
GET /api/admin/orders (check admin session)
Verify admin role
getAllOrders()
SELECT * FROM orders JOIN users ON orders.userId = users.id
List of all orders + user info
Orders list
Orders list
Display orders table
View Specific Order Details
Click "View Details" on an order
GET /api/admin/orders/:id
getOrderWithItems(id)
SELECT * FROM orders WHERE id = orderId
SELECT * FROM order_items WHERE orderId = id
Detailed order & item data
Order details object
Order detail data
Show modal/page with full order info

2. Methodology

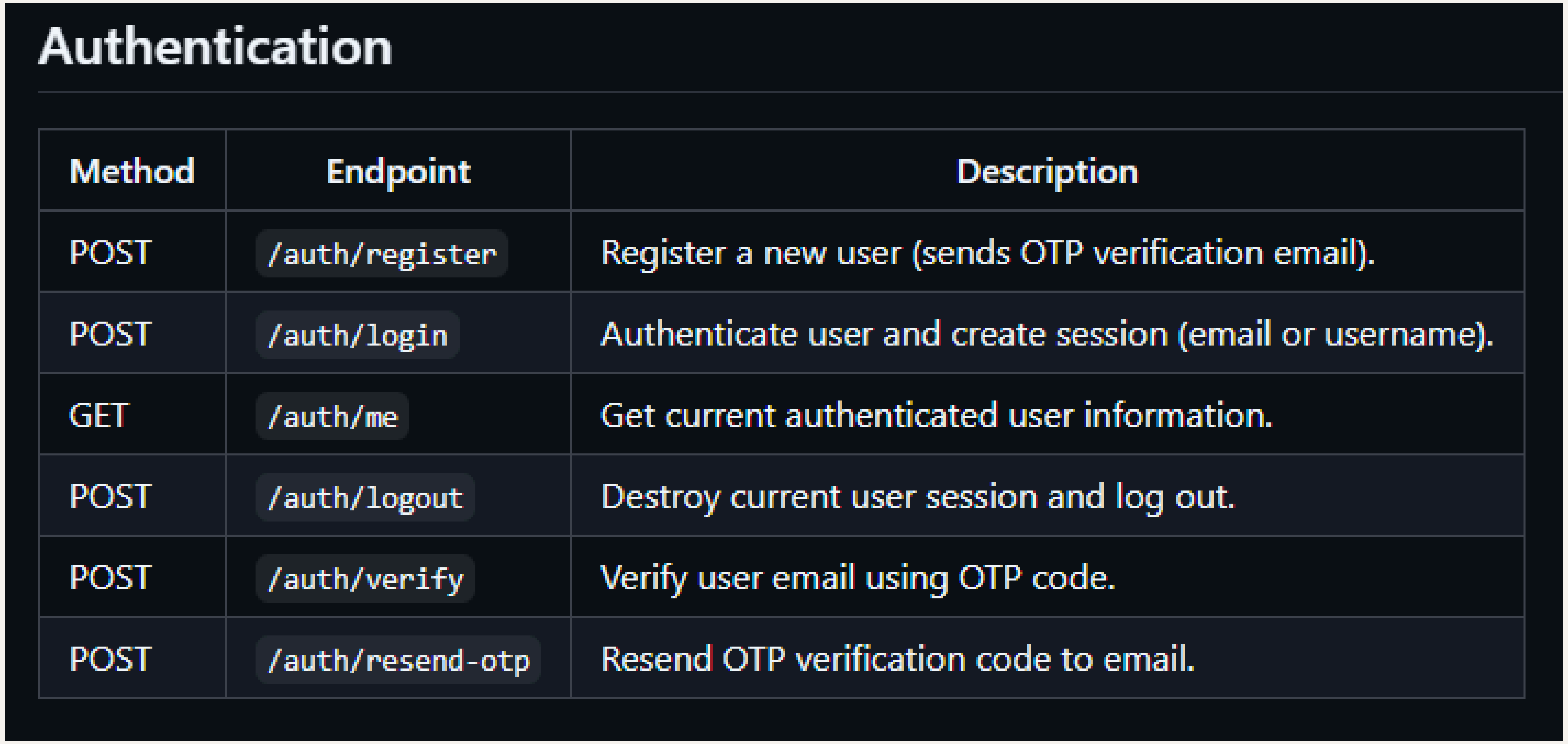
MVC Pattern



Input: The user interacts with the GUI, which sends a request to a specific endpoint on the Server.
Processing: The Controller receives the request and coordinates with the Model (e.g., UserProfileModel) to fetch or validate data.
Database Interaction: The Model interacts with PostgreSQL to retrieve or update the relevant tables.
Response: Once the database confirms success, the result travels back through the Controller to the Frontend, which updates the interface for the user.

3. Implementation

Authentication API

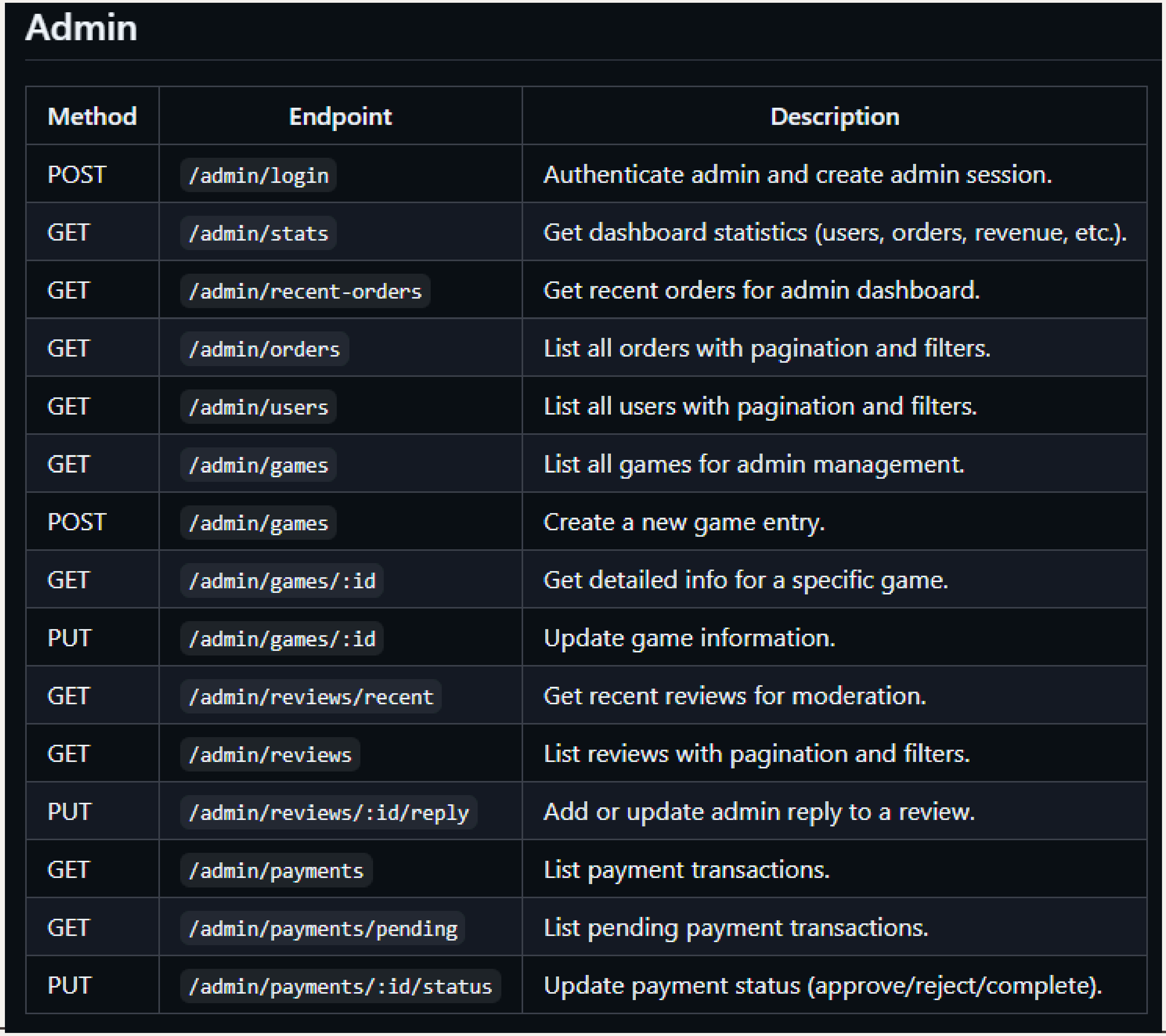


Authentication

Method	Endpoint	Description
POST	`/auth/register`	Register a new user (sends OTP verification email).
POST	`/auth/login`	Authenticate user and create session (email or username).
GET	`/auth/me`	Get current authenticated user information.
POST	`/auth/logout`	Destroy current user session and log out.
POST	`/auth/verify`	Verify user email using OTP code.
POST	`/auth/resend-otp`	Resend OTP verification code to email.

3. Implementation

Admin API

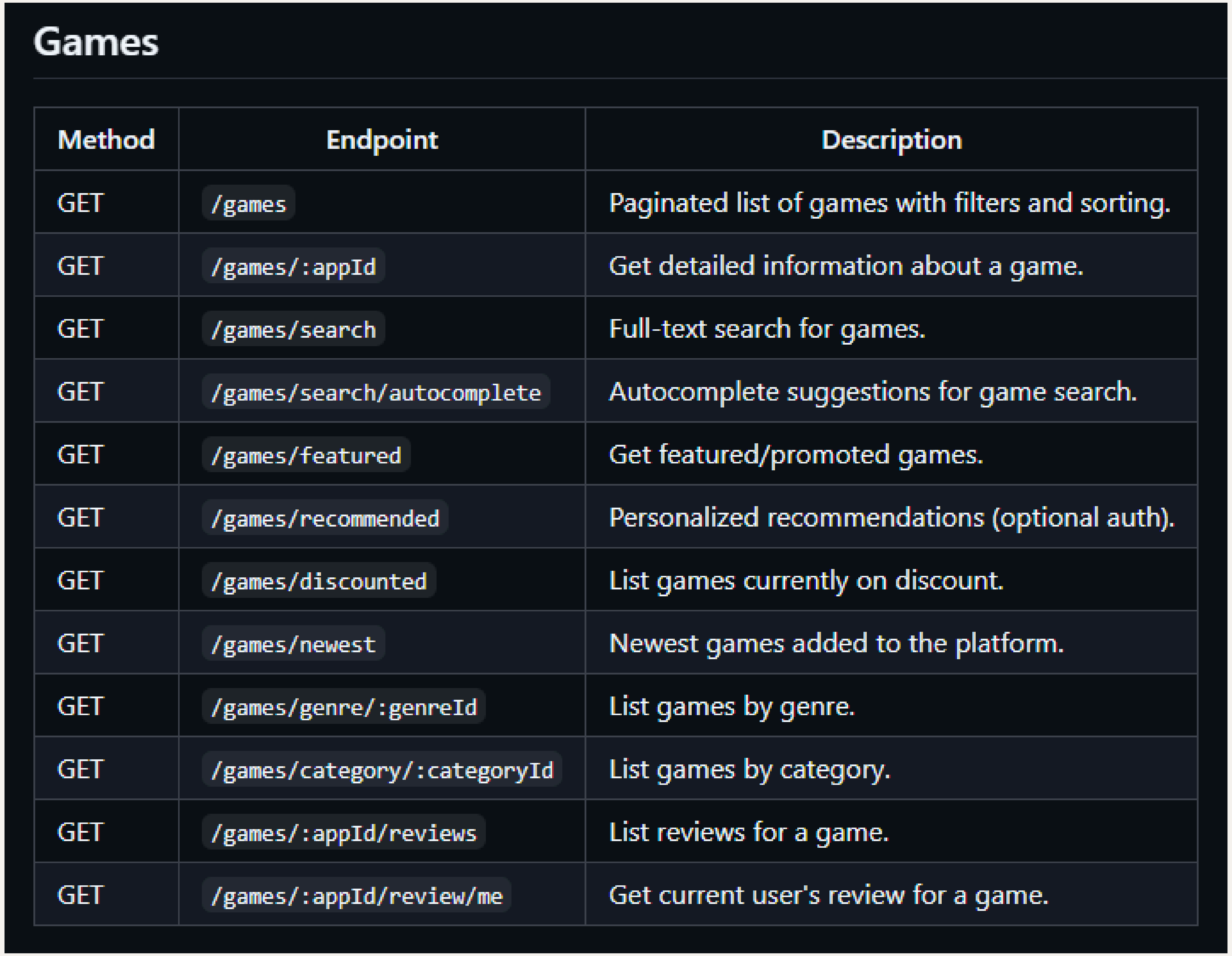


Admin

Method	Endpoint	Description
POST	`/admin/login`	Authenticate admin and create admin session.
GET	`/admin/stats`	Get dashboard statistics (users, orders, revenue, etc.).
GET	`/admin/recent-orders`	Get recent orders for admin dashboard.
GET	`/admin/orders`	List all orders with pagination and filters.
GET	`/admin/users`	List all users with pagination and filters.
GET	`/admin/games`	List all games for admin management.
POST	`/admin/games`	Create a new game entry.
GET	`/admin/games/:id`	Get detailed info for a specific game.
PUT	`/admin/games/:id`	Update game information.
GET	`/admin/reviews/recent`	Get recent reviews for moderation.
GET	`/admin/reviews`	List reviews with pagination and filters.
PUT	`/admin/reviews/:id/reply`	Add or update admin reply to a review.
GET	`/admin/payments`	List payment transactions.
GET	`/admin/payments/pending`	List pending payment transactions.
PUT	`/admin/payments/:id/status`	Update payment status (approve/reject/complete).

3. Implementation

Games API



Games

Method	Endpoint	Description
GET	`/games`	Paginated list of games with filters and sorting.
GET	`/games/:appId`	Get detailed information about a game.
GET	`/games/search`	Full-text search for games.
GET	`/games/search/autocomplete`	Autocomplete suggestions for game search.
GET	`/games/featured`	Get featured/promoted games.
GET	`/games/recommended`	Personalized recommendations (optional auth).
GET	`/games/discounted`	List games currently on discount.
GET	`/games/newest`	Newest games added to the platform.
GET	`/games/genre/:genreId`	List games by genre.
GET	`/games/category/:categoryId`	List games by category.
GET	`/games/:appId/reviews`	List reviews for a game.
GET	`/games/:appId/review/me`	Get current user's review for a game.

3. Implementation

Demo

4. Conclusion and Future Work

Conclusion

This project implements a full-stack e-commerce platform focused on digital game distribution. The system is organized as a traditional web application using a React single-page application (SPA) frontend and an Express backend with PostgreSQL as the primary datastore.

Future Work

- **Security Hardening:** Implement TLS/SSL termination to encrypt all data in transit
- **DevOps:** Apply Load Balancer to handle Web traffic such as Nginx
- **Scalability and Storage:** Moving from local memory to Redis cache for further horizontal scalling

Thank you <3