

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 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 Cov- ered
FR-002	System shall allow providers to <b>browse jobs on a map</b> with basic filters (radius, category).	TC-003a-d, TC-004		Partially Cov- ered
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 Cov- ered
FR-004	System shall enforce <b>KYC for all users</b> before posting or bidding.	TC-007		Partially Cov- ered
FR-005	System shall support <b>user authentication</b> with <b>Duo 2FA</b> for sensitive actions.	TC-008		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
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 IDs	Case	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 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.	TC-007, TC-012		Partially Covered
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 (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

<b>Test Case ID</b>	TC-003a
<b>Title</b>	The haversineFormulakm function accurately calculates distance between 2 coordinates.
<b>Requirement</b>	FR-002
<b>Preconditions</b>	Valid coordinates are input.
<b>Steps</b>	<ol style="list-style-type: none"><li>1. Coordinates are input into the function.</li><li>2. Function calculates the distance between the coordinates.</li><li>3. Function returns the distance between the coordinates.</li></ol>
<b>Expected Result</b>	Returns distance in km of the 2 coordinates, returns infinity when there is invalid input.
<b>Actual Result</b>	Passed - Returns distance accurately within a specific tolerance. Coordinates tested are up to 250km apart.
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-003b

<b>Test Case ID</b>	TC-003b
<b>Title</b>	The isValidCoords formula accurately determines whether the given inputs are valid latitude and longitude values.
<b>Requirement</b>	FR-002
<b>Preconditions</b>	None.
<b>Steps</b>	<ol style="list-style-type: none"><li>Coordinates are input into the function.</li><li>Function checks input for validity then checks whether the input values are valid coordinates.</li><li>Function returns whether the coordinates are valid.</li></ol>
<b>Expected Result</b>	Returns true or false depending on whether the input is valid coordinates.
<b>Actual Result</b>	Passed - accurately surmises whether the input coordinates are valid.
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-003c

<b>Test Case ID</b>	TC-003c
<b>Title</b>	The degToRad function accurately converts a number in degrees to radians.
<b>Requirement</b>	FR-002
<b>Preconditions</b>	Input should be a number.
<b>Steps</b>	<ol style="list-style-type: none"><li>a number in degrees is input into the function.</li><li>Function checks input for validity then returns the input value in radians.</li></ol>
<b>Expected Result</b>	Returns the input number converted into radians if the input is valid, returns infinity otherwise.
<b>Actual Result</b>	Passed - accurately converts a valid input to radians and returns infinity otherwise.
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-003d

<b>Test Case ID</b>	TC-003c
<b>Title</b>	The createMap function loads a google map. (uses api mocking)
<b>Requirement</b>	FR-002
<b>Preconditions</b>	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.
<b>Steps</b>	<ol style="list-style-type: none"><li>1. The google maps api loader is initialized.</li><li>2. The markers array input is filtered for valid markers or is set to an empty array if the markers input is bad.</li><li>3. The google maps api loader is used to load the api.</li><li>4. A new google map is created and valid markers are attached to the map.</li></ol>
<b>Expected Result</b>	Creates a map with valid markers attached to it.
<b>Actual Result</b>	Passed - creates a map with valid markers attached to it, invalid markers are discarded and a bad markers input is gracefully handled.
<b>Status</b>	Passed
<b>Priority</b>	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 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

<b>Test Case ID</b>	TC-014a
<b>Title</b>	Firebase Signup - Provision Account and Send Verification Email
<b>Requirement</b>	FR-008
<b>Preconditions</b>	Email is brand new inside the Firebase project; Identity Toolkit is online.
<b>Steps</b>	<ol style="list-style-type: none"><li>1. Send POST /api/auth/signup with first/last name, valid email, password, confirmPassword.</li><li>2. Let Firebase create the account and queue the verification email.</li><li>3. Read the JSON response from our API.</li></ol>
<b>Expected Result</b>	HTTP 201 with sanitized user data, email + KYC both pending, and Firebase verification email requested.
<b>Actual Result</b>	Passed – Confirmed in <code>server/src/routes/_tests__/auth.integration.real.routes.test.js</code>
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-014b

<b>Test Case ID</b>	TC-014b
<b>Title</b>	Firebase Signup - Reject Invalid Credentials
<b>Requirement</b>	FR-008
<b>Preconditions</b>	Same Firebase project as TC-014a; email still unused.
<b>Steps</b>	<ol style="list-style-type: none"><li>1. POST /api/auth/signup with a broken email or a password shorter than eight characters.</li><li>2. Watch the server reject the payload before any Firebase call.</li></ol>
<b>Expected Result</b>	HTTP 400 with a clear error; Firebase signup helper never runs.
<b>Actual Result</b>	Passed – Shown in the same Jest suite for real adapters.
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-014c

<b>Test Case ID</b>	TC-014c
<b>Title</b>	Firebase Login - Verified User Receives Session Tokens
<b>Requirement</b>	FR-008
<b>Preconditions</b>	User already exists in Firebase with both email and KYC marked <b>verified</b> .
<b>Steps</b>	<ol style="list-style-type: none"><li>POST /api/auth/login with the correct email and password.</li><li>Inspect the response for session + requirement flags.</li></ol>
<b>Expected Result</b>	HTTP 200 with sanitized user, Firebase ID + refresh tokens, and <b>requirements = {emailVerified: true, kycVerified: true}</b> .
<b>Actual Result</b>	Passed – Assertions live in auth.integration.real.routes.test.js.
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-014d

<b>Test Case ID</b>	TC-014d
<b>Title</b>	Firebase Login - Reject Wrong Passwords
<b>Requirement</b>	FR-008
<b>Preconditions</b>	User exists in Firebase; test double forces Identity Toolkit to return INVALID_PASSWORD.
<b>Steps</b>	<ol style="list-style-type: none"><li>POST /api/auth/login with the wrong password.</li><li>Capture HTTP status + body.</li></ol>
<b>Expected Result</b>	HTTP 401 with <b>{error: "invalid credentials"}</b> ; no session issued.
<b>Actual Result</b>	Passed – Covered in auth.integration.real.routes.test.js.
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-015a

<b>Test Case ID</b>	TC-015a
<b>Title</b>	Role Switch - Authorization Required
<b>Requirement</b>	FR-009
<b>Preconditions</b>	User is signed in and has a valid Firebase ID token.
<b>Steps</b>	<ol style="list-style-type: none"><li>PATCH /api/auth/role without the Authorization header.</li><li>Repeat with Authorization: Bearer &lt;idToken&gt; and payload <b>{role: "contractor"}</b>.</li></ol>
<b>Expected Result</b>	Unauthorized request returns 401; authorized request returns 200 with updated <b>userType</b> and refreshed requirement flags.
<b>Actual Result</b>	Passed – Exercised via Jest role-switch integration test.
<b>Status</b>	Passed
<b>Priority</b>	Medium

## TC-015b

<b>Test Case ID</b>	TC-015b
<b>Title</b>	Auth Me - Return Sanitized User Context
<b>Requirement</b>	FR-009
<b>Preconditions</b>	Contractor user plus valid Firebase ID token.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. GET <code>/api/auth/me</code> without Authorization header and confirm 401.</li> <li>2. Repeat with Authorization header; inspect payload for sanitized user fields.</li> </ol>
<b>Expected Result</b>	Unauthorized call blocked; authorized call returns sanitized <code>user</code> payload (no <code>passwordHash</code> ) plus metadata.
<b>Actual Result</b>	Passed – Documented in <code>auth.integration.real.routes.test.js</code> .
<b>Status</b>	Passed
<b>Priority</b>	Medium

## TC-016a

<b>Test Case ID</b>	TC-016a
<b>Title</b>	Session Service - Persist and Notify Subscribers
<b>Requirement</b>	FR-010
<b>Preconditions</b>	Browser environment available (Vitest JSDOM); subscribers registered.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Call <code>setSession</code> with a real Firebase-authenticated user payload.</li> <li>2. Verify <code>subscribeSession</code> listener triggered.</li> <li>3. Inspect <code>localStorage</code> for serialized session data.</li> </ol>
<b>Expected Result</b>	Session stored under <code>openbid_session</code> ; only real-user metadata persisted; listeners invoked once.
<b>Actual Result</b>	Passed – Covered by <code>client/src/__tests__/session.test.js</code> .
<b>Status</b>	Passed
<b>Priority</b>	Medium

## TC-016b

<b>Test Case ID</b>	TC-016b
<b>Title</b>	Session Service - setUser Preserves Requirements
<b>Requirement</b>	FR-010
<b>Preconditions</b>	Existing session with requirement flags stored.
<b>Steps</b>	<ol style="list-style-type: none"> <li>1. Initialize session via <code>setSession</code>.</li> <li>2. Invoke <code>setUser</code> with new user payload and explicit requirements.</li> <li>3. Query <code>getRequirements</code> and <code>getSession</code>.</li> </ol>
<b>Expected Result</b>	Updated user persisted; requirements reflect provided override (email + KYC flags).
<b>Actual Result</b>	Passed – Validated in <code>session.test.js</code> .
<b>Status</b>	Passed
<b>Priority</b>	Medium

## TC-016c

<b>Test Case ID</b>	TC-016c
<b>Title</b>	Session Service - Inactivity Monitor Auto Logout
<b>Requirement</b>	FR-010
<b>Preconditions</b>	Session established; Vitest fake timers.
<b>Steps</b>	<ol style="list-style-type: none"><li>1. Call <code>startInactivityMonitor</code> with spy callback.</li><li>2. Advance timers to just before the 2-minute limit; ensure no callback.</li><li>3. Advance timers past the limit; ensure callback fires and cleanup works.</li></ol>
<b>Expected Result</b>	Timeout callback executes exactly once after 2 minutes of inactivity, indicating logout.
<b>Actual Result</b>	Passed – Covered by Vitest case in <code>session.test.js</code> .
<b>Status</b>	Passed
<b>Priority</b>	Medium

## TC-017a

<b>Test Case ID</b>	TC-017a
<b>Title</b>	Job Create - Verified Contractor Path
<b>Requirement</b>	FR-011
<b>Preconditions</b>	Contractor account with <code>kycStatus = verified</code> and a valid Firebase ID token.
<b>Steps</b>	<ol style="list-style-type: none"><li>1. POST <code>/api/jobs</code> with the Authorization header plus title, description, budget, and location.</li><li>2. Read the JSON response.</li></ol>
<b>Expected Result</b>	HTTP 200 with a job whose <code>posterId</code> matches the contractor and status <code>open</code> .
<b>Actual Result</b>	Passed – Verified in <code>jobs.integration.real.routes.test.js</code> .
<b>Status</b>	Passed
<b>Priority</b>	High

## TC-017b

<b>Test Case ID</b>	TC-017b
<b>Title</b>	Job Create - Bidder Rejected
<b>Requirement</b>	FR-011
<b>Preconditions</b>	Bidder account logged in with Firebase token.
<b>Steps</b>	<ol style="list-style-type: none"><li>1. POST <code>/api/jobs</code> using the bidder's token and minimal payload.</li></ol>
<b>Expected Result</b>	HTTP 403 with <code>{error: "contractor_only"}</code> ; no job written.
<b>Actual Result</b>	Passed – Documented in <code>jobs.integration.real.routes.test.js</code> .
<b>Status</b>	Passed
<b>Priority</b>	High

### TC-017c

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

### TC-017d

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

### TC-017e

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

### TC-017f

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

### TC-017g

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

### TC-017h

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

### TC-017i

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

### TC-018a

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

### TC-018b

<b>Test Case ID</b>	TC-018b
<b>Title</b>	Bid Accept - Lock Job/Bid Lifecycle
<b>Requirement</b>	FR-012
<b>Preconditions</b>	Contractor owns a job; bidder submits a valid bid; Firebase tokens exist for contractor, bidder, and observer.
<b>Steps</b>	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.
<b>Expected Result</b>	Job flips to awarded with awardedBidId; further edits return job_locked / bidding_closed; observers no longer see the job.
<b>Actual Result</b>	Passed – “Accepting a bid hides the job from outsiders and locks further edits” test.
<b>Status</b>	Passed
<b>Priority</b>	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', ... }))
})
```