

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

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Iteration Goals and Objectives</b>	<b>2</b>
<b>3</b>	<b>Requirement to Test Case Traceability</b>	<b>2</b>
<b>4</b>	<b>Detailed Test Case Descriptions</b>	<b>3</b>
<b>5</b>	<b>Code Repository and Branching Strategy</b>	<b>8</b>
<b>6</b>	<b>Task Allocation and Timeline</b>	<b>8</b>
<b>7</b>	<b>Prototype Overview</b>	<b>9</b>
<b>8</b>	<b>Submission Guidelines</b>	<b>9</b>

# 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 <b>post jobs</b> with title, description, budget, photos, and location.	TC-001, 002	TC-	Partially Covered
FR-002	System shall allow providers to <b>browse jobs on a map</b> with basic filters (radius, category).	TC-003, 004	TC-	Partially Covered
FR-003	System shall allow providers to <b>place bids</b> on open jobs; posters can view and accept a bid.	TC-005, 006	TC-	Partially Covered
FR-004	System shall enforce <b>KYC for all users</b> before posting or bidding.	TC-007, 008a-f	TC-	Partially Covered
FR-005	System shall support <b>user authentication</b> with <b>Duo 2FA</b> for sensitive actions.	TC-008		Partially Covered
FR-006	System shall provide <b>in-thread messaging</b> per job after acceptance.	TC-009		Planned
FR-007	System shall record <b>ratings and reviews</b> after job completion.	TC-010		Planned

## Non-Functional Requirements (NFR)

Req ID	Requirement Description	Test IDs	Case	Coverage
NFR-001	<b>Performance:</b> Search/browse should return results within 800 ms P95 for a metro with 5k open jobs (stub data).	TC-011		Planned
NFR-002	<b>Security:</b> Only KYC-verified users can hit write endpoints for jobs/bids.			Planned
NFR-003	<b>Usability:</b> 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-003

**Title:** Map Browse - Within Radius

**Requirement:** FR-002

**Preconditions:** Seeded jobs exist within 10 km.

**Steps:**

1. Open /jobs.
2. Set radius filter = 10 km.

**Expected Result:** Pins and list only include jobs within 10 km.

**Status:** Pending    **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

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

### TC-008b

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

### TC-009a

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

### TC-009b

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

### TC-010a

<b>Test Case ID</b>	TC-010a
<b>Title</b>	Mock KYC Verification
<b>Requirement</b>	FR-004
<b>Preconditions</b>	Prototype = TRUE enabled, and user authenticated.
<b>Steps</b>	1. Send POST request to <code>/api/kyc/verification</code> with mock auth header. 2. System returns mock verification URL instead of calling Stripe.
<b>Expected Result</b>	Returns mock KYC URL ( <code>https://mock-kyc.local/session/*</code> ) and mock session ID.
<b>Actual Result</b>	Passed - Returns mock URL without Stripe API call.
<b>Status</b>	Passed
<b>Priority</b>	Medium

### TC-010b

<b>Test Case ID</b>	TC-010b
<b>Title</b>	Mock KYC Verification - Unauthorized
<b>Requirement</b>	FR-004, NFR-002
<b>Preconditions</b>	Prototype = TRUE enabled, no authentication.
<b>Steps</b>	1. Send POST request to <code>/api/kyc/verification</code> without mock auth header (x-mock-uid).
<b>Expected Result</b>	Returns 401 Unauthorized.
<b>Actual Result</b>	Passed - Returns 401 error.
<b>Status</b>	Passed
<b>Priority</b>	Medium

### TC-011a

<b>Test Case ID</b>	TC-011a
<b>Title</b>	Mock KYC Status Check
<b>Requirement</b>	FR-004
<b>Preconditions</b>	Prototype = TRUE enabled, and user authenticated with mock auth header.
<b>Steps</b>	1. Send GET request to <code>/api/kyc/status</code> with mock auth header (x-mock-uid). 2. System retrieves KYC status from mock database without calling Stripe.
<b>Expected Result</b>	Returns KYC status ( <b>verified</b> or <b>pending</b> ) from mock database.
<b>Actual Result</b>	Passed - Returns status without external API calls.
<b>Status</b>	Passed
<b>Priority</b>	Medium

### TC-011b

<b>Test Case ID</b>	TC-011b
<b>Title</b>	Mock KYC Status Check - Unauthorized
<b>Requirement</b>	FR-004, NFR-002
<b>Preconditions</b>	Prototype = TRUE enabled, no authentication.
<b>Steps</b>	1. Send GET request to <code>/api/kyc/status</code> without mock auth header.
<b>Expected Result</b>	Returns 401 Unauthorized.
<b>Actual Result</b>	Passed - Returns 401 error.
<b>Status</b>	Passed
<b>Priority</b>	Medium

### TC-012

<b>Test Case ID</b>	TC-012
<b>Title</b>	Mock KYC Force Pass
<b>Requirement</b>	FR-004 (Prototype Mode)
<b>Preconditions</b>	Prototype = TRUE enabled, and user authenticated.
<b>Steps</b>	1. Send POST request to <code>/api/kyc/force-pass</code> . 2. System immediately marks user as KYC verified without Stripe.
<b>Expected Result</b>	User KYC status updated to <b>verified</b> , returns success.
<b>Actual Result</b>	Passed - KYC status updated in mock database.
<b>Status</b>	Passed
<b>Priority</b>	Low

### TC-013

**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-014

**Title:** Submit Review on Completion

**Requirement:** FR-007

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

**Status:** Planned    **Priority:** Medium

## TC-015

**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-016

**Title:** Ruleset Audit - No Write Without Claims

**Requirement:** NFR-002

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

**Status:** Pending    **Priority:** High

## TC-017

**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

# 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 />)
  // ... test logic ...
})
```

```

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', ... }))
})

```

## 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')
})
```