StayCation Technical Report

1st Juan Daniel Vanegas Mayorquin
System Engineering
Faculty of Engineering
Unversidad Distrital Francisco José de Caldas
Bogotá, Colombia
jdvanegasm@udistrital.edu.co

2nd Josue Urrego Lopez
System Engineering
Faculty of Engineering
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
jurregol@udistrital.edu.co

I. Business Model

A. Introduction

StayCation is an innovative platform designed to connect property owners with travelers seeking unique and comfortable accommodation experiences. StayCation aims to revolutionize the homestay rental market by offering a seamless and user-friendly interface for both hosts and guests. The business model of StayCation revolves around generating revenue through commission fees on each booking made via the platform. This model ensures a sustainable income stream while providing value-added services to our users.

B. Revenue Generation

StayCation's primary source of revenue is the commission fee charged to property owners for each successful booking facilitated through the platform. This commission is a percentage of the total rental price and is deducted at the time of booking. By adopting a commission-based model, StayCation aligns its financial interests with those of the property owners, ensuring mutual benefit from every transaction.

C. Service Offering

StayCation offers a comprehensive range of services to both property owners and travelers. For property owners, StayCation provides an easy-to-use platform to list their properties, manage bookings, and communicate with potential guests. The platform also includes tools for pricing optimization, calendar management, and automated messaging, which help hosts maximize their occupancy rates and income. For travelers, StayCation offers a diverse selection of accommodations, from urban apartments to rural homestays, catering to various preferences and budgets. The platform ensures a high level of security and trust through verified reviews, secure payment processing, and 24/7 customer support.

D. Database-Driven Functionalities

The StayCation platform is built upon a robust database that supports a wide range of functionalities, ensuring a smooth and efficient operation. Key functionalities include:

- **User Management**: Efficient handling of user information, roles, and contact details, ensuring secure authentication and personalized experiences for different user roles, such as hosts and guests.
- **Property Management**: Comprehensive management of property details, including types, locations, amenities, and availability. This allows hosts to provide detailed information about their listings, which can be easily searched and filtered by guests.
- **Booking System**: A streamlined reservation process that accurately tracks property availability and bookings, ensuring smooth coordination between hosts and guests.
- **Payment Processing**: Secure handling of user payment information, supporting financial transactions for bookings and ensuring that payments are processed efficiently and safely.
- **Reviews and Ratings**: A system for users to leave reviews and ratings for properties they have stayed at, enhancing trust and reliability by providing feedback to future guests.
- **Billing and Invoicing**: Generation and management of invoices for bookings, ensuring transparent and accurate financial records for both hosts and the platform.

E. Competitive Advantage

StayCation differentiates itself from competitors by focusing on personalized customer service and fostering a sense of community among its users. The platform emphasizes local experiences and cultural exchanges, appealing to travelers seeking more than just a place to stay. Additionally, StayCation leverages advanced technology, such as machine learning algorithms for personalized recommendations and dynamic pricing strategies, to enhance the user experience and increase conversion rates.

II. Stakeholders

The primary stakeholders of StayCation are the hosts and guests. Hosts, who list their properties on the platform, are crucial as they provide the inventory and variety of accommodations that attract travelers. Their satisfaction and engagement are essential for maintaining a diverse and appealing property selection. Guests, on the other hand, drive the demand side of the marketplace. Their experiences and feedback shape the platform's reputation and drive continuous improvement. Both groups are fundamental to StayCation's success, as they create a dynamic ecosystem where value is exchanged, ensuring the platform's sustainability and growth.

III. User Stories

A. For Guests

- As a guest, I want to create an account so that I can start booking properties.
- As a guest, I want to search for properties based on location and available dates so that I can find a place to stay that fits my schedule and preferences.
- As a guest, I want to view detailed descriptions, photos, and amenities of properties so that I can choose the best option for my stay.
- As a guest, I want to book a property for a specific date range so that I can secure my accommodation.
- As a guest, I want to pay for my booking securely using my credit card so that I can complete the reservation.
- As a guest, I want to leave a review and rating after my stay so that I can share my experience with other users.

B. For Hosts

- As a host, I want to create an account so that I can list my properties for rent.
- As a host, I want to list a new property with detailed information including type, location, amenities, and pricing so that potential guests can find and book it.
- As a host, I want to manage my property listings, including updating descriptions, photos, and availability so that my listings remain accurate and appealing.
- As a host, I want to view and manage bookings for my properties so that I can keep track of reservations and prepare for guests.
- As a host, I want to receive payments for bookings made on my properties so that I can earn income from rentals.
- As a host, I want to respond to reviews left by guests so that I can maintain a good reputation and address any concerns.

IV. Concept Model

The concept model for StayCation includes several core entities that represent the main components of the system. The relationships between these entities are crucial for understanding how the system operates and how data flows within it. Below is a detailed explanation of each entity and their relationships.

A. Entities

- **User**: Represents both property owners and travelers. Each user has a unique ID, role, name, email, hashed password, and phone number.
- **PropertyType**: Represents different types of properties available on the platform. Each type has a unique ID, name, and description.
- **PropertyAddon**: Represents additional features or services that a property can offer, such as WiFi, kitchen, parking, etc. Each addon has a unique ID and several boolean attributes indicating the availability of these features.
- **Property**: Represents properties listed by users on the platform. Each property has a unique ID, a reference to the user who owns it, a property type, a set of addons, location, guest capacity, available rooms, beds, baths, media URL, name, description, and price.
- **Card**: Represents payment cards linked to users for processing transactions. Each card has a unique ID, a reference to the user who owns it, card number, card owner, due date, CVV, and balance.
- **Booking**: Represents reservations made by travelers for properties. Each booking has a unique ID, references to the property and user, and a starting date.
- **Comment**: Represents reviews and ratings given by users for properties they have booked. Each comment has a unique ID, references to the booking and user, content, upload date, and a rating.
- **Bill**: Represents invoices generated for bookings. Each bill has a unique ID, references to the booking, property, and user, and a status indicating the bill's state.

B. Relationships

- A **User** can own multiple **Properties**.
- A **Property** belongs to one **User**.
- A **Property** has one **PropertyType**.
- A **Property** has one set of **PropertyAddons**.
- A **User** can have multiple **Cards**.
- A **Booking** is associated with one **Property** and one **User**.
- A **Comment** is associated with one **Booking** and one **User**.
- A **Bill** is associated with one **Booking**, one **Property**, and one **User**.

V. Tools

The development of StayCation utilizes various tools and technologies to ensure a robust, scalable, and efficient platform. Below is a detailed description of the tools used:

A. Database Management Systems

- **MySQL**: Used for managing relational data during the development phase. MySQL provides a reliable and efficient database management system that supports structured query language (SQL).
- **PostgreSQL**: Employed for managing relational data in the production environment. PostgreSQL is known for its advanced features, such as support for complex queries, foreign keys, triggers, and views.

B. Programming Languages and Frameworks

- **Python**: The primary programming language used for backend development. Python's simplicity and extensive libraries make it an ideal choice for rapid development and data manipulation.
- **Flask**: A lightweight web framework for Python used to create the web services for StayCation. Flask provides the necessary tools and libraries to build robust and scalable

web services, facilitating communication between the frontend and the database.

C. Virtualization

- **Docker**: Utilized for containerization and virtualization of both MySQL and PostgreSQL databases. Docker ensures that the application runs consistently across different environments by encapsulating the databases and their dependencies into isolated containers. This enhances the development workflow by providing a reproducible and isolated environment for testing and deployment.

VI. ER Diagram

The Entity-Relationship (ER) diagram for StayCation illustrates the key entities within the system and their relationships. This diagram is crucial for understanding the structure of the database and how different entities interact with each other. The ER diagram includes the following entities:

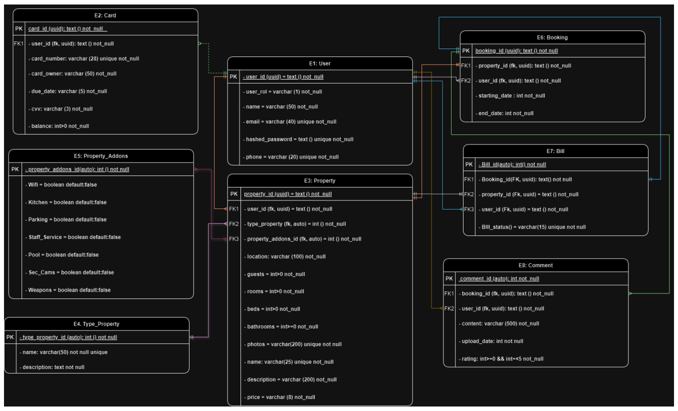
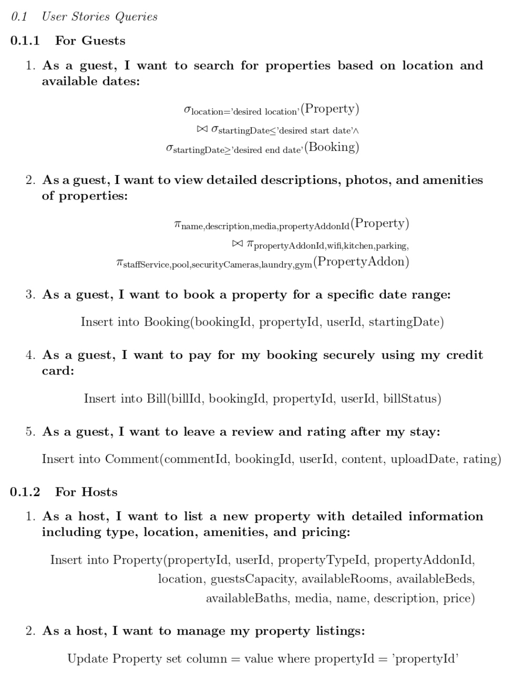

E2: Card
PK card_id (uuid): text () not_null
FK1 - user_id (fk, uuid): text () not_null
- card_number: varchar (28) unique not_null
- card_owner: varchar (50) not_null
- due_date: varchar (5) not_null
- cvv: varchar (3) not_null
- balance: int>0 not_null
E1: User
PK - user_id (uuid) = text () not_null
- user_rol = varchar (1) not_null
- name = varchar (50) not_null
- email = varchar (40) unique not_null
- hashed_password = text () unique not_null
- phone = varchar (20) unique not_null
E5: Property_Addons
PK - property_addons_id(auto): int () not null
- Wifi = boolean default:false
- Kitchen = boolean default:false
- Parking = boolean default:false
- Staff_Service = boolean default:false
- Pool = boolean default:false
- Sec_Cams = boolean default:false
- Weapons = boolean default:false
E3: Property
PK property_id (uuid) = text () not_null
FK1 - user_id (fk, uuid) = text () not_null
FK2 - type_property (fk, auto) = int () not_null
FK3 - property_addons_id (fk, auto) = int () not_null
- location: varchar (100) not_null
- guests = int>0 not_null
- rooms = int>0 not_null
- beds = int>0 not_null
- bathrooms = int>=0 not_null
- photos = varchar(200) unique not null
- name: varchar(25) unique not_null
- description = varchar (200) not_null
- price = varchar (8) not_null
E4. Type_Property
PK - type_property_id (auto): int () not null
- name: varchar(50) not null unique
- description: text not null
E6: Booking
PK booking_id (uuid): text () not null
FK1 - property_id (fk, uuid): text () not_null
FK2 - user_id (fk, uuid): text () not_null
- starting_date : int not_null
- end_date: int not_null
E7: Bill
PK - Bill_id(auto): int() not null
FK1 - Booking_id(FK, uuid): text() not null
FK2 - property_id (Fk, uuid) = text () not_null
FK3 - user_id (Fk, uuid) = text () not_null
- Bill_status() = varchar(15) unique not null
E8: Comment
PK comment_id (auto): int not_null
FK1 - booking_id (fk, uuid): text () not_null
FK2 - user_id (fk, uuid): text () not_null
- content: varchar (500) not_null
- upload_date: int not null
- rating: int>=0 && int=<5 not_null

Fig. 1. Entity-Relationship Diagram for StayCation

- **User**: Represents both property owners and travelers. Attributes include userId, userRole, name, email, hashedPassword, and phone.
- **PropertyType**: Represents different types of properties. Attributes include propertyTypeId, name, and description.
- **PropertyAddon**: Represents additional features a property can offer. Attributes include propertyAddonId, wifi, kitchen, parking, staffService, pool, securityCameras, laundry, and gym.
- **Property**: Represents properties listed by users. Attributes include propertyId, userId, propertyTypeId, propertyAddonId, location, guestsCapacity, availableRooms, availableBeds, availableBaths, media, name, description, and price.
- **Card**: Represents payment cards linked to users. Attributes include cardId, userId, cardNumber, cardOwner, dueDate, cvv, and balance.
- **Booking**: Represents reservations made by travelers. Attributes include bookingId, propertyId, userId, and startingDate.
- **Comment**: Represents reviews and ratings given by users. Attributes include commentId, bookingId, userId, content, uploadDate, and rating.
- **Bill**: Represents invoices generated for bookings. Attributes include billId, bookingId, propertyId, userId, and billStatus.

VII. Relational Algebra Queries

The relational algebra queries for StayCation address various functionalities required by the user stories and the database structure. Below are the queries expressed in relational algebra:



0.1 User Stories Queries

0.1.1 For Guests

1. **As a guest, I want to search for properties based on location and available dates:**

$$\sigma_{\text{location='desired location'}}(\text{Property})$$
$$\bowtie \sigma_{\text{startingDate}\leq\text{'desired start date'}\wedge}$$
$$\sigma_{\text{startingDate}\geq\text{'desired end date'}}(\text{Booking})$$

2. **As a guest, I want to view detailed descriptions, photos, and amenities of properties:**

$$\pi_{\text{name,description,media,propertyAddonId}}(\text{Property})$$
$$\bowtie \pi_{\text{propertyAddonId,wifi,kitchen,parking,}}$$
$$\pi_{\text{staffService,pool,securityCameras,laundry,gym}}(\text{PropertyAddon})$$

3. **As a guest, I want to book a property for a specific date range:**

$$\text{Insert into Booking(bookingId, propertyId, userId, startingDate)}$$

4. **As a guest, I want to pay for my booking securely using my credit card:**

$$\text{Insert into Bill(billId, bookingId, propertyId, userId, billStatus)}$$

5. **As a guest, I want to leave a review and rating after my stay:**

$$\text{Insert into Comment(commentId, bookingId, userId, content, uploadDate, rating)}$$

0.1.2 For Hosts

1. **As a host, I want to list a new property with detailed information including type, location, amenities, and pricing:**

$$\text{Insert into Property(propertyId, userId, propertyTypeId, propertyAddonId,}$$
$$\text{location, guestsCapacity, availableRooms, availableBeds,}$$
$$\text{availableBaths, media, name, description, price)}$$

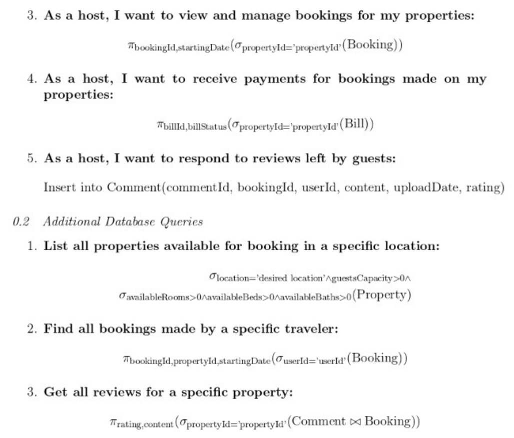
2. **As a host, I want to manage my property listings:**

$$\text{Update Property set column} = \text{value where propertyId} = \text{'propertyId'}$$

Fig. 2. Relational Algebra Queries 1

VIII. Database Filling Method

In the context of our staycation project, the database filling method encompasses the strategies and techniques employed to populate the database with initial or test data. This step is crucial to ensure that the database is ready for use in both development and testing phases, allowing for realistic simulations and verifications of system functionality. To achieve a comprehensive and effective database filling, we utilized a combination of manual data entry, automated scripts, and data simulation techniques. Each method was carefully chosen based on its suitability for different types of data and the requirements of the project.


3. **As a host, I want to view and manage bookings for my properties:**

$$\pi_{\text{bookingId,startingDate}}(\sigma_{\text{propertyId='propertyId'}}(\text{Booking}))$$

4. **As a host, I want to receive payments for bookings made on my properties:**

$$\pi_{\text{billId,billStatus}}(\sigma_{\text{propertyId='propertyId'}}(\text{Bill}))$$

5. **As a host, I want to respond to reviews left by guests:**

Insert into Comment(commentId, bookingId, userId, content, uploadDate, rating)

0.2 Additional Database Queries

1. **List all properties available for booking in a specific location:**

$$\sigma_{\text{location='desired location'}\wedge\text{guestsCapacity}>0\wedge}$$
$$\sigma_{\text{availableRooms}>0\wedge\text{availableBeds}>0\wedge\text{availableBaths}>0}(\text{Property})$$

2. **Find all bookings made by a specific traveler:**

$$\pi_{\text{bookingId,propertyId,startingDate}}(\sigma_{\text{userId='userId'}}(\text{Booking}))$$

3. **Get all reviews for a specific property:**

$$\pi_{\text{rating,content}}(\sigma_{\text{propertyId='propertyId'}}(\text{Comment} \bowtie \text{Booking}))$$

Fig. 3. Relational Algebra Queries 2

A. Manual Data Entry

Manual data entry was employed for inserting critical initial data that required high accuracy and validation. This included user profiles, staycation package details, and administrative settings. By manually entering this information, we ensured that the foundational data was accurate and met the predefined standards. This method, though time-consuming, was essential for establishing a reliable baseline dataset.

B. Automated Scripts

Automated scripts played a pivotal role in efficiently populating the database with bulk data. We developed custom scripts using Python and SQL to insert large volumes of data into the database tables. These scripts were designed to handle repetitive tasks, such as filling tables with user-generated content, booking histories, and service provider information. The use of automated scripts significantly reduced the time and effort required for database population while maintaining consistency and accuracy.

C. Data Simulation

To create a realistic testing environment, we generated simulated data that mimicked real-world scenarios. This involved using data simulation tools and techniques to produce synthetic data for various database entities. For instance, we simulated user interactions, booking patterns, and service reviews to test the system's performance under different conditions. The simulated data helped in identifying potential issues and optimizing the database design for better scalability and efficiency.

D. Verification and Validation

After populating the database, thorough verification and validation processes were conducted to ensure data integrity and consistency. This involved running SQL queries to cross-check data entries, performing sanity checks, and conducting data audits. Additionally, we used unit tests and automated testing frameworks to verify that the populated data met the expected criteria and supported the intended use cases. In summary, the database filling method for our staycation project was a multifaceted approach combining manual data entry, automated scripting, and data simulation. Each method contributed to building a robust and comprehensive database, enabling effective development, testing, and deployment of the staycation system.