OpenBid: Map-first Bidding Marketplace

DIGT 2107 — Project Iteration 1.1: Initial Vision & Planning

Team 1: Tyler, Mani, Yanness, Alaister

September 18, 2025

Course: DIGT2107 – Fall/Winter Term 2025-2026 | **Instructor:** Dr. May Haidar

Project Link: https://github.com/tjung-git/OpenBid-Map-first-Bidding-Markplace

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 services in the Toronto GTA such as home repairs, moving help, yardwork, and small trades are fragmented across classifieds and social platforms, leading to high search costs, price opacity, and safety concerns. Consumers lack verified identity signals and protected payment flows, while providers face lead uncertainty, travel inefficiency across dispersed neighborhoods, and payment risk. GTA-specific factors-high population density with strong neighborhood heterogeneity, significant travel-time variability, and documented fraud/scam exposure-exacerbate these frictions. A map-first marketplace with mandatory identity verification (KYC), escrowed payments, and on-platform messaging/reviews can reduce search and trust costs, enable fair price discovery, and improve job completion outcomes in the GTA.

Target Users

- **Job Posters**: individuals or small businesses needing short tasks (e.g., repairs, yard work, errands).
- **Job Bidders**: KYC verified individuals who can earn income from small jobs.
- **Job Browser**: anyone can visit the website and browse 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 → capture on completion; refund/dispute flow).
- Ratings and reviews on completion.

2. In-App Messaging & Negotiation

- Secure real-time chat between posters and providers linked to each job.
- Chat is available from the moment a bid is placed; posters and bidders can negotiate price, clarify scope, and attach images/documents.
- Messages are logged in Firestore under '/messages/messageId' per job, enabling dispute resolution and moderation.
- Notifications (push + SMS proxy fallback) alert users of new messages.
- For safety, personal phone/email contact info is blocked or masked until a job is awarded.

3. 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 and exact address masking after acceptance; report/block; moderation queue.

4. 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.

5. Customer Support & Contact

- In-app "Contact Us" form available under profile/settings.
- Support tickets stored in Firestore and routed to admin dashboard.
- Automatic email confirmation sent to user upon submission.

- Options for common categories: billing, identity verification, job disputes, technical issues.
- Escalation path: tickets auto-flagged for urgent review if related to payments or safety.
- Future enhancement: live chat or chatbot triage.

6. Reporting & Analytics

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

4 Iteration Plan

Note on dates The course header says **Fall 2025**. Where the original draft referenced 2024–2025, we now organize by **iterations (2-week groupings)** per the instructor's template. Adjust the "Weeks" to match the actual syllabus calendar if needed.

Iteration 1 (Weeks 1–2): Initial Vision & Planning

Objectives: Align on scope, ship the vision doc, stand up repo/process, and define initial feature categories.
Deliverables (per template):

- **Project Vision Document:** this PDF (Sections 1–3).
- **Initial feature categories & high-level functionality:** see §3 *High-Level Features* and §6 *High-Level User Stories*.
- **Basic repository setup:** see §5 *Repository Setup & Branching Model* (includes link: https://github.com/tjung-git/OpenBid-Map-first-Bidding-Markplace).

Planned work (Weeks 1–2):

- Create GitHub org/repo, add branch protections, CI, and deploy previews.
- Establish Firestore schema draft (*Appendix: Firestore Schema*).
- Draft wireframes for map-first discovery and core job flows.
- Define acceptance criteria at the epic level (to be decomposed next iteration).

Iteration 2 (Weeks 3–4): Foundations & Map-first Basics

Goals: Working skeleton of app + basic map and jobs.
Scope:

- Auth via Firebase Auth; Duo 2FA scaffold (gate kept behind feature flag).
- Google Maps JS integration (map, markers, clustering stub).
- Job posting CRUD (create/view/edit/delete), file upload to Cloud Storage.
- CI pipelines (build, lint, test) blocking on PR; CD to Firebase Hosting.

Iteration 3 (Weeks 5–6): Bidding Loop & Messaging (MVP paths)

Goals: End-to-end "post → bid → award" flow; basic chat.
Scope:

- Bids: create, list, accept/decline; notifications for bid events.
- In-app messaging linked to job/bid; masking of contact info prior to award.
- Stripe Connect sandbox escrow flow (hold on award; manual capture stub).
- Map filters: category, budget type/range; address autocomplete.

Iteration 4 (Weeks 7–8): Trust & Compliance

Goals: Raise trust bar before broader testing.
Scope:

- Mandatory KYC via Stripe Identity (pre-posting & pre-bidding gates).
- Background checks (Checkr) with consent; admin moderation queue.
- Neighborhood Safety Score stub → v1 thresholds and UI nudges.

Iteration 5 (Weeks 9–10): Payments, Reviews & Support

Goals: Close the loop on jobs and strengthen support.
Scope:

- Escrow capture on completion; refund/dispute path stub; webhooks hardening.
- Ratings & reviews on completion; profile reputation display.
- Support "Contact Us" form → Firestore tickets + email confirmation; admin responses.

Iteration 6 (Weeks 11–12): Polish, Accessibility & Final QA

Goals: Quality and submission readiness.
Scope:

- Accessibility/mobile audits; performance passes; error telemetry (Sentry).
- E2E tests (Cypress) for critical flows; unit/integration coverage (Jest).
- Documentation pass (runbooks, architecture overview, API inventory).

Milestone Deliverables (rolling)

- **Iteration 1:** Vision doc + feature categories + repo set-up (this submission).
- **Iteration 3:** Demo of post→bid→award with chat; map filters.
- **Iteration 5:** Payments escrow loop + reviews + support tickets.
- **Iteration 6:** Final QA, documentation, and submission package.

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 Repository Setup & Branching Model

Repository & Links

GitHub repository: https://github.com/tjung-git/OpenBid-Map-first-Bidding-Markplace

Setup Process

- **Default branch:** `main` (protected). Require 1+ approving review, status checks (build, lint, tests), and up-to-date branch before merge.
- **CI/CD:** GitHub Actions.
  - *frontend*: install, lint, unit tests (Jest), build; preview deploy on PR.
  - *backend*: install, lint, unit tests; deploy to Firebase Functions/Cloud Run on tagged release.
- **Environments:** `dev` (PR previews), `staging` (protected), `prod` (tagged releases).
- **Secrets:** Stored in GitHub Encrypted Secrets (Firebase, Stripe, Duo, Twilio, Sentry keys).
- **Issue tracking:** GitHub Issues + Projects board (Kanban: Backlog, In Progress, Review, Done). Labels: `feature`, `bug`, `chore`, `docs`, `security`, `good-first-issue`.
- **PR hygiene:** PR template (scope, screenshots, tests), CODEOWNERS for review routing, conventional commits.

Branching Strategy

We use **trunk-based development** with short-lived branches and release tagging.

- **Branch types & naming:**
  - Feature: `feature/<area>-<short-desc>` (e.g., `feature/maps-clustering`)
  - Fix: `fix/<area>-<issue#>` (e.g., `fix/payments-escrow-123`)
  - Chore: `chore/<topic>` (e.g., `chore/ci-lint`)

  - Hotfix (prod): `hotfix/<desc>`
- **Flow:** branch from `main` → PR → checks + review → squash-merge to `main`.
- **Release tags:** semantic versions on `main` (e.g., `v0.1.0`) trigger staging→prod deploy.
- **Long-running work:** behind feature flags; keep branches small to avoid drift.

7 High-Level User Stories

Job Posters

- As a **poster**, I want to complete KYC and enable Duo 2FA so I can use the platform securely.
- As a **poster**, I want to create a job with photos, budget options, and a map location so providers can find and bid on it.
- As a **poster**, I want to receive and compare multiple bids, so I can choose the best provider for the job.
- As a **poster**, I want to chat with bidders for clarification and negotiation before accepting one.
- As a **poster**, I want to fund escrow when I accept a provider, so the payment is secured until work is completed.
- As a **poster**, I want to confirm when the job is finished, so the provider is paid automatically from escrow.
- As a **poster**, I want to leave ratings and reviews after the job ends, so I can help build provider reputation.
- As a **poster**, I want to contact customer support if I have disputes or technical problems.

Job Bidders (Providers)

- As a **bidder**, I want to complete KYC verification before bidding so my identity is trusted on the platform.
- As a **bidder**, I want to browse nearby jobs on a map with filters (category, date, budget) so I can find relevant opportunities.
- As a **bidder**, I want to place bids with notes and estimated time of arrival, so posters understand my proposal.
- As a **bidder**, I want to chat securely with posters inside the app to negotiate details or clarify requirements.
- As a **bidder**, I want to receive real-time notifications when a poster responds to my bid or message.
- As a **bidder**, I want to start work once I am awarded a job and see that payment is secured in escrow.
- As a **bidder**, I want to get paid automatically from escrow when the poster confirms completion.
- As a **bidder**, I want to build a reputation with ratings and reviews from past jobs, so I can win future bids more easily.

- As a **bidder**, I want to contact support if I experience fraud, safety issues, or payment disputes.

Job Browsers (Not KYC-Verified Users)

- As a **browser**, I want to explore posted jobs on a map in read-only mode, so I can understand the platform's services before committing.
- As a **browser**, I want to view limited job details (e.g., title, category, approximate location, budget range) without registering, so I see the platform's value.
- As a **browser**, I want to be prompted to create an account if I try to post a job or place a bid, so I know registration is required.
- As a **newly registered user**, I want clear instructions that I must complete KYC verification and enable Duo 2FA before being allowed to post jobs or bid, so I understand the trust policy.
- As a **browser**, I want access to support or FAQ pages, so I can ask questions or learn more before verifying my account.

8 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
}
```