DIGT2107: Practice of Software Development

Project Iteration 3: Testing and Initial Development

OpenBid

Team 1: Tyler Mani Yanness Alaister

Due: November 2, 2025

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

1 Introduction

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

Document Overview This document outlines the deliverables for Iteration 3, focusing on the initial development phase. It shows how our *test cases* map to the *functional and non-functional requirements* from earlier iterations, and it reports the current working prototype. Goals: implement core functionality tied to high-priority stories, develop comprehensive unit tests, and deliver a working prototype demonstrating traceability.

2 Iteration Goals and Objectives

- Implement core flows: **auth + Duo 2FA**, **KYC gate** (stubbed via Stripe Identity sandbox token), **post a job**, **browse on map**, **place a bid**.
- Develop and run unit tests mapped to requirements (frontend component tests and backend handler tests).
- Deliver a prototype demonstrating requirement ↔ test case links.

3 Requirement to Test Case Traceability

Tables below map each requirement to the test cases that validate it. Coverage status will be updated as testing completes.

Functional Requirements (FR)

Req ID	Requirement Description	Test Case IDs	Coverage
FR-001	System shall allow authenticated users to **post jobs** with title, description, budget, photos, and location.	TC-001, TC-002	Partially Covered
FR-002	System shall allow providers to **browse jobs on a map** with basic filters (radius, category).	TC-003a-d, TC-004	Partially Covered
FR-003	System shall allow providers to **place bids** on open jobs; posters can view and accept a bid.	TC-005, TC-006	Partially Covered
FR-004	System shall enforce **KYC for all users** before posting or bidding.	TC-007	Partially Covered
FR-005	System shall support **user authentication** with **Duo 2FA** for sensitive actions.	TC-008	Covered
FR-006	System shall provide **in-thread messaging** per job after acceptance.	TC-009	Planned
FR-007	System shall record **ratings and reviews** after job completion.	TC-010	Planned
FR-008	System must create Firebase accounts and keep users out until they click the email verification link.	TC-014a–d	Covered

Req ID	Requirement Description	Test Case IDs	Coverage
FR-009	System must let a signed-in person switch between bidder and contractor and keep their requirement flags accurate.	TC-015a–b	Covered
FR-010	System must save the session in local storage and log the user out after two minutes with no activity.	TC-016a–c	Covered
FR-011	Only KYC-approved contractors can create, edit, or delete their own jobs in Firestore.	TC-017a–i	Covered
FR-012	System must block self-bids and lock both the job and bids once a contractor accepts an offer.	TC-018a–b	Covered

Non-Functional Requirements (NFR)

Req ID	Requirement Description	Test Case IDs	Coverage
NFR-001	**Performance:** Search/browse should return results within 800 ms P95 for a metro with 5k open jobs (stub data).	TC-011	Planned
NFR-002	**Security:** Only KYC-verified users can hit write endpoints for jobs/bids.	TC-007, TC-012	Partially Covered
NFR-003	**Usability:** Map view and list view maintain accessible contrast and keyboard navigation for core actions.	TC-013	Planned

Notes

- Each requirement is validated by one or more test cases. Gaps are flagged as *Planned*.
- NFR coverage status is informative only (not required for grading), but we track it to guide future work.

4 Detailed Test Case Descriptions

For brevity we include the highest-priority cases now; the full catalog will live in the repo under `tests/`.

TC-001

Title: Post Job - Valid Inputs
Requirement: FR-001
Preconditions: User is logged in, KYC status = verified.
Steps:

1. Navigate to `/new-job`.
2. Enter valid title, description, budget, and pick a map location (Google Maps widget).
3. Upload a photo (sample file).

4. Click `Post`.

Expected Result: Job document is created in Firestore; UI redirects to Job Detail.
Actual Result: To be filled after execution.
Status: Pending **Priority:** High

TC-002

Title: Post Job - Validation Errors
Requirement: FR-001
Preconditions: Logged in, KYC verified.
Steps:

1. Navigate to `/new-job`.
2. Leave title empty; click `Post`.

Expected Result: Client-side validation shows error; no write occurs.
Status: Pending **Priority:** High

TC-003a

Test Case ID	TC-003a
Title	The haversineFormulakm function accurately calcuates distance between 2 coordinates.
Requirement	FR-002
Preconditions	Valid coordinates are input.
Steps	1. Coordinates are input into the function. 2. Function calculates the distance between the coordinates. 3. Function returns the distance between the coordinates.
Expected Result	Returns distance in km of the 2 coordinates, returns infinity when there is invalid input.
Actual Result	Passed - Returns distance accurately within a specific tolerance. Coordinates tested are up to 250km apart.
Status	Passed
Priority	High

TC-003b

Test Case ID	TC-003b
Title	The isValidCoords formula accurately determines whether the given inputs are valid latitude and longitude values.
Requirement	FR-002
Preconditions	None.
Steps	1. Coordinates are input into the function. 2. Function checks input for validity then checks whether the input values are valid coordinates. 3. Function returns whether the coordinates are valid.
Expected Result	Returns true or false to depending on whether the input is valid coordinates.
Actual Result	Passed - accurately surmises whether the input coordinates are valid.
Status	Passed
Priority	High

TC-003c

Test Case ID	TC-003c
Title	The degToRad function accurately converts a number in degrees to radians.
Requirement	FR-002
Preconditions	Input should be a number.
Steps	1. a number in degrees is input into the function. 2. Function checks input for validity then returns the input value in radians.
Expected Result	Returns the input number converted into radians if the input is valid, returns infinity otherwise.
Actual Result	Passed - accurately converts a valid input to radians and returns infinity otherwise.
Status	Passed
Priority	High

TC-003d

Test Case ID	TC-003c
Title	The createMap function loads a google map. (uses api mocking)
Requirement	FR-002
Preconditions	The lat, lng, ref, and apikey inputs are valid. The lat and lng and tested for validness before the function call and the logic outside the function affirms that apiKey and ref are valid.
Steps	1. The google maps api loader is initialized. 2. The markers array input is filtered for valid markers or is set to an empty array if the markers input is bad. 3. The google maps api loader is used to load the api. 4. A new google map is created and valid markers are atached to the map.
Expected Result	Creates a map with valid markers attached to it.
Actual Result	Passed - creates a map with valid markers attached to it, invalid markers are discarded and a bad markers input is gracefully handled.
Status	Passed
Priority	High

TC-004

Title: Map Browse - Category Filter
Requirement: FR-002
Expected Result: Only jobs of selected category are shown.
Status: Pending **Priority:** Medium

TC-005

Title: Place Bid - Valid
Requirement: FR-003
Preconditions: Provider is logged in, KYC verified, job is open.
Expected Result: Bid document created; poster sees bid in Job Detail.
Status: Pending **Priority:** High

TC-006

Title: Accept Bid - Poster Flow
Requirement: FR-003
Expected Result: Job status transitions to `awarded`; winning bid marked `accepted`.
Status: Pending **Priority:** High

TC-007

Title: KYC Gate - Block Unverified Writes
Requirement: FR-004, NFR-002
Preconditions: User KYC status = pending.
Expected Result: Attempts to post job or bid are rejected by Firestore rules/API; UI shows

KYC required.
Status: Pending **Priority:** High

TC-008

Title: Duo 2FA Challenge on Login
Requirement: FR-005
Preconditions: User exists, Duo enabled.
Steps:

1. POST `/api/auth/login` with valid credentials.
2. Follow `mfa.startUrl` to begin Duo: GET `/api/auth/duo/start?state=...`.
3. Simulate Duo callback: GET `/api/auth/duo/callback?state=...&code=allow-123`.
4. Finalize: POST `/api/auth/duo/finalize` with one-time code.

Expected Result: Login returns 202 with Duo start URL; callback redirects to `/login/finish?code=...`; finalize returns session JSON.
Actual Result: Passed – Covered by Jest `tests/auth.duo.test.js`.
Status: Passed **Priority:** Medium

TC-009

Title: Messaging After Acceptance
Requirement: FR-006
Expected Result: Parties can exchange messages in job thread; messages persist in Firestore.
Status: Planned **Priority:** Medium

TC-010

Title: Submit Review on Completion
Requirement: FR-007
Expected Result: Review saved and visible on profile; duplicate review blocked.
Status: Planned **Priority:** Medium

TC-011

Title: Performance P95 - Map Query
Requirement: NFR-001
Expected Result: P95 end-to-end from filter change to results render $\leq 800\,\text{ms}$ on stub dataset.
Status: Planned **Priority:** Low

TC-012

Title: Ruleset Audit - No Write Without Claims
Requirement: NFR-002
Expected Result: Firestore rules reject writes missing `kycVerified == true`.
Status: Pending **Priority:** High

TC-013

Title: Accessibility - Keyboard Nav on Map/List
Requirement: NFR-003
Expected Result: Tabbing reaches filters, list items, and primary actions; visible focus outlines present.
Status: Planned **Priority:** Low

TC-014a

Test Case ID	TC-014a
Title	Firebase Signup - Provision Account and Send Verification Email
Requirement	FR-008
Preconditions	Email is brand new inside the Firebase project; Identity Toolkit is online.
Steps	1. Send POST `/api/auth/signup` with first/last name, valid email, password, confirmPassword. 2. Let Firebase create the account and queue the verification email. 3. Read the JSON response from our API.
Expected Result	HTTP 201 with sanitized user data, email + KYC both `pending`, and Firebase verification email requested.
Actual Result	Passed – Confirmed in `server/src/routes/__tests__/auth.integration.real.routes.test.js`.
Status	Passed
Priority	High

TC-014b

Test Case ID	TC-014b
Title	Firebase Signup - Reject Invalid Credentials
Requirement	FR-008
Preconditions	Same Firebase project as TC-014a; email still unused.
Steps	1. POST `/api/auth/signup` with a broken email or a password shorter than eight characters. 2. Watch the server reject the payload before any Firebase call.
Expected Result	HTTP 400 with a clear error; Firebase signup helper never runs.
Actual Result	Passed – Shown in the same Jest suite for real adapters.
Status	Passed
Priority	High

TC-014c

Test Case ID	TC-014c
Title	Firebase Login - Verified User Receives Session Tokens
Requirement	FR-008
Preconditions	User already exists in Firebase with both email and KYC marked `verified`.
Steps	1. POST `/api/auth/login` with the correct email and password. 2. Inspect the response for session + requirement flags.
Expected Result	HTTP 200 with sanitized user, Firebase ID + refresh tokens, and `requirements = {emailVerified: true, kycVerified: true}`.
Actual Result	Passed – Assertions live in `auth.integration.real.routes.test.js`.
Status	Passed
Priority	High

TC-014d

Test Case ID	TC-014d
Title	Firebase Login - Reject Wrong Passwords
Requirement	FR-008
Preconditions	User exists in Firebase; test double forces Identity Toolkit to return `INVALID_PASSWORD`.
Steps	1. POST `/api/auth/login` with the wrong password. 2. Capture HTTP status + body.
Expected Result	HTTP 401 with `{error: "invalid credentials"}`; no session issued.
Actual Result	Passed – Covered in `auth.integration.real.routes.test.js`.
Status	Passed
Priority	High

TC-015a

Test Case ID	TC-015a
Title	Role Switch - Authorization Required
Requirement	FR-009
Preconditions	User is signed in and has a valid Firebase ID token.
Steps	1. PATCH `/api/auth/role` without the Authorization header. 2. Repeat with `Authorization: Bearer <idToken>` and payload `{role: "contractor"}`.
Expected Result	Unauthorized request returns 401; authorized request returns 200 with updated `userType` and refreshed requirement flags.
Actual Result	Passed – Exercised via Jest role-switch integration test.
Status	Passed
Priority	Medium

TC-015b

Test Case ID	TC-015b
Title	Auth Me - Return Sanitized User Context
Requirement	FR-009
Preconditions	Contractor user plus valid Firebase ID token.
Steps	1. GET `/api/auth/me` without Authorization header and confirm 401. 2. Repeat with Authorization header; inspect payload for sanitized user fields.
Expected Result	Unauthorized call blocked; authorized call returns sanitized `user` payload (no `passwordHash`) plus metadata.
Actual Result	Passed – Documented in `auth.integration.real.routes.test.js`.
Status	Passed
Priority	Medium

TC-016a

Test Case ID	TC-016a
Title	Session Service - Persist and Notify Subscribers
Requirement	FR-010
Preconditions	Browser environment available (Vitest JSDOM); subscribers registered.
Steps	1. Call `setSession` with a real Firebase-authenticated user payload. 2. Verify `subscribeSession` listener triggered. 3. Inspect `localStorage` for serialized session data.
Expected Result	Session stored under `openbid_session`; only real-user metadata persisted; listeners invoked once.
Actual Result	Passed – Covered by `client/src/__tests__/session.test.js`.
Status	Passed
Priority	Medium

TC-016b

Test Case ID	TC-016b
Title	Session Service - setUser Preserves Requirements
Requirement	FR-010
Preconditions	Existing session with requirement flags stored.
Steps	1. Initialize session via `setSession`. 2. Invoke `setUser` with new user payload and explicit requirements. 3. Query `getRequirements` and `getSession`.
Expected Result	Updated user persisted; requirements reflect provided override (email + KYC flags).
Actual Result	Passed – Validated in `session.test.js`.
Status	Passed
Priority	Medium

TC-016c

Test Case ID	TC-016c
Title	Session Service - Inactivity Monitor Auto Logout
Requirement	FR-010
Preconditions	Session established; Vitest fake timers.
Steps	1. Call `startInactivityMonitor` with spy callback. 2. Advance timers to just before the 2-minute limit; ensure no callback. 3. Advance timers past the limit; ensure callback fires and cleanup works.
Expected Result	Timeout callback executes exactly once after 2 minutes of inactivity, indicating logout.
Actual Result	Passed – Covered by Vitest case in `session.test.js`.
Status	Passed
Priority	Medium

TC-017a

Test Case ID	TC-017a
Title	Job Create - Verified Contractor Path
Requirement	FR-011
Preconditions	Contractor account with `kycStatus = verified` and a valid Firebase ID token.
Steps	1. POST `/api/jobs` with the Authorization header plus title, description, budget, and location. 2. Read the JSON response.
Expected Result	HTTP 200 with a job whose `posterId` matches the contractor and status `open`.
Actual Result	Passed – Verified in `jobs.integration.real.routes.test.js`.
Status	Passed
Priority	High

TC-017b

Test Case ID	TC-017b
Title	Job Create - Bidder Rejected
Requirement	FR-011
Preconditions	Bidder account logged in with Firebase token.
Steps	1. POST `/api/jobs` using the bidder's token and minimal payload.
Expected Result	HTTP 403 with `{error: "contractor_only"}`; no job written.
Actual Result	Passed – Documented in `jobs.integration.real.routes.test.js`.
Status	Passed
Priority	High

TC-017c

Test Case ID	TC-017c
Title	Job Create - KYC Pending Block
Requirement	FR-011
Preconditions	Contractor account exists but `kycStatus = pending`.
Steps	1. POST `/api/jobs` with the contractor's token while KYC is still pending.
Expected Result	HTTP 403 with `{error: "KYC required"}`; job not created.
Actual Result	Passed – See "enforces KYC verification" Jest test.
Status	Passed
Priority	High

TC-017d

Test Case ID	TC-017d
Title	Job Update - Owner Edits Open Job
Requirement	FR-011
Preconditions	Contractor owns an `open` job and is signed in.
Steps	1. PATCH `/api/jobs/:jobId` with a new title and budget. 2. Read response body.
Expected Result	HTTP 200; returned job reflects updated fields; timestamps preserved.
Actual Result	Passed – "updates an open job" Jest test.
Status	Passed
Priority	Medium

TC-017e

Test Case ID	TC-017e
Title	Job Update - Foreign Owner Forbidden
Requirement	FR-011
Preconditions	Two contractors exist; the job belongs to contractor A; contractor B is signed in.
Steps	1. Contractor B PATCHes `/api/jobs/:jobId` owned by contractor A.
Expected Result	HTTP 403 with `{error: "forbidden"}`; job unchanged.
Actual Result	Passed – "forbids editing another contractor's job" Jest test.
Status	Passed
Priority	Medium

TC-017f

Test Case ID	TC-017f
Title	Job Update - Locked Status
Requirement	FR-011
Preconditions	Contractor owns a job already marked `awarded` or `completed`.
Steps	1. Try to PATCH the locked job.
Expected Result	HTTP 409 with `{error: "job_locked"}`.
Actual Result	Passed – "returns job_locked when status is no longer open" Jest test.
Status	Passed
Priority	Medium

TC-017g

Test Case ID	TC-017g
Title	Job Delete - Owner Removes Open Job
Requirement	FR-011
Preconditions	Contractor owns an open job.
Steps	1. DELETE `/api/jobs/:jobId` with the owner's token. 2. Try reading the job from Firestore.
Expected Result	HTTP 204; subsequent read returns null.
Actual Result	Passed – "removes an open job owned by contractor Jane Doe" Jest test.
Status	Passed
Priority	Medium

TC-017h

Test Case ID	TC-017h
Title	Job Delete - Foreign Owner Forbidden
Requirement	FR-011
Preconditions	Job owned by contractor A; contractor B signed in.
Steps	1. Contractor B DELETEs `/api/jobs/:jobId`.
Expected Result	HTTP 403 with `{error: "forbidden"}`; job remains.
Actual Result	Passed – "forbids deleting someone else's job" Jest test.
Status	Passed
Priority	Medium

TC-017i

Test Case ID	TC-017i
Title	Job Delete - Locked Status
Requirement	FR-011
Preconditions	Contractor owns a job that is no longer `open`.
Steps	1. Try to DELETE the locked job.
Expected Result	HTTP 409 with `{error: "job_locked"}`.
Actual Result	Passed – "returns job_locked when job status is not open" Jest test.
Status	Passed
Priority	Medium

TC-018a

Test Case ID	TC-018a
Title	Bid Create - Contractor Cannot Bid Own Job
Requirement	FR-012
Preconditions	Contractor owns an open job and signs in as the bidder.
Steps	1. POST `/api/bids/:jobId` with the contractor's token and a valid amount.
Expected Result	HTTP 403 with `{error: "own_job_bid"}`; no bid written.
Actual Result	Passed – "prevents contractors from bidding on their own jobs" test in `bids.integration.real.routes.test.js`.
Status	Passed
Priority	High

TC-018b

Test Case ID	TC-018b
Title	Bid Accept - Lock Job/Bid Lifecycle
Requirement	FR-012
Preconditions	Contractor owns a job; bidder submits a valid bid; Firebase tokens exist for contractor, bidder, and observer.
Steps	1. Bidder POSTs `/api/bids/:jobId` with a valid amount. 2. Contractor POSTs `/api/bids/:jobId/:bidId/accept`. 3. Contractor tries to PATCH the job; bidder tries to PATCH the accepted bid. 4. Observer GETs `/api/jobs` to check visibility.
Expected Result	Job flips to `awarded` with `awardedBidId`; further edits return `job_locked` / `bidding_closed`; observers no longer see the job.
Actual Result	Passed – "Accepting a bid hides the job from outsiders and locks further edits" test.
Status	Passed
Priority	High

5 Code Repository and Branching Strategy

Repository: `https://github.com/your-org/openbid` *(placeholder)*

Branches:

- `main`: stable releases.
- `develop`: ongoing integration.
- `feature/{slug}`: e.g., `feature/auth`, `feature/map`, `feature/bids`, `feature/kyc`, `feature/tests`.

6 Task Allocation and Timeline

Pair-Programming Rotation (weekly): two pairs; driver/navigator swap daily; Scrum Master rotates weekly (Tyler → Mani → Yanness → Alaister).

Sprint Plan (3 weeks for Iteration 3)

- **Week 1:** Implement auth + Duo 2FA hook, KYC gate stub, Post Job UI/API, initial tests (TC-001, TC-002, TC-007, TC-008).
- **Week 2:** Map browse (pins, filters), Place Bid flow, tests (TC-003, TC-004, TC-005, TC-006, TC-012).
- **Week 3:** Prototype hardening, accessibility pass (keyboard focus), performance harness scaffolding, test documentation (TC-011, TC-013 planning).

Task Breakdown (examples)

- Frontend: NewJob form, MapView, JobList, Bid modal, validation.
- Backend (Express on Firebase Functions/Cloud Run): endpoints for jobs, bids; KYC/Duo middleware.
- Firestore rules: enforce `kycVerified == true` for job/bid writes.
- Testing: Vitest/Jest + React Testing Library for UI; supertest for API; rules-unit-testing for Firestore.
- Docs: update traceability, add test run notes.

7 Prototype Overview

Stack (MVP): React + SCSS (Firebase Hosting), Node.js + Express (Firebase Functions or Cloud Run), Firestore, Google Maps JS, Stripe (Connect/Identity), Duo 2FA.

Implemented in Iteration 3 (target)

- Auth shell with Duo challenge on sensitive action.
- KYC gate (UI + rules) using a sandbox token flow.
- Post Job UI + server write; basic Job Detail.
- Map browse with radius/category filters on seeded data.
- Place Bid basic happy path.

8 Submission Guidelines

- **GitHub:** Push code and documentation; tag release as `ITR2.1`.
- **eClass PDF:** include *Prototype Overview*, *Updated Requirements/Use Cases*, *Test Plan and (initial) Results*, and *Updated Iteration Plan/Backlog*.

Appendix A: Test Skeletons (illustrative)

Frontend (React Testing Library)

```
import { render, screen, fireEvent } from '@testing-library/react'
import NewJob from '../src/pages/NewJob'

test('TC-002: shows validation error when title empty', async () => {
  render(<NewJob />)
  fireEvent.click(screen.getByRole('button', { name: /post/i }))
  expect(await screen.findByText(/title is required/i)).toBeInTheDocument()
})
```

API (Express + supertest)

```
import request from 'supertest'
import app from '../functions/app'

test('TC-007: blocks job create when kyc not verified', async () => {
  const token = await getAuthToken({ kycVerified: false })
  const res = await request(app)
    .post('/jobs')
    .set('Authorization', `Bearer ${token}`)
    .send({ title: 'Yard help', ... })
  expect(res.status).toBe(403)
})
```

Firestore Rules (rules-unit-testing)

```
it('TC-012: rejects write without kycVerified', async () => {
  const db = authedDB({ uid: 'u1', kycVerified: false })
  const ref = db.collection('jobs').doc('j1')
  await assertFails(ref.set({ title: 'T', ... }))
})
```