La mia casa

Project Specification

Zuzana Skubenova

March 2025

1 Basic Description of the Web Application

La mia casa is an online booking system for accommodation in a 6-room house in Italy. Users can book individual rooms or the entire house for selected dates. After their stay, they can leave reviews and ratings. Registered users earn loyalty points and receive birthday discounts.

The primary target users for the application are travelers searching for accommodation in Italy, as well as groups or families who wish to book the entire house. Additionally, administrators will use the system to manage reservations and reviews.

2 Functional Requirements

2.1 User Roles and Their Functionality

The application distinguishes 2 roles - guest and admin:

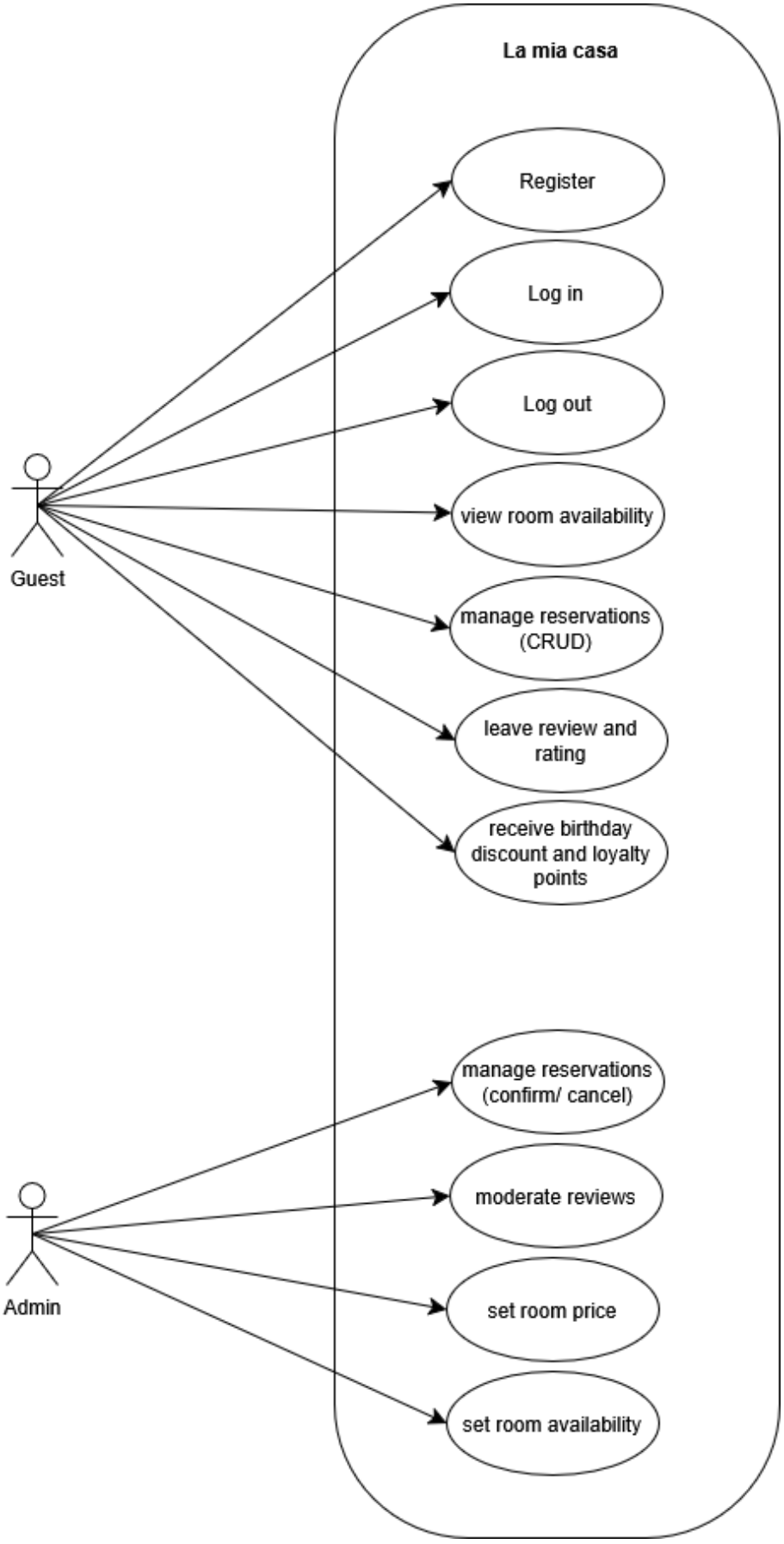
2.1.1 Guest

Guest is a traveler looking for accommodation in the house. They can register and log into the system to search for available rooms, make reservations, and manage their bookings. After their stay, they can leave reviews and ratings. Guests also earn loyalty points and receive birthday discounts, which can be used for future bookings.

2.1.2 Admin

The administrator is responsible for managing reservations, confirming or canceling bookings, and moderating user reviews. They also control room pricing and availability, ensuring that the booking system remains up to date. Admin accounts are created and managed at the database level.

3 Use Case Diagram



4 Data Model

The application follows a relational database model with multiple interconnected tables to manage users, reservations, rooms, reviews, and discounts.

4.1 Entities and Their Attributes

User

- `id` (integer, primary key) – Unique identifier for each user.
- `name` (string) – Full name of the user.
- `email` (string) – Email address used for authentication.
- `password` (string, hashed) – Securely stored password.
- `birth_date` (datetime) – User's date of birth, used for birthday discounts.
- `role` (enum: "guest", "admin") – Determines user permissions.

Room

- `id` (integer, primary key) – Unique identifier for each room.
- `name` (string) – Room name or number.
- `description` (text) – Detailed description of the room.
- `price_per_night` (decimal) – Cost per night for booking the room.
- `capacity` (integer) – Maximum number of guests allowed.
- `is_available` (boolean) – Indicates if the room is available for booking.

Reservation

- `id` (integer, primary key) – Unique identifier for each reservation.
- `user_id` (integer, foreign key) – References the user who made the reservation.
- `room_id` (integer, foreign key) – References the booked room.
- `start_date` (datetime) – Check-in date.
- `end_date` (datetime) – Check-out date.
- `status` (enum: "pending", "confirmed", "canceled") – Current reservation status.

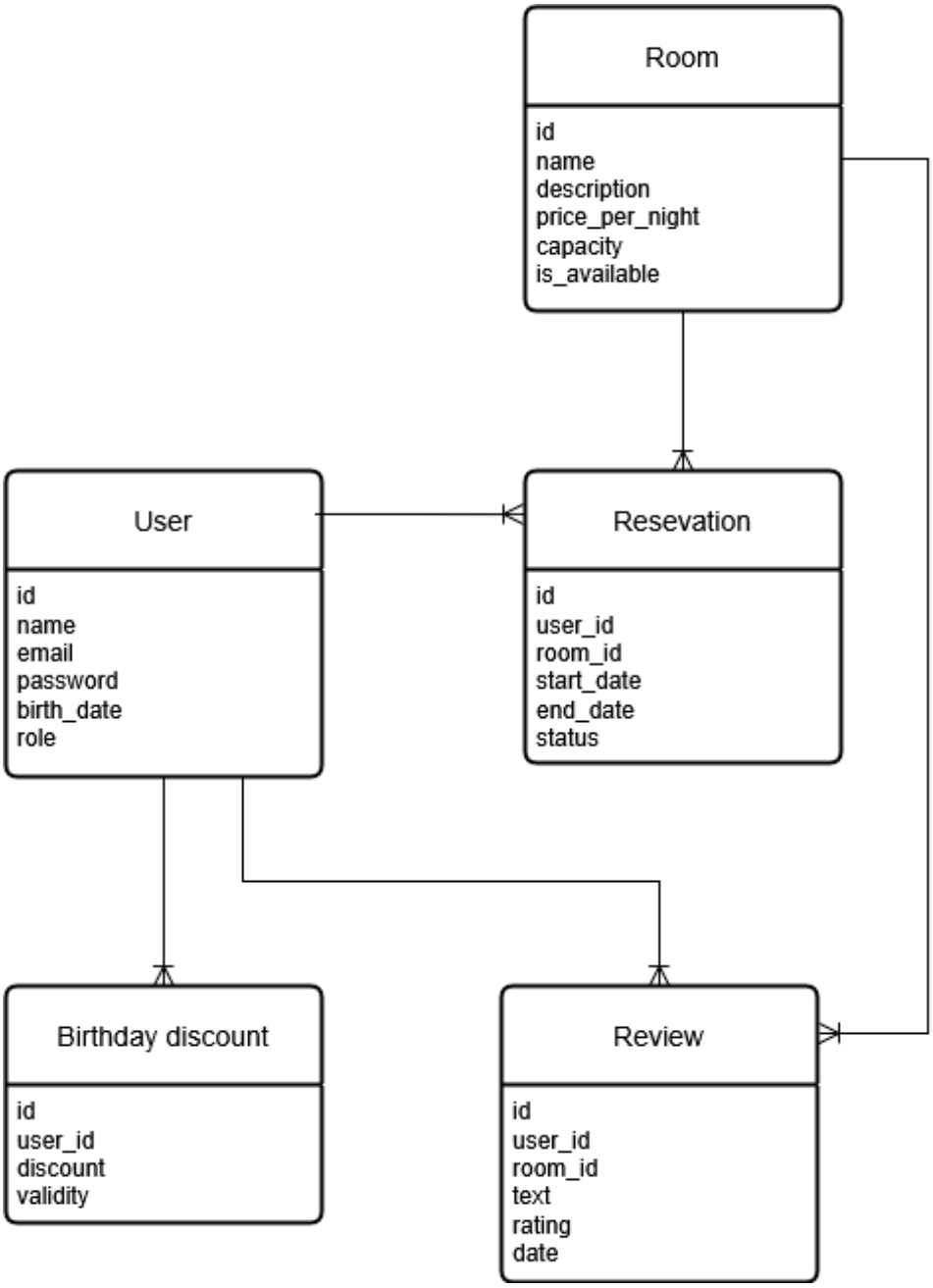
Review

- `id` (integer, primary key) – Unique identifier for each review.
- `user_id` (integer, foreign key) – References the user who left the review.
- `room_id` (integer, foreign key) – References the reviewed room.
- `text` (text) – Review content.
- `rating` (integer, 1-5) – Star rating given by the user.
- `date` (datetime) – Date when the review was submitted.

Birthday Discount

- `id` (integer, primary key) – Unique identifier for each discount.
- `user_id` (integer, foreign key) – References the user eligible for the discount.
- `discount` (decimal) – Discount percentage.

- `validity` (datetime) – Expiration date of the discount.



4.2 Relationships Between Entities

- A **user** can make multiple **reservations**, but a **reservation** belongs to only one **user**.
- A **room** can have multiple **reservations** over time, but each **reservation** is linked to a single **room**.
- A **user** can leave multiple **reviews**, but a **review** is associated with only one **user**.
- A **review** belongs to a specific **room**, but a **room** can have multiple **reviews**.
- A **user** can receive multiple **birthday discounts**, but each discount is assigned to only one **user**.

5 Architecture

The application will be based on the client-server architecture and it will use the SPA (Single Page Application) approach.

6 Technological Requirements

- **Client-side**: React 18, HTML5, CSS3, JavaScript, Bootstrap
- **Server-side**: Node.js, Express.js
- **Database**: PostgreSQL
- **Interface client-server**: REST API
- **Hosting**: render.com
- **Supported browsers**: Chrome, Firefox, MS Edge, Safari, Opera

7 Time schedule

Week	Task
4th week	Registration, Login
5th week	Booking system (CRUD operations for booking)
6th week	Review & rating system (reviews, loyalty points & discounts), Backend functionality + connection to FE
7th week	Admin interface (admin features & moderation, manage reservations, approve/remove reviews, managing room prices and availability)
8th week	Authentication (store hashed passwords, login/ logout, role-based access control - admin vs. guest), Deployment to hosting, Testing
9th week	Testing (ensure error handling for invalid requests, frontend and backend testing)
10th week	Incorporate feedback received on the beta version
11th week	Deployment to hosting and final adjustments (final bug fixes and testing, deploy on render)

8 Future Work

- Online payment integration.
- Multilingual support for the application.
- Expansion to include additional accommodation properties.