DIGT2107: Practice of Software Development

Project Iteration 2: Detailed User Stories, Requirements, and Initial Prototype

OpenBid

Tyler, Mani, Yanness, Alaister

October 5, 2025

Course: DIGT2107 – Fall Term 2025 | **Instructor:** Dr. May Haidar | **Team: #1**

Contents

1 Introduction

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

Document Overview This document outlines the deliverables for Iteration 2 of OpenBid: detailed user stories, functional and non-functional requirements, and an initial prototype plan. The aim is to establish a solid foundation for subsequent iterations by clarifying what the system should do and validating the core experience with a working prototype.

2 Functional and Non-Functional Requirements

2.1 Functional Requirements

1. **Account & Identity:** Users shall be able to sign up and sign in via OAuth (Google/Apple) and email+password. Mandatory KYC must be completed before posting or bidding.
2. **Two-Factor:** Users shall confirm sign-in with Duo 2FA for sensitive actions (login, payouts).
3. **Job Posting:** Posters shall create, update, and delete jobs with title, description, photos, budget (fixed/open), category, date, and location (map pin / address).
4. **Discovery:** Providers shall browse jobs on a map and list, filter by radius, category, budget range, and date.
5. **Bidding:** Providers shall place, edit, and withdraw bids (amount, note, ETA). Posters shall accept a winning bid.
6. **Payments (Escrow):** Posters shall fund an escrow hold on accept; on completion, funds shall capture and pay out to the provider (Stripe Connect).
7. **Messaging:** Posters and providers shall exchange messages within the job thread, with attachments and report/block controls.
8. **Reviews:** Both parties shall leave ratings and text reviews after completion.
9. **Safety Score:** The system shall compute a location safety score (0–100) and apply graduated friction (tips, daylight default, verified-only, or manual review).
10. **Admin:** Admins shall review reports, manage categories, and handle disputes via a basic dashboard.

2.2 Non-Functional Requirements

- **Performance:** Map/list browse and search shall return results in $\leq$ 500 ms p95 for target metro; prototype may use simplified filters.
- **Usability:** Mobile-first, accessible (WCAG 2.1 AA where practical); clear copy for KYC/2FA and escrow steps.
- **Security:** All traffic over TLS; short-lived JWTs; server-side validation; reCAPTCHA on signup/post/bid.

- **Scalability:** Stateless API (Node/Express) behind Firebase/Cloud Run; Firestore as primary data store; storage via Firebase Cloud Storage.
- **Reliability:** Error tracking via Sentry; observability with OpenTelemetry; automated CI checks on PRs.
- **Privacy:** Approximate location shown pre-accept; exact address released to the accepted provider only; phone masking post-accept.

3 Use Cases

3.1 Use Case Diagram

Include your UML use case diagram image here (PNG/PDF).

3.2 Use Case Descriptions

UC-1: Manage Jobs (Poster) **Actors:** Poster
Description: Create, update, and delete job postings with required details.
Preconditions: User is authenticated and KYC verified.
Main Flow:

1. Poster selects "New Job".
2. System prompts for title, description, category, budget type/amount, date, and location.
3. Poster submits; system validates and saves to database; job appears on map/list.
4. Poster may edit or delete until a bid is accepted.

Postconditions: Job is persisted and visible to eligible providers.

UC-2: Browse & Bid (Provider) **Actors:** Provider
Description: Discover nearby jobs and place bids.
Preconditions: User is authenticated; KYC verified.
Main Flow:

1. Provider opens map view; system shows jobs within radius with filters.
2. Provider opens a job, reviews details, and selects "Place Bid".
3. Provider enters amount, note, and ETA; system validates and saves bid.
4. Provider can edit/withdraw bid before award.

Postconditions: Bid saved and visible to the poster.

UC-3: Award & Escrow (Poster) **Actors:** Poster
Description: Compare bids, accept a winner, and fund escrow.
Preconditions: Poster is authenticated; job has active bids; poster has a payment method.
Main Flow:

1. Poster reviews bids (amount, ETA, ratings).

use_case_diagram.png

Figure 1: Use Case Diagram: Actors and Main Use Cases

2. Poster accepts a bid; system creates a Stripe PaymentIntent (hold).
3. On success, system marks job as awarded and opens thread with winner.

Postconditions: Funds on hold; winner can proceed.

UC-4: Complete & Payout **Actors:** Provider, Poster
Description: Provider marks work done; poster confirms; funds captured and paid out.
Preconditions: Job awarded; escrow hold succeeded.
Main Flow:

1. Provider marks job completed; system notifies poster.
2. Poster confirms completion (or opens dispute).
3. On confirmation, system captures funds and pays out via Stripe Connect.

Postconditions: Payment captured; both parties can review each other.

4 Detailed User Stories

Category: Job & Bid Management

US-1: Post a New Job (High, 3 pts) As a **poster**, I want to add a job with title, description, photos, budget, date, and a map location so providers can bid.

- **Acceptance:** Required fields validated; job saved; appears on map/list; poster can edit/delete until award.

US-2: Browse Nearby Jobs (High, 3 pts) As a **provider**, I want to see jobs near me with filters so I can quickly find relevant work.

- **Acceptance:** Map/list render; radius/category/budget/date filters; open job details view.

US-3: Place a Bid (High, 5 pts) As a **provider**, I want to place a bid with amount, note, and ETA so the poster can evaluate me.

- **Acceptance:** Bid saved; poster notified; provider can edit/withdraw prior to award.

US-4: Accept a Bid & Fund Escrow (High, 5 pts) As a **poster**, I want to accept a bid and put funds on hold so payment is guaranteed on completion.

- **Acceptance:** Stripe hold succeeds; job moves to `awarded`; chat thread opens with winner.

Category: Trust & Safety

US-5: Complete KYC (High, 3 pts) As a **new user**, I want to verify my identity (document + selfie) so I can post or bid safely.

- **Acceptance:** Stripe Identity flow completes; app shows verified badge; features unlock.

US-6: Duo 2FA (Medium, 2 pts) As a **user**, I want Duo phone confirmation so my account is protected during sign-in and payouts.

- **Acceptance:** 2FA challenge required; recovery path documented.

US-7: Safety Score Guidance (Medium, 3 pts) As a **poster**, I want safety guidance at post-time so I can choose safer options (e.g., daylight/public meetup).

- **Acceptance:** Score shown with tips; low-score flows add friction or require verification.

Category: Comms & Reviews

US-8: Job Thread Messaging (Medium, 3 pts) As **either party**, I want a per-job chat so we can coordinate details.

- **Acceptance:** Send/receive text; attachments; report/block; notifications.

US-9: Leave Reviews (Medium, 2 pts) As **either party**, I want to rate and review after completion so future users can trust matches.

- **Acceptance:** 1–5 stars plus comment; appears on profile; single review per party per job.

5 Initial Prototype Guidelines

5.1 Prototype Features (Iteration 2 scope)

- Minimal UI (React + SCSS) with routes: Home (Map/List), Job Details, Post Job, Bids.
- Core flows: create job, browse jobs, place bid, accept bid *(escrow call can be stubbed if needed)*.
- Basic navigation and client-side validation.
- Firestore setup (collections: users, jobs, bids, messages); mock/safe seed data for demo.

5.2 Development Approach

- **Start Small:** Implement US-1, US-2, US-3 first; wire KYC screen stub.
- **Iterate Quickly:** Short PRs; CI checks; deploy preview to Firebase Hosting.
- **Modularity:** Split UI into reusable components; keep server routes RESTful.
- **Document Progress:** Track tasks in GitHub Projects; reference commits in issues.

5.3 Task Completion & Collaboration

- Break stories into tasks with estimates; label by feature and owner.
- Use a shared task board; commit frequently with meaningful messages.
- Pair programming rotation (round robin); driver/navigator swap daily; 1 reviewer outside the pair.

6 Documentation Updates

6.1 System Architecture (Prototype)

Frontend: React + SCSS (Firebase Hosting)
Backend: Node.js + Express (Firebase Functions or Cloud Run)
Database: Firebase Firestore (NoSQL, real-time)
Storage: Firebase Cloud Storage
Auth: Firebase Auth (OAuth, email+password) + Duo 2FA; KYC via Stripe Identity
Payments: Stripe Connect (escrow)
Maps: Google Maps JS, Places, Geocoding, Distance Matrix

6.2 Planning Document and Log

- **Changes from Iteration 1.1:** Committed to Firebase stack; clarified escrow flow; added explicit safety score gating.
- **Plan Delta:** Pulled reviews into Iteration 2 tail; background checks remain later.

7 Iteration Plan and Backlog

7.1 Iteration Plan (Weeks 3–5)

- **Week 3:** Finalize functional requirements; implement US-1 (post job) and base map/list; Firestore rules draft.
- **Week 4:** Implement US-2 (browse) and US-3 (bid); start US-4 (accept & escrow stub); basic messaging skeleton.
- **Week 5:** Review feedback; refine UX; stabilize prototype; add US-8 (chat MVP) and US-9 (reviews, stub).

7.2 Updated Backlog (Top)

ID	Story	Priority	Pts
US-1	Post a new job	High	3
US-2	Browse nearby jobs	High	3
US-3	Place a bid	High	5
US-4	Accept bid & escrow (stub/real)	High	5
US-5	Complete KYC	High	3
US-8	Job thread messaging (MVP)	Medium	3
US-9	Leave reviews	Medium	2
US-6	Duo 2FA	Medium	2
US-7	Safety score guidance	Medium	3

8 Submission Guidelines

- **GitHub Repository:** Push prototype code and documentation; tag release as `ITR1.2`.
- **eClass Submission:** Submit one PDF containing: Requirements; Use Case Diagram & Descriptions; Detailed User Stories; Initial Prototype Overview; Updated Plan and Backlog.

Appendix A: API Inventory (Condensed)

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

Appendix B: Firestore Schema (Indicative)

Listing 1: Firestore Collections & Example Documents

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

/jobs/{jobId}
{
  "posterId": "users/{userId}",
  "title": "string",
  "description": "string",
  "category": "string",
  "budgetType": "fixed" | "open",
  "budgetAmount": 120.50,
  "location": {"lat": 43.6426, "lng": -79.3871, "address": "string" },
  "desiredDate": "timestamp",
  "status": "open" | "awarded" | "in_progress" | "completed" | "cancelled",
  "createdAt": "timestamp",
  "updatedAt": "timestamp"
}

/bids/{bidId}
{
  "jobId": "jobs/{jobId}",
  "providerId": "users/{userId}",
  "amount": 95.00,
  "note": "string",
  "etaHours": 6,
  "status": "active" | "declined" | "accepted" | "cancelled",
  "createdAt": "timestamp"
}

/messages/{messageId}
{
  "jobId": "jobs/{jobId}",
  "senderId": "users/{userId}",
  "body": "string",
  "attachmentUrl": "string",
  "createdAt": "timestamp"
}

/reviews/{reviewId}
{
  "jobId": "jobs/{jobId}",
  "raterId": "users/{userId}",
  "rateeId": "users/{userId}",
```

```
  "rating": 5,
  "comment": "string",
  "createdAt": "timestamp"
}
```