Key Partners

Property owners / landlords and property management companies: They enable the business model by supplying the assets (properties) that generate bookings and revenue.

Mapping & geolocation providers: They enable the interactive, map-based interface that defines Havenly main competitive advantage.

Cloud service provider (Contabo): Hosts the platform's infrastructure, databases, and APIs, ensuring availability, scalability, and performance at low cost.

Analytics/BI vendors and ML tooling providers: They transform raw platform data into actionable insights and smarter user experiences.

Key Activities

Booking & reservation workflow: It operationalizes the platform's core function: connecting guests and hosts through confirmed reservations.
Recommendation engine and Personalization: It turns data into smarter interactions, enhancing user retention and competitiveness.

Key Resources

Listings database (properties, availability calendars, photos): It provides the foundational dataset that supports every core function of Havenly
Recommendation & ML models: They enable intelligent, data-driven decision-making.
Cloud Infrastructure: It offers a cost-effective and reliable environment to deploy and maintain Havenly's services

Value Propositions

Fast, accurate location-based property discovery with rich map UX: It delivers a visually engaging and efficient way to discover properties using precise geolocation.

Filtered search by location, price, amenities, dates and host type: It personalizes and streamlines the discovery process, saving users time and increasing booking conversions.

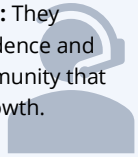
Real-time availability and instant booking / flexible booking options: It provides speed, accuracy, and convenience during the booking process.

Analytics dashboard for hosts/property managers to optimize revenue: It empowers hosts with actionable insights to improve profitability and listing quality.

Customer Relationships

Self-service portals for guests and hosts (manage listings, bookings): They empower users to manage their interactions directly, improving efficiency and scalability.

Community features (reviews, ratings, host badges): They strengthen user confidence and foster a sense of community that sustains long-term growth.



Channels

Responsive website: It provides universal access to the platform, guaranteeing usability and reliability for all users.

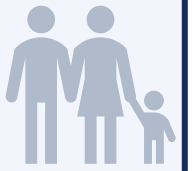
Third-party integrations / APIs (channel managers, OTAs): They expand the platform's reach and interoperability through secure and efficient data integrations.

Customer Segments

Individual hosts (single-property owners): They supply the inventory that sustains the platform's business model and depend on it for income generation.

Property management companies and real-estate agencies: They represent high-value, high-volume partners that expand the platform's professional network.

Corporate travel managers and long-term renters: They generate recurring, predictable bookings that increase platform stability.



Cost Structure

Personnel (engineering, data science, support): These costs guarantee operational continuity and innovation.

Costs for recommendation model training and BI infrastructure: They support intelligent decision-making and enhance user personalization.

Cloud Infrastructure: it provides a cost-effective foundation for secure, 24/7 cloud operations.

Revenue Streams

Service / booking fees charged to guests (percentage per booking): It monetizes the platform's core service while maintaining accessibility for users.

Commission / fee charged to hosts per booking: It balances revenue between both sides of the marketplace.

Data & analytics subscriptions for enterprise customers (aggregated, anonymized insights): They create a long-term, scalable revenue stream through data-driven services.

Corporate Finance Institute. (2019). Business Model Canvas Examples. CFI Education Inc. Retrieved from

For the Business Canvas Model, we chose the ones we considered were the most crucial in each block in Havenly's development considering is a database-focused application. Thus, when we finally picked a couple, we decided to give a brief explanation on how they are adding value to our project.

Functional Requirements (FR)

1. **User Registration & Authentication**
    - FR1.1: Users must be able to register as **guest** or **host**.
    - FR1.2: Users must be able to login/logout with secure password handling.
    - FR1.3: Hosts must be able to verify their properties (upload info, photos).
2. **Property Management (Hosts)**
    - FR2.1: Hosts must be able to create, update, and delete property listings.
    - FR2.2: Hosts must manage availability calendars for their properties.
    - FR2.3: Hosts must be able to view bookings and revenue history.
3. **Property Discovery (Guests)**
    - FR3.1: Guests must filter results by price range, amenities, and availability.
    - FR3.2: Guests must be able to view property details, reviews, and host profile.
4. **Booking & Reservations**
    - FR4.1: Guests must be able to book available properties for selected dates.
    - FR4.2: Users must not be able to double book for the same property/date.
    - FR4.3: Users must receive booking confirmations.
5. **Payments**
    - FR5.1: Guests must be able to pay for bookings using a payment gateway.
    - FR5.2: Hosts must receive payouts after bookings are completed.
6. **Reviews & Ratings**
    - FR6.1: Guests must be able to leave reviews and ratings after a stay.
    - FR6.2: Hosts must be able to respond to reviews.

Non-Functional Requirements (NFRs)

Performance

1. **NFR1.1:** The system must return property search results within **2 seconds** for at least **95% of user requests**.
   Justification: Based on an estimated **50 concurrent users** for this regional prototype. With indexed geospatial queries in PostgreSQL (PostGIS) and Contabo's 2 vCPU / 4 GB RAM VPS, this response time is achievable.
2. **NFR1.2:** Booking confirmations must be processed in **3 seconds or less** from payment submission to confirmation.
   Justification: Stripe and PayPal APIs average 1–1.5 s latency; additional processing time covers backend validation and message delivery.

Scalability

3. **NFR2.1:** The system must support **up to 100 concurrent users** without a response-time increase greater than **10%**.
   Justification: Value scaled down from Airbnb's global user base (275 M users) to a realistic regional prototype scope.
4. **NFR2.2:** The database must handle **50,000 property records** and **250,000 booking entries** while maintaining query performance under **3 seconds**.
   Justification: This dataset size represents a small, defensible subset of Airbnb's scale, feasible on PostgreSQL with spatial indexing.

Availability

5. **NFR3.1:** The platform must maintain **99.0% uptime**, allowing no more than **7 hours of downtime per month**.
   Justification: Contabo's SLA guarantees 99.9% uptime; the extra margin accounts for local maintenance activities.
6. **NFR3.2:** Scheduled maintenance windows must not exceed **2 hours per month** and should occur during off-peak usage hours.
   Justification: Minimizes user disruption while enabling necessary updates.

Security

7. **NFR4.2:** All client-server communication must use **HTTPS/TLS 1.2 or higher**.
   Justification: Required for PCI DSS compliance when processing payments.

8. **NFR4.3:** The system must perform **input validation on all user forms and API endpoints (100%)**.
   Justification: Prevents SQL injection, XSS, and other common web vulnerabilities.

Usability

10. **NFR5.1:** The web interface must fully load within **3 seconds** on a **10 Mbps** internet connection.
    Justification: Matches Google Lighthouse's performance benchmarks for responsive websites.
11. **NFR5.2:** A user must be able to complete the full booking process in **five clicks or fewer** from the search results page.
    Justification: Reduces interaction friction and improves conversion, consistent with Nielsen usability principles.

Maintainability

12. **NFR6.2:** The database schema documentation must be updated within **48 hours** of any structural modification.
    Justification: Keeps technical documentation synchronized with the system's actual design.

For the functional requirements, at first, we made them user-focused, once we got the user global vision, we tried to erase some of them to adjust the project's range. On the other hand, for the non-functional ones, we redefined them as they were too generic. We transformed them from general expectations into verifiable performance and quality targets that can be measured, tested, and justified technically.

Research on the number of users in a AIR-BNB

https://igier.unibocconi.eu/sites/default/files/media/publication/684.pdf

https://radicalstorage.com/travel/airbnb-statistics/

https://www.businessofapps.com/data/airbnb-statistics