OpenBid: Map-first Bidding Marketplace

DIGT 2107 — Project Iteration 1.1: Initial Vision & Planning

Team 1: Tyler, Mani, Yanness, Alaister

September 10, 2025

Course: DIGT2107 – Fall Term 2025 | **Instructor:** Dr. May Haidar

Contents

1 Introduction

Project Name: OpenBid
Team Number: 1
Team Members: Tyler, Mani, Yanness, Alaister

Document Overview This document follows the university's Iteration 1.1 guidelines for Initial Vision and Planning. It outlines the project vision, high-level features, and a near-term iteration plan aligned with the course schedule. A detailed breakdown of user stories and requirements will be provided in the next iteration submission.

2 Project Vision

Vision Statement

Our project delivers a **web-based local work marketplace** where anyone can post small jobs and nearby providers **bid** to win them. By centering discovery on a map and implementing **KYC for all users**, escrowed payments, and transparent reviews, we help posters get trustworthy help quickly and providers find nearby work efficiently. We will work iteratively and incorporate feedback at each stage.

Problem Statement

Informal local labor markets are fragmented and risky. Posters struggle to find trustworthy help at fair prices; providers waste time searching for nearby opportunities and face payment uncertainty.

Target Users

- **Posters**: individuals or small businesses needing short tasks (e.g., repairs, yard work, errands).
- **Providers**: local freelancers/contractors earning income from small jobs.

Project Goals

- Ship an intuitive, map-first marketplace for posting, bidding, and completing jobs.
- Increase trust with **mandatory KYC**, Duo-based 2FA, escrowed payments, and reviews.
- Ensure privacy and safety with approximate locations pre-acceptance and phone masking post-acceptance.
- Build on a pragmatic stack: **React + SCSS (Firebase Hosting), Node.js + Express, Firebase Firestore**.

3 High-Level Features

1. Core Marketplace

- Post jobs with photos, budget (fixed or open), category, and date.
- Bid on jobs with amount, note, and ETA; accept/decline; anti-sniping window (optional).
- Escrowed payments (hold $\rightarrow$ capture on completion; refund/dispute flow).
- Ratings and reviews on completion.

2. Safety, Identity & Trust

- **KYC for all users** (document + selfie) before posting/bidding.
- Duo Security 2FA for login and sensitive actions.
- Neighborhood Safety Score (0–100) informing friction: tips, daylight defaults, verified-only, or manual review.
- Background checks for providers (where permitted; consent required).
- Phone masking after acceptance; report/block; moderation queue.

3. Maps & Discovery

- Map-based job discovery: clustering, radius filtering, category/budget/date filters.
- Address autocomplete and validation; forward/reverse geocoding.
- Optional time-to-site sorting (distance matrix) in later iterations.

4. Reporting & Analytics

- Basic dashboards: job throughput, bid velocity, completion rates.
- Admin tools: dispute intake, category management, policy/config tuning.

4 Iteration Plan

Long-Term Timeline (Sept 2024 – Mar 30, 2025)

- **Sept – Oct 2024: Foundations**
  Repo setup, CI/CD scaffolding, Auth + Duo 2FA, Firestore schemas, job CRUD, SCSS setup, design wireframes.
- **Nov – Dec 2024: Requirements & Map-first Features**
  Finalize user stories, integrate Google Maps (JS, geocoding, Places autocomplete), stub Safety Score, messaging skeleton.
- **Jan 2025: Bidding Loop**
  Bids, accept/decline flows, notifications, escrow flow with Stripe Connect.
- **Feb 2025: Trust & Compliance**
  KYC mandatory (Stripe Identity), provider background checks (Checkr), admin moderation tools.
- **Mar 2025: Polish & Submission**
  Reviews and ratings, refined Safety Score, accessibility/mobile audit, final testing and docs.
  Deliverables due: Sunday, 30 March 2025, 11:59 PM.

5 Tech Stack (MVP)

- **Frontend:** React + SCSS, Firebase Hosting.
- **Backend:** Node.js + Express (on Firebase Functions/Cloud Run).
- **Database:** Firebase Firestore (NoSQL, real-time).
- **Storage:** Firebase Cloud Storage for job photos/docs.
- **Auth:** Firebase Auth (OAuth, email+password) + Duo 2FA.
- **Payments:** Stripe Connect (escrow), Stripe Identity (KYC).
- **Maps/Geo:** Google Maps JS, Places, Geocoding, Distance Matrix.
- **Comms:** Twilio SMS + Proxy, Firebase Cloud Messaging.
- **Observability:** Sentry, OpenTelemetry, product analytics via PostHog.

6 High-Level User Stories

- As a **new user**, I complete KYC and Duo 2FA so I can safely use the platform.
- As a **poster**, I create a job with photos, budget, and a map location so providers can bid.
- As a **provider**, I browse nearby jobs on a map, filter them, and place bids with notes and ETAs.
- As a **poster**, I compare bids, accept one, fund escrow, and chat with the winner.
- As a **provider**, I complete work and get paid automatically upon confirmation.
- As **either party**, I leave a rating/review and can report issues for moderation.

7 Planning: Task Allocation (Agile Rotation)

Principle: Everyone works across frontend, backend, and Firebase. Pairs rotate weekly in a round-robin so each member pairs with every other member during the term.

Process

- Sprint cadence: 1-week sprints with backlog refinement and sprint review each Friday.
- Daily stand-up: 10 minutes; blockers captured as GitHub issues.
- Rotation: Two pairs per week; cycle repeats every 3 weeks.
- Scrum Master: rotates weekly (Tyler $\rightarrow$ Mani $\rightarrow$ Yanness $\rightarrow$ Alaister, then repeat).
- Quality gates: PR requires one reviewer outside the pair, passing tests, lint compliance.

Appendix: API Inventory

Area	**APIs / Notes**
Maps	Google Maps JS; Places Autocomplete/Details; Geocoding; Distance Matrix
Auth	Firebase Auth; Duo 2FA; Stripe Identity (KYC)
Payments	Stripe Connect (escrow), Webhooks, Radar
Notifications	Firebase Cloud Messaging; Twilio SMS + Proxy
Storage	Firebase Cloud Storage
Analytics/Ops	PostHog; Sentry; OpenTelemetry

Appendix: Firestore Schema (Indicative)

Listing 1: Firestore Collections & Example Documents

```
/users/{userId}
{
  name: string,
  email: string,
  phone: string,
  avatarUrl: string,
  isProvider: boolean,
  kycStatus: "pending" | "verified" | "failed",
  duoEnabled: boolean,
  createdAt: timestamp,
  updatedAt: timestamp
}


/jobs/{jobId}
{
  posterId: string (ref: users/{userId}),
  title: string,
  description: string,
  category: string,
  budgetType: "fixed" | "open",
  budgetAmount: number,
  location: { lat: number, lng: number, address: string },
  desiredDate: timestamp,
  status: "open" | "awarded" | "in_progress" | "completed" | "cancelled",
  createdAt: timestamp,
  updatedAt: timestamp
}


/bids/{bidId}
{
  jobId: string (ref: jobs/{jobId}),
  providerId: string (ref: users/{userId}),
  amount: number,
  note: string,
  etaHours: number,
  status: "active" | "declined" | "accepted" | "cancelled",
  createdAt: timestamp
}


/messages/{messageId}
{
  jobId: string (ref: jobs/{jobId}),
  senderId: string (ref: users/{userId}),
  body: string,
  attachmentUrl: string,
  createdAt: timestamp
}
```

```
/reviews/{reviewId}
{
  jobId: string (ref: jobs/{jobId}),
  raterId: string,
  rateeId: string,
  rating: number (1-5),
  comment: string,
  createdAt: timestamp
}
```