

DIGT2107: Practice of Software Development
Project Iteration 3: Testing and Initial Development
OpenBid

Team 1: Tyler Mani Yanness Alaister

Due: November 2, 2025

Course: DIGHT2107 – 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 \leftrightarrow 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 IDs	Case	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, TC-008a-f, TC-009a-b, TC-010a-b, TC-011a-b, TC-012, TC-018a-f		Covered
FR-005	System shall support user authentication with Duo 2FA for sensitive actions.	TC-008		Partially Covered

Req ID	Requirement Description	Test IDs	Case	Coverage
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 shall allow users to manage their profile including avatar upload and personal information.	TC-019a–d		Covered

Non-Functional Requirements (NFR)

Req ID	Requirement Description	Test IDs	Case	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.			Planned
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 (mocked Google Maps).
3. Upload a photo (mock 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 calculates distance between 2 coordinates.
Requirement	FR-002
Preconditions	Valid coordinates are input.
Steps	<ol style="list-style-type: none">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	<ol style="list-style-type: none">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	<ol style="list-style-type: none">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 are tested for validity before the function call and the logic outside the function affirms that apiKey and ref are valid.
Steps	<ol style="list-style-type: none">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 attached 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: Duo 2FA Challenge on Sensitive Action

Requirement: FR-005

Preconditions: User logged in without recent 2FA; action = add payout method.

Expected Result: Duo prompt required; action proceeds only on success.

Status: Pending **Priority:** Medium

TC-008a

Test Case ID	TC-008a
Title	KYC Verification - Create Stripe Verification Session
Requirement	FR-004
Preconditions	User is authenticated; KYC status = pending ; real KYC mode enabled.
Steps	<ol style="list-style-type: none">1. Send POST request to <code>/api/kyc/verification</code> with valid auth JWT token.2. System calls Stripe Identity API to create verification session.3. System stores session ID in user record.4. System returns Stripe verification URL and session ID.
Expected Result	Returns Stripe verification URL (<code>https://verify.stripe.com/*</code>) and session ID (<code>vs_*</code>)
Actual Result	Passed - Returns mocked Stripe URL and session ID.
Status	Passed
Priority	High

TC-008b

Test Case ID	TC-008b
Title	KYC Verification - Unauthorized Access
Requirement	FR-004, NFR-002
Preconditions	User not authenticated or invalid token.
Steps	<ol style="list-style-type: none">1. Send POST request to <code>/api/kyc/verification</code> without auth token or with invalid token.
Expected Result	Returns 401 Unauthorized with error message.
Actual Result	Passed - Returns 401 with <code>{error: "unauthorized"}</code> .
Status	Passed
Priority	High

TC-009a

Test Case ID	TC-009a
Title	KYC Status Check
Requirement	FR-004
Preconditions	User is authenticated.
Steps	<ol style="list-style-type: none">1. Send GET request to <code>/api/kyc/status</code> with valid auth JWT token.2. System retrieves user KYC status from database.3. If verification session exists and status is pending, checks Stripe for updated status.4. System returns KYC status (pending, verified, or failed).
Expected Result	Returns KYC status (pending , verified , or failed).
Actual Result	Passed - Returns valid status from database.
Status	Passed
Priority	High

TC-009b

Test Case ID	TC-009b
Title	KYC Status - Unauthorized Access
Requirement	FR-004, NFR-002
Preconditions	User not authenticated.
Steps	1. Send GET request to <code>/api/kyc/status</code> without auth token.
Expected Result	Returns 401 Unauthorized.
Actual Result	Passed - Returns 401 with error message.
Status	Passed
Priority	High

TC-010

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-011

Title: Submit Review on Completion

Requirement: FR-007

Expected Result: Review saved and visible on profile; duplicate review blocked.

Status: Planned **Priority:** Medium

TC-012

Title: Performance P95 - Map Query

Requirement: NFR-001

Expected Result: P95 end-to-end from filter change to results render \leq 800 ms on stub dataset.

Status: Planned **Priority:** Low

TC-013

Title: Ruleset Audit - No Write Without Claims

Requirement: NFR-002

Expected Result: Firestore rules reject writes missing `kycVerified == true`.

Status: Pending **Priority:** High

TC-014

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-015a

Test Case ID	TC-015a
Title	Profile Page - Display KYC Pending Status
Requirement	FR-004
Preconditions	User authenticated, KYC status = pending .
Steps	<ol style="list-style-type: none">1. Navigate to /profile.2. Profile component renders and displays account status section.3. System retrieves user KYC status from session.
Expected Result	KYC status displays "Pending" tag with appropriate styling.
Actual Result	Passed - Profile page correctly displays pending status.
Status	Passed
Priority	High

TC-015b

Test Case ID	TC-015b
Title	Profile Page - Display KYC Verified Status
Requirement	FR-004
Preconditions	User authenticated, KYC status = verified .
Steps	<ol style="list-style-type: none">1. Navigate to /profile.2. Profile component renders with verified KYC status.
Expected Result	KYC status displays "Verified" tag; no action buttons shown.
Actual Result	Passed - Profile page displays verified status correctly.
Status	Passed
Priority	High

TC-015c

Test Case ID	TC-015c
Title	Profile Page - Complete KYC Button Displayed
Requirement	FR-004
Preconditions	User authenticated, KYC status = pending .
Steps	<ol style="list-style-type: none">1. Navigate to /profile.2. System checks KYC status is not verified.3. Profile component renders action buttons.
Expected Result	"Complete KYC" and "Refresh Status" buttons are visible.
Actual Result	Passed - Action buttons displayed for pending status.
Status	Passed
Priority	High

TC-015d

Test Case ID	TC-015d
Title	Profile Page - Initiate KYC Verification
Requirement	FR-004
Preconditions	User authenticated, KYC pending, production mode.
Steps	<ol style="list-style-type: none">1. User clicks "Complete KYC" button.2. System calls <code>/api/kyc/verification</code> endpoint.3. System receives Stripe verification URL.4. System opens URL in new browser tab.
Expected Result	Stripe Identity verification opens in new tab; notification displayed.
Actual Result	Passed - Verification URL opened correctly.
Status	Passed
Priority	High

TC-015e

Test Case ID	TC-015e
Title	Profile Page - Refresh KYC Status
Requirement	FR-004
Preconditions	User authenticated, KYC pending.
Steps	<ol style="list-style-type: none">1. User clicks "Refresh Status" button.2. System calls <code>/api/kyc/status</code> endpoint.3. System receives updated status.4. System updates UI and session with new status.
Expected Result	KYC status refreshed; UI updated with current status.
Actual Result	Passed - Status refresh works correctly.
Status	Passed
Priority	High

TC-016a

Test Case ID	TC-016a
Title	Profile Page - Upload Avatar Successfully
Requirement	FR-008 (User Profile Management)
Preconditions	User authenticated.
Steps	<ol style="list-style-type: none">1. User clicks "Change Avatar" button.2. User selects valid image file (PNG, < 5MB).3. System uploads file to storage.4. System updates user profile with avatar URL.
Expected Result	Avatar uploaded; preview displayed; success notification shown.
Actual Result	Passed - Avatar upload successful.
Status	Passed
Priority	Medium

TC-016b

Test Case ID	TC-016b
Title	Profile Page - Reject Large Avatar Files
Requirement	FR-008, NFR-002 (Security)
Preconditions	User authenticated.
Steps	<ol style="list-style-type: none">1. User clicks "Change Avatar" button.2. User selects image file larger than 5MB.3. System validates file size before upload.
Expected Result	Upload rejected; error message "Image must be smaller than 5MB" displayed.
Actual Result	Passed - Large files rejected correctly.
Status	Passed
Priority	Medium

TC-016c

Test Case ID	TC-016c
Title	Profile Page - Handle Avatar Upload Error
Requirement	FR-008
Preconditions	User authenticated; network/server error occurs.
Steps	<ol style="list-style-type: none">1. User selects valid image file.2. System attempts upload to /api/avatar/upload.3. Server returns error response.
Expected Result	User-friendly error message displayed; no profile update.
Actual Result	Passed - Error handled gracefully.
Status	Passed
Priority	Medium

TC-016d

Test Case ID	TC-016d
Title	Profile Page - Display Existing Avatar
Requirement	FR-008
Preconditions	User authenticated; avatar URL exists in profile.
Steps	<ol style="list-style-type: none">1. Navigate to /profile.2. System retrieves user avatar URL from database.3. Profile component renders avatar image.
Expected Result	User's avatar displayed in profile header.
Actual Result	Passed - Existing avatar displayed correctly.
Status	Passed
Priority	Low

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 expect(async () => {
    await ref.set({ title: 'T', ... })
  }).rejects()
})
```

KYC Routes (Express + supertest + Jest)

```
import request from 'supertest'
import express from 'express'

// TC-008a: Create Stripe verification session
test('should return Stripe verification URL', async () => {
  const response = await request(app)
    .post('/api/kyc/verification')
```

```

        .set('Authorization', 'Bearer fake-token')
        .expect(200)

    expect(response.body.url).toContain('https://verify.stripe.com/')
    expect(response.body.sessionId).toContain('vs')
  })

  // TC-009a: Check KYC status
  test('should return KYC status', async () => {
    const response = await request(app)
      .get('/api/kyc/status')
      .set('Authorization', 'Bearer fake-token')
      .expect(200)

    expect(['verified', 'pending', 'failed'])
      .toContain(response.body.status)
  })

  // TC-008b: Unauthorized access
  test('should reject unauthorized user', async () => {
    mockAuth.auth.verify.mockResolvedValue(null)

    const response = await request(app)
      .post('/api/kyc/verification')
      .expect(401)

    expect(response.body.error).toBe('unauthorized')
  })

```