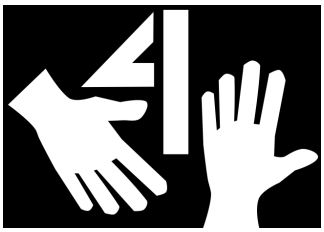

Hands4Hire - platform for connecting handymen with customers

Marta Borek Krzysztof Fijałkowski Tomasz Owienko
Michał Jakomulski Wojciech Sekuła Arkadiusz Niedzielski

Contents

1 Introduction

Connecting Customers with Trusted Service Providers — Simplified

Traditionally, finding a trustworthy handyman or home repair specialist has relied heavily on word-of-mouth recommendations from friends or neighbors. While personal referrals can be valuable, this informal system often makes it difficult for customers to discover available professionals — especially on short notice — and for skilled service providers to consistently find work or expand their client base.

Our platform was designed to address these challenges by offering a centralized, transparent, and reliable digital solution for household services. Whether it's plumbing, electrical repairs, or general handyman tasks, we simplify how Customers connect with Vendors based on availability, location, and service type.

At the same time, Vendors benefit from a dedicated space to showcase their services, manage scheduled visits, communicate with clients, and receive payments — all in one interface.

By streamlining this process, we aim to create a modern, scalable, and accessible ecosystem that benefits both sides of the marketplace.

2 Requirements

2.1 Use cases

2.1.1 Create Profile

Actor: Unregistered User
Preconditions: User is not yet registered or logged in
Basic steps:

1. User opens registration page
2. Authenticates Google account
3. Picks role
    - *Vendor* - service provider
    - *Customer* - service recipient
4. Provides detailed data necessary for the role
5. Pays registration fee
6. System creates profile and redirects to dashboard

Postconditions: Customer profile is stored in the system

2.1.2 Edit Profile

Actor: Registered Customer or Vendor
Preconditions: User is logged in
Basic steps:

1. User navigates to "Edit Profile" from the home page
2. Updates fields (e.g. skills, address, contact info)
3. Submits changes
4. System syncs data and confirms updates

Postconditions: Profile gets updated and reflects latest data

2.1.3 Update Calendar

Actor: Registered Vendor
Preconditions: Vendor is logged in
Basic steps:

1. Vendor opens calendar from profile page
2. Adds/deletes available slots
3. Can change status of a previously *accepted* booking to *canceled*
4. Submits changes
5. System syncs data and confirms updates

Postconditions:

- *Canceled* booking removed from the calendar, still visible in history
- Updated calendar is reflected in searches and Vendor profile
- Customer gets notification about Booking's status change

2.1.4 Search for Vendor

Actor: Registered Customer
Preconditions: Customer is logged in
Basic steps:

1. Customer opens search page
2. Selects filters (e.g. type of service, geographical range, availability, customer rating)
3. Submits search
4. System displays matching Vendors profiles

Postconditions: Filtered list is shown

2.1.5 View Vendor Profile

Actor: Registered Customer or Vendor
Preconditions:

1. User is logged in
2. Vendor Search/Filtering results available
3. **Or For Customer:** Vendor already in Customer's Booking History

Basic steps:

1. Select Vendor from Search List or Booking History
2. View Full Profile
    - Contact Info
    - Skills & Experience
    - Calendar availability
    - User rating and comments from previous jobs

Postconditions: System displays public Vendor's Profile

2.1.6 Book Vendor

Actor: Registered Customer
Preconditions:

1. Customer is logged in
2. Selected Vendor's Profile open

Basic steps:

1. Select time slot from Vendor's Calendar
2. Leave optional service note
3. Submit request

Postconditions:

- Booking request stored in the system with *pending* status
- Request added to Customer's booking history, visible in Customer's dashboard
- Notification sent to Vendor, request visible in Vendor's dashboard

2.1.7 Identity Verification during Visit

Actor: Registered Customer and Registered Vendor

Preconditions:

1. Customer is logged in
2. Vendor is logged in
3. Current time corresponds to predefined range before/during/after the scheduled Booking time (e.g., Booking time slot + 30 minutes before and after the slot)

Basic steps:

1. Customer gets notification about the upcoming visit + verification code (both email and in-app)
2. Vendor selects appropriate Booking from their Booking List
3. Selects "Verify"
4. Enters Customer's code into a form
5. Vendor sends completed form
6. Vendor gets immediate feedback on whether the code matches & option to retry if incorrect

Postconditions:

- Booking gets status update to *verified* in Customer's and Vendor's Booking Histories
- Notification about the Booking verification sent to both parties

2.1.8 View Customer Profile

Actor: Registered Vendor or Customer

Preconditions:

1. User is logged in
2. Vendor received Booking from a Customer
3. OR User views comments left by other Customers on Vendor's Profile

Basic steps:

1. Select Customer from Booking request, or Comments List
2. View Full Profile
    - Contact Info
    - Booking History
    - Comments and ratings for previous jobs

Postconditions: System displays public Customer Profile

2.1.9 Accept or Decline Booking

Actor: Registered Vendor

Preconditions:

1. Vendor received Booking from a Customer

2. Booking status is *pending*

Basic steps:

1. Vendor received a Booking Request notification (in-app/e-mail)
2. Notification redirection to Booking Request, with eventual authentication step
3. OR Open Booking request directly from dashboard
4. Booking Request shows:
    - Customer who placed the request
    - Date, Time & Duration
    - Service note from the Customer
    - Current status *pending, canceled*
5. If not *canceled* Vendor can Accept or Decline with designated buttons
6. System confirms the status update

Postconditions:

- Booking status changes to *accepted* or *declined*
- If *accepted*, is now visible in the Vendor availability Calendar and will be reflected in the Vendor Searches
- Customer gets a notification about the Booking status change

2.1.10 View & Cancel Booking Status

Actor: Registered Customer

Preconditions:

1. Customer placed at least one Booking

Basic steps:

1. Customer goes to Booking History on home page
2. Selects an entry to open a Booking page
3. OR Notification about Booking status update redirection to Booking Request, with eventual authentication step
4. Booking entry shows:
    - Booked Vendor
    - Date, Time & Duration
    - Service note from the Customer
    - Current status *accepted, declined, pending, canceled*
5. If *accepted* or *pending*, Customer can cancel the request
6. System confirms eventual status update

Postconditions:

- Booking status changes to *canceled*
- If previously *accepted* by Vendor, it is now removed from the Vendor availability Calendar and will be reflected in the Vendor Searches
- Vendor gets a notification about the Booking status change

2.1.11 Contact Between Customer & Vendor

Actor: Registered Customer or Vendor

Preconditions:

1. Booking request placed by Customer
2. OR User Profile viewed
3. OR Previously registered in the chat history

Basic steps:

1. User selects “Message” for a new chat from Recipient’s Profile
2. OR an active chat from chats list on the dashboard
3. Enters message in chat interface
4. Sends message
5. Recipient receives notification

Postconditions:

- Notification about a new message sent to the Recipient
- System stores new message and updates the chat history
- If no previous chat history with the Recipient, System adds new chat to the chat history, available from User dashboard

2.1.12 Comment & Rate Past Booking

Actor: Registered Customer
Preconditions:

1. Booking accepted by Vendor
2. Successful identity verification of Booking parties
3. Booking time slot passed

Basic steps:

1. User selects Booking from their Booking History
2. User selects “Rate and Comment”
3. Enters rating within a specified range
4. AND/OR Writes a comment
5. Selects “Publish”

Postconditions:

- Notification about a new rating sent to the Vendor
- System stores new rating and updates the affected Vendor’s profile
- Booking marked with *rated* status in Client’s Booking History
- Client cannot leave another comment/rating on that Booking

2.2 Functional requirements

2.2.1 User Profile Management

Registration

- Through Google account
- Requires a one time registration fee

User Profile

- Each user owns a profile with editable personal data
- Additionally:
  - *Vendors* profiles contain:
    - skills and specialization areas
    - User ratings and comments from previous jobs
    - Calendar with availability
  - *Customers* profiles contain:
    - Booking history
    - Ratings for previous bookings

Ratings and Comments

- Customers can leave feedback for the jobs from the Booking History
- Feedback in a form of a star range rating (scale up to 5 or 10) and an optional comment

2.2.2 Calendar

- Vendors manage their availability in a personal calendar, which is visible on their profile
- Customers view Vendors's calendar on their profile and can make booking requests within the free slots

2.2.3 Bookings Management

Vendor Search System for searching and filtering Vendors profiles according to:

- skills
- price
- location or geographical range
- availability
- customer rating

Booking placement Through calendars available at Vendor profiles, Customers can place bookings:

- Bookings can be accepted or declined by Vendors
- Bookings can be canceled by both Vendors and Customers, regardless of their status
- Status change of a Booking results in a notification to the second party

Report generation Automatic statistical report generation per Vendor profile. Includes:

- bookings count
- work hours
- earnings
- types of services

2.2.4 Communicator

- User Chat for communication between Vendors and Customers
- Code for Booking confirmation and identity authentication sent via chat

2.3 Non-functional requirements

- **Architecture** As dictated by the general project requirements:
  - Application is hosted on a public cloud (Google Cloud Provider)
  - Application consists of at least 3 types of microservices
  - The infrastructure build and setup are automated with the use of automation tool, Terraform
  - Database and state are owned by each microservice type separately, with two of them having private cache of the other's data subset
- **Performance & Responsiveness** System should support real-time updates for chat and booking confirmations.
- **Authorization & Security**

  - User identity verification is done with the use of trusted identity provider Google OAuth 2.0
  - Authentication in payment processing is done via Stripe secret keys, securely stored in GCP Secret Manager
- **Maintainability & Observability**
  - The use of automation tool, Terraform, allows for reproducible deployment
  - All microservices emit structured messages to Kafka and the Google Cloud Logging Agent

3 System Architecture

3.1 Visit Scheduler Module

3.2 Visit Manager Module

3.2.1 Routing

- **/register** - submit User's registration details
- **/register_visit** - visit registration, passed from the Visit Scheduler module, which handles Customer's input
- **/client/my_visits** -
- **/vendor/my_visits** - returns a list of visits that a Vendor has scheduled
- **/get_visit_code** - returns generated code for Vendor's identity confirmation during the visit (hash of visit_id for simplicity reasons)
- **/check_visit_code** - returns information about validity of the code submitted in a request by the Customer
- **/add_opinion** - enabled only after the scheduled visit took place, allows Customer to leave a rating (1-5 scale) and review

3.3 User Chat

3.4 Web Server

3.5 Kafka

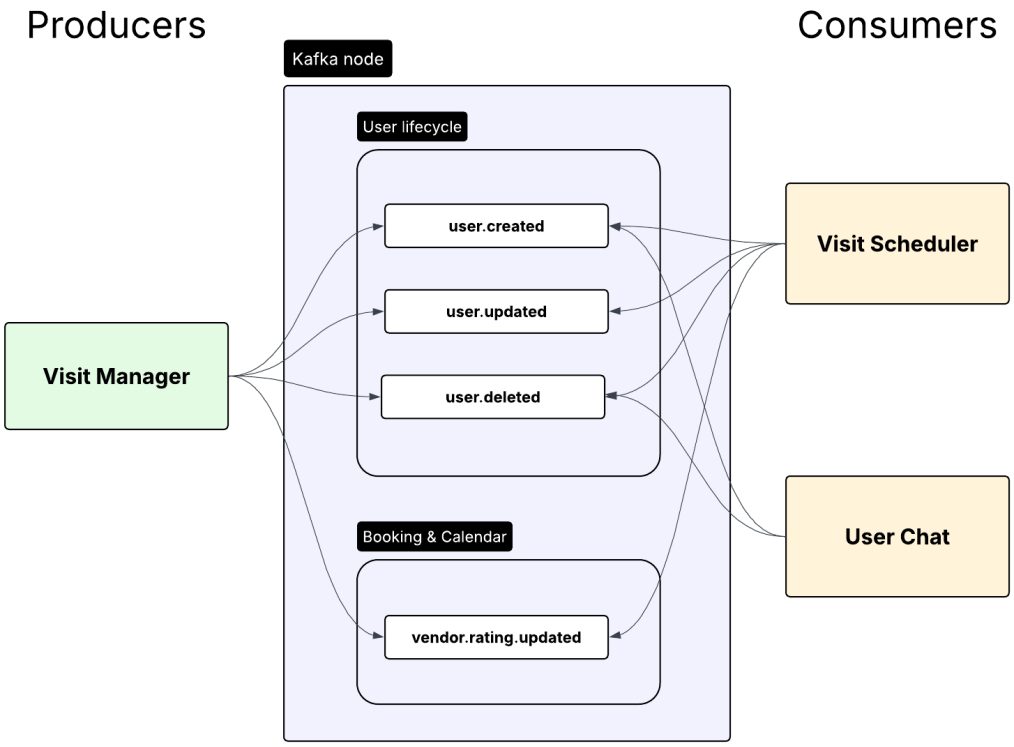

Figure 1: Kafka Topics Diagram

4 Security threat model

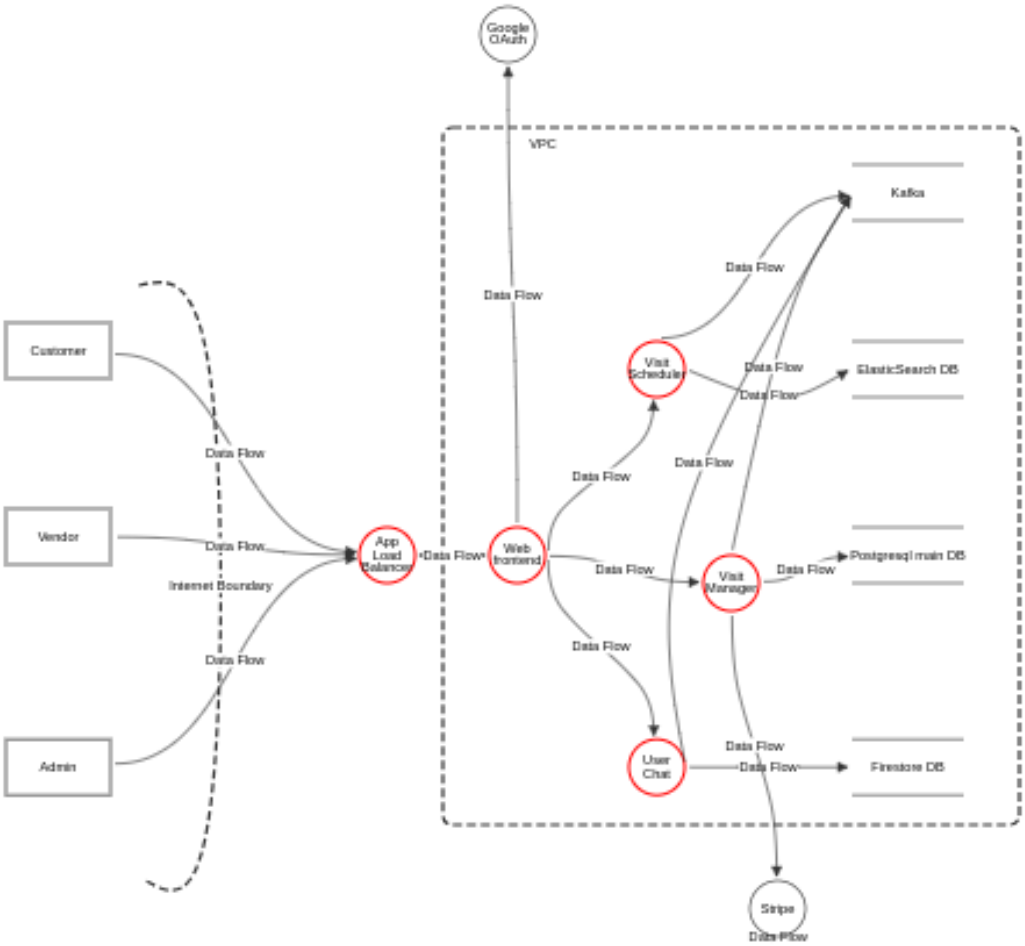

Figure 2: Threat modeling Data Flow Diagram

H4H

Owner:
Reviewer:
Contributors:
Date Generated: Wed May 28 2025

Executive Summary

High level system description

Not provided

Summary

Total Threats	26
Total Mitigated	15
Not Mitigated	11
Open / High Priority	5
Open / Medium Priority	6
Open / Low Priority	0
Open / Unknown Priority	0

DFD-H4H

Visit Scheduler (Process)

Description: FastAPI App

Number	Title	Type	Priority	Status	Score	Description	Mitigations
11	Forged user ID in requests	Spoofing	High	Open		Attackers could manipulate request bodies to act on behalf of another vendor/client and tampering with the issuing of visits that they book/accept and e.g. pay a deposit for.	
12	ElasticSearch query manipulation	Tampering	High	Open		Injection of malicious search filters to bypass constraints and return unauthorized data.	Use of proper ElasticSearch query builders, more resistant to such attempts.
13	Missing audit logs for bookings	Repudiation	Medium	Mitigated		Without proper event logging, users could deny placing a booking or changing their availability.	Proper logging and persistance and data retention.
14	Improper role enforcement	Elevation of privilege	Medium	Mitigated		Clients potentially accessing vendor-only functionality (calendar availability) and vice-versa, Vendors being able to book visits with other Vendors.	User role validation via JWT.
15	Heavy search queries	Denial of service	Medium	Open		Unbounded queries stressing ElasticSearch and slowing down its performance due to heavy processing.	Employ proper pagination within ES.

Visit Manager (Process)

Description: FastAPI app

Number	Title	Type	Priority	Status	Score	Description	Mitigations
23	Payment data manipulation	Tampering	High	Open		Modifying Stripe request data for altering payment amount, recipient etc.	
24	SQL injection	Tampering	High	Mitigated		Direct injection of malicious code.	Use of parametrized queries and ORM.
25	User Personally Identifiable Information leakage	Information disclosure	Medium	Open		Email, phone number, address exposed via poor API restrictions.	Following of GDPR guidelines in handling of data. Selective logging and masking of potentially sensitive information.
26	Stripe webhook forgery	Spoofing	High	Mitigated		Malicious requests resulting in faking of the payment events and status changes.	Validating of Stripe signatures and secrets.

User Chat (Process)

Description: FastAPI app

Number	Title	Type	Priority	Status	Score	Description	Mitigations
16	Impersonation via JWT replay	Spoofing	High	Mitigated		Interception and reusing of JWT tokens resulting in attackers sending messages on behalf of other users.	Short-lived JWT, enforce TLS encryption (HTTPS).

Number	Title	Type	Priority	Status	Score	Description	Mitigations
17	Unauthenticated message sending	Spoofing	Medium	Mitigated		Bots or anonymous users sending spam messages if endpoints lack strict authentication.	JWT authentication enforced on all endpoints.
18	Denial of message sending	Repudiation	Medium	Mitigated		User denying sending a message, difficult to prove message origin.	Proper logging and storing of message metadata, including origin verification.
21	Accidental logging of sensitive data	Information disclosure	Medium	Open		Sensitive message content may unintentionally get logged.	Log redaction, selective storing of only the necessary metadata.
22	Lack of logging for message delivery	Repudiation	Low	Mitigated		Loss of message traceability upon sending.	Implementation of proper message delivery receipts in the logging system.

App Load Balancer (Process)

Description: Nodejs App

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	JWT forwarding	Spoofing	High	Open		Attacker intercepting or replaying tokens due to improper security controls at ALB level; could result in full account takeover.	TLS encryption enforced upon the communication between the services (HTTPS). Proper token validation, short-lived JWTs.
2	Volumetric attacks	Denial of service	Medium	Mitigated		Large scale traffic floods designed to overwhelm ALB or downstream resources.	GCP Cloudflare employed.
3	Request header mangling	Tampering	Medium	Open		Attackers replacing specific request headers leading to potential bypass of security checks.	Proper header validation and sanitization.
4	Routing misconfiguration	Information disclosure	Medium	Mitigated		Misconfigured routing may expose internal endpoints to the public.	Blocking and restricting access to internal paths.
5	Lack of request attribution	Repudiation	Low	Mitigated		Without proper logging at the level of App Load Balancer it may be difficult to notice and track malicious activitiy.	Accurate monitoring setup at the outer-most layer and alerting on anomalies (Cloud Logging).
6	CORS misconfiguration	Information disclosure	Medium	Open		Without proper restriction, App Load Balancer may allow cross-origin requests, leaking data to potentially malicious sites.	Proper CORS configuration, specific headers and only trusted origins.

Firestore DB (Store)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations
19	Document manipulation	Tampering	High	Mitigated		If security rules are improperly configured, attackers may modify chat history records.	Strict security rules within the DB.
20	Access to unauthorized chat histories	Information disclosure	Medium	Mitigated		Missing access checks may lead to the potentially malicious users seeing conversations that they should not have access to.	Enforce Firestore access control for all documents.

Web frontend (Process)

Description: Nodejs app

Number	Title	Type	Priority	Status	Score	Description	Mitigations
7	Token leakage from localStorage	Spoofing	High	Open		Attackers getting user's locally stored credentials with the use of malicious scripts.	Use of PKCE in OAuth.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
8	Sensitive data leakage from localStorage	Information disclosure	Medium	Mitigated		Data such as chat history, or payment information, if stored locally can be leaked.	Avoid storing sensitive data client-side.
9	Volumetric attacks on backend endpoints	Denial of service	Medium	Open		Risk of repeated spam on the endpoints of the downstream microservices.	Enforcing rate limits per IP address or user.
10	Unauthorized access to dev tools	Elevation of privilege	High	Mitigated		Routes that are not blocked/restricted properly may expose access to administration/development features.	Routing safeguarding, access restriction via backend authorization.