Community Help & Micro-Task Coordination App
("Uber Works")

Cloud Computing – Homework 5

May 13, 2025

Contents

1 Executive Summary **1**

2 Case Study: Community Information Platforms **1**

3 Technical Review of Existing Solutions **2**

4 Proposed Architecture & Technology Stack **2**

5 Business Model Canvas **3**

6 Architectural Diagram **3**

7 Use-Case Diagram **4**

8 API Specification & Functionality Flows **4**

9 Example Functionality Flows **4**

1 Executive Summary

This report presents a proposed *Community Help & Micro-Task Coordination App* (codename: "Uber Works") that enables residents to share local information (e.g. roadblocks, lost items) and request on-demand assistance (e.g. tire changes, key retrievals) via a map-based interface. Leveraging Google Cloud Platform (GCP) services, the platform facilitates real-time pin drops, task matching, volunteer and paid help coordination, and secure communication, thereby strengthening neighborhood cohesion and improving response times for everyday needs.

2 Case Study: Community Information Platforms

International Context

- **Nextdoor** (U.S., 2011–present): Hyper-local social network for alerts, lost-and-found, and service requests. Relies on verified users and active moderation to maintain trust.
- **Carro** (India, 2022–present): WhatsApp-powered bot where community members report hazards; local volunteers respond with cleanup or warnings.

National / EU Context

- **Strada Mea** (Romania, 2017): Mobile app for reporting civic issues (potholes, broken streetlights) to municipalities; limited peer-to-peer help.

Key Lessons

- Verified identity & reputation scoring foster trust and safety.
- Combining public alerts with direct peer assistance covers both information *and* action needs.
- A simple map-pin + chat UX maximises engagement.

3 Technical Review of Existing Solutions

Solution	Technologies	Strengths	Gaps
Nextdoor	React Native, AWS, ElasticSearch	Large user base; robust moderation tools	Lacks integrated task payment and instant help
Facebook Neighborhoods	React, PHP, MySQL	Familiar platform; broad reach	No geospatial task matching; no paid-help support
TaskRabbit	Native iOS/Android, AWS Lambda, Stripe	Secure paid help; trusted payments	Not community-driven; no free volunteer mode

Table 1: Comparison of related platforms

4 Proposed Architecture & Technology Stack

Core Components

- **UI Layer**: App Engine (Node.js/Express or Java/Spring) delivering a PWA and mobile clients.
- **Database**: Cloud SQL (PostgreSQL) for structured data; PostGIS extension for spatial queries.
- **Object Storage**: Cloud Storage for media.
- **Eventing & Messaging**: Pub/Sub for broadcast alerts; Firebase Cloud Messaging (FCM) for push.
- **Serverless Logic**: Cloud Functions for pin validation, helper matching, Stripe payments.
- **Real-Time Sync**: Firestore for live map pins & chat.
- **Analytics**: Cloud Monitoring, Logging, BigQuery for insights and auditing.

Motivation

App Engine & Cloud Functions scale instantly during local events; Firestore + Pub/Sub offer sub-second updates; PostGIS delivers efficient nearest-helper matching.

5 Business Model Canvas

Key Partners	Local municipalities; Volunteer organizations; Payment processors; Telecom providers.
Key Activities	Platform maintenance; Geospatial data integration; Fraud & reputation management; Real-time notifications.
Key Resources	GCP services; DevOps team; GIS data; User profiles.
Value Propositions	Faster community response; safer neighborhoods; low-cost volunteer mobilization; micro-payments and tipping; context-rich hazard alerts.
Customer Relationships	In-app support; community forums; reputation scoring system.
Channels	Progressive web & mobile apps; social media referrals; API integrations; SMS fallback.
Customer Segments	Residents; local volunteers; gig-workers; elderly / non-technical users.

6 Architectural Diagram

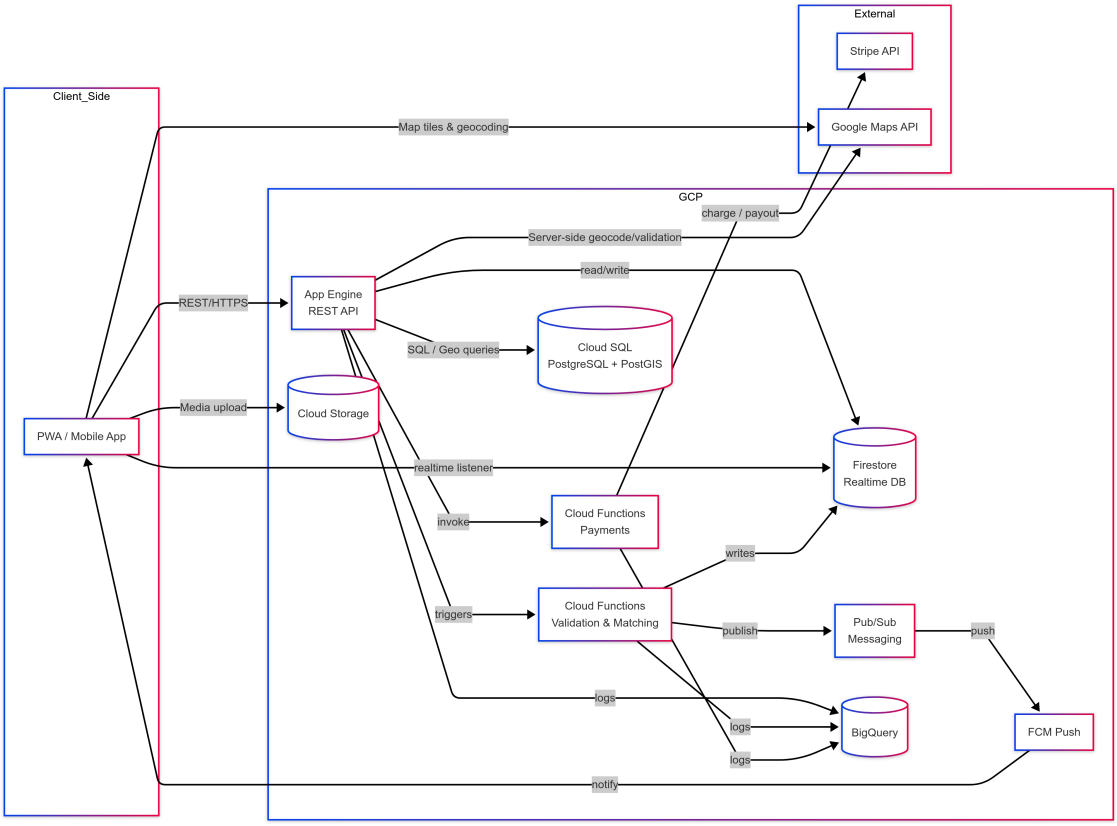

Figure 1: High-level container architecture

7 Use-Case Diagram

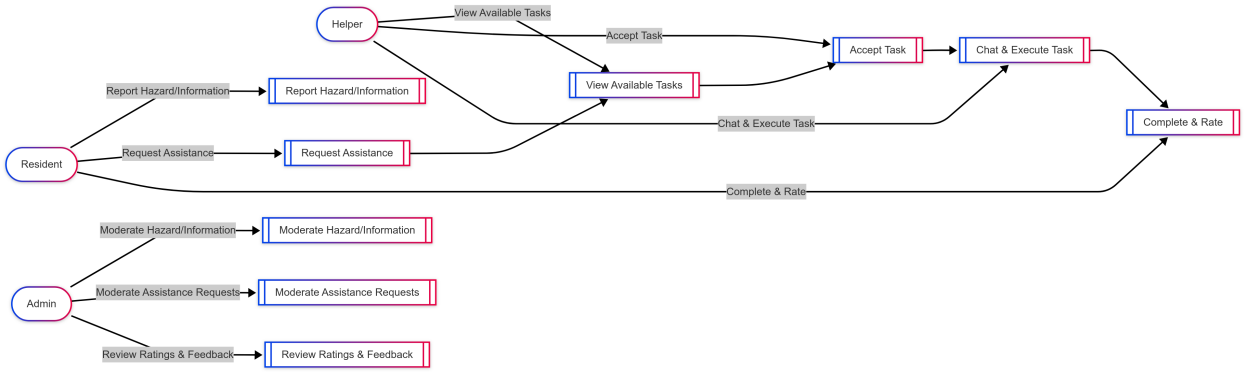

Figure 2: Primary actors and use-cases

8 API Specification & Functionality Flows

Full OpenAPI 3.0 definition is published on SwaggerHub under the project `community-uberworks-swagger`. Core endpoints:

Map Pin API (`/pins`)

- **POST** `/pins` – create pin (`type, lat, lng, mediaUrl?, description?`)
- **GET** `/pins?bbox=` – fetch pins in bounding box

Task API (`/tasks`)

- **POST** `/tasks` – request help (`category, lat, lng, details, paid, price?`)
- **GET** `/tasks/nearby` – list nearby tasks
- **PUT** `/tasks/{id}/accept` – accept assignment

User & Reputation API (`/users`)

- **GET** `/users/{id}/reputation`
- **POST** `/users/{id}/report` – flag content

9 Example Functionality Flows

Reporting a Hazard

1. Resident → `POST /pins`
2. Cloud Function validates media
3. Pin stored in Cloud SQL/PostGIS

4. Pub/Sub publishes event → FCM push → map refresh via Firestore

On-Demand Task

1. Resident → `POST /tasks` (indexed in PostGIS)
2. Helpers query `/tasks/nearby`
3. Helper → `PUT /tasks/{id}/accept` (Stripe payment link)
4. Chat opens in Firestore
5. Resident rates helper; Cloud Function updates reputation

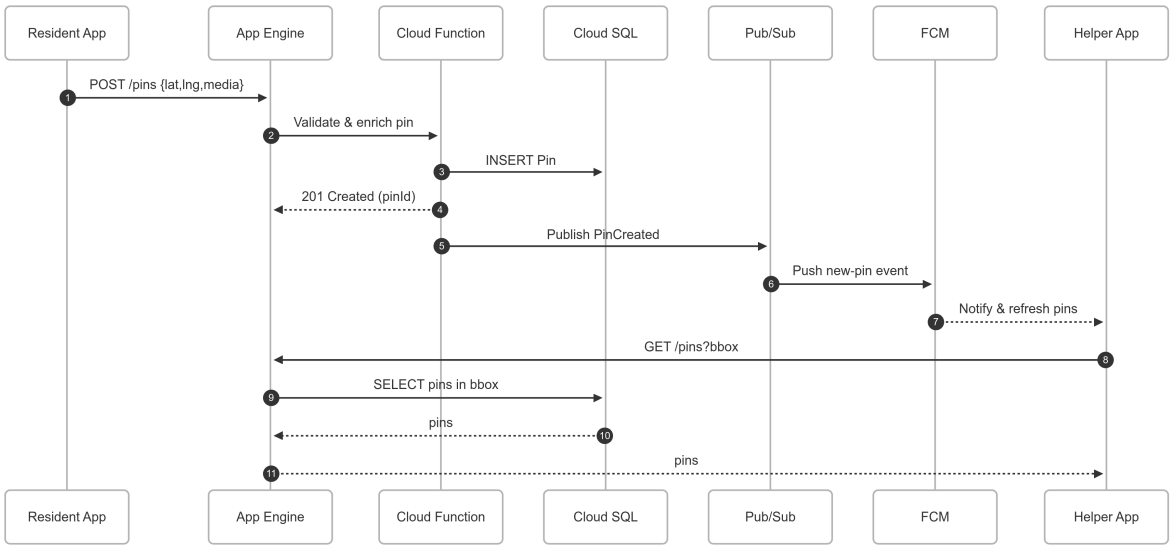

Figure 3: Sequence flow – "Report Hazard/Info"