CSC1097

HireTrack System Test

Siri Nandipaty - 21449384

Anushree Umak - 21343003

Graham Healy

2nd May

0. Table of Contents

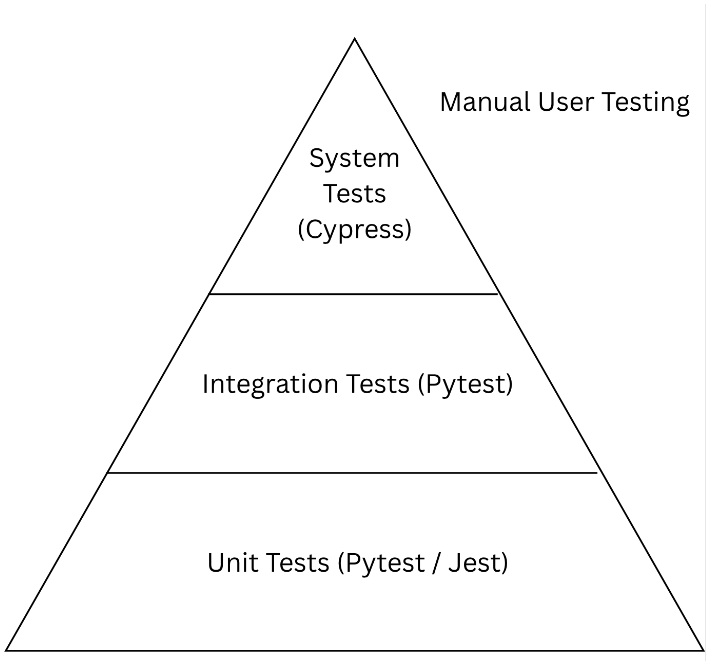
1. Introduction

This document outlines the testing approach and implementation details for the platform we worked on, called HireTrack.Given the system's reliance on dynamic user interfaces and RESTful backend APIs, the testing strategy emphasised end-to-end (system) testing using Cypress, unit testing on both frontend and backend components, and integration testing for cross-component flows.

This testing framework was developed to ensure each part of the system works as intended individually, and that recruiter-facing features operate correctly when all components are integrated.

2. Testing Strategy

The testing strategy for the Hiretrack platform was divided into three primary categories: system testing, unit testing, and integration testing. System testing was conducted using Cypress to validate complete recruiter workflows through the user interface and backend interactions. These tests ensured that key features such as job posting, applicant viewing, and dashboard navigation functioned as expected in real-world scenarios. Unit testing was implemented for both frontend and backend components. On the frontend, Jest was used to verify the behavior of individual React components, including input handling, button state changes, and interface responses. On the backend, Pytest was used to test specific route logic and utility functions in isolation, ensuring that each individual operation behaved correctly under different conditions. Integration testing focused on validating multi-step backend flows where multiple components needed to work together. These included user authentication, job lifecycle management, resume parsing, calendar integration, and messaging features. By applying this comprehensive testing approach, the system was evaluated at multiple levels to ensure reliability, accuracy, and performance across all recruiter-facing features.



2.1 Testing Scope

System Tests (Cypress):

Recruiter workflows including job posting, job search, job tracking, and applicant viewing. Jobseeker workflows including job application, inbox messaging, tracking, and job search.

Integration Tests (Pytest):

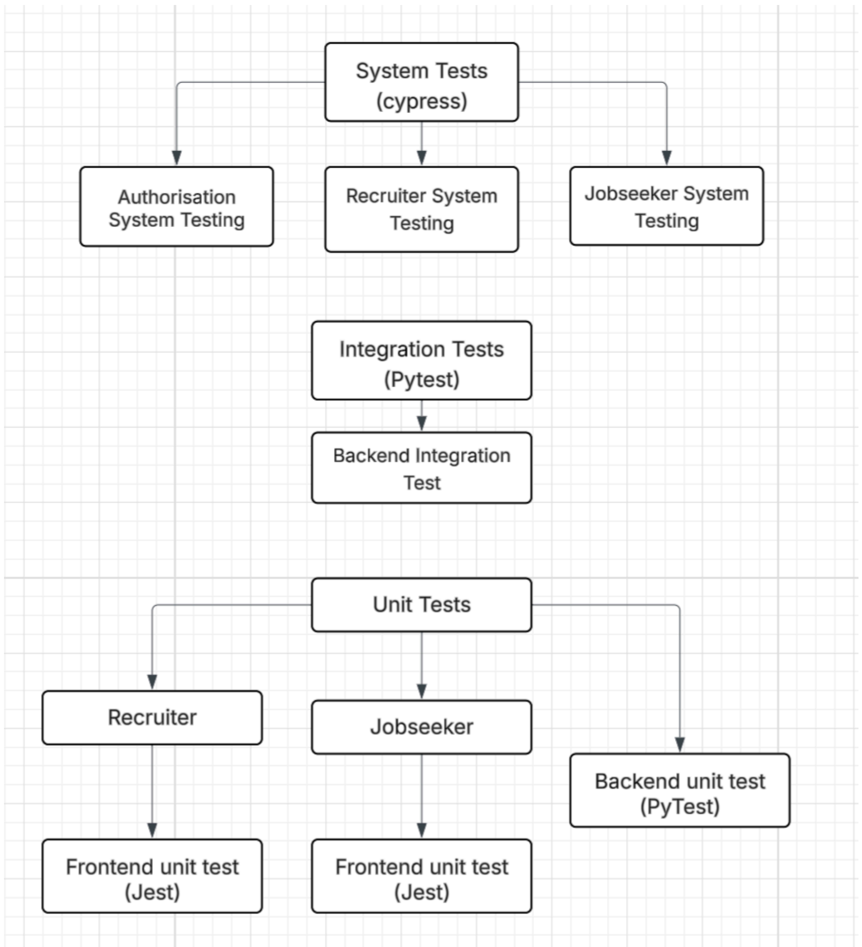
Backend multi-step workflows such as user authentication, job lifecycle, resume parsing, calendar scheduling, and chat messaging.

Unit Tests (Pytest / Jest):

Backend route logic, input validation, and response formatting; frontend component rendering, state updates, and user interaction handlers.

2.2 Manual User Testing

In addition to formal testing, manual user testing was conducted to validate usability and overall user experience. Testers manually explored key recruiter workflows, including login, job posting, applicant viewing, and calendar scheduling. Feedback indicated that the user interface was intuitive and that the resume suggestion feature added meaningful value. This informal testing supported iterative development and was especially useful during UI refinement stages.


System Tests (cypress)
Authorisation System Testing
Recruiter System Testing
Jobseeker System Testing
Integration Tests (Pytest)
Backend Integration Test
Unit Tests
Recruiter
Jobseeker
Backend unit test (PyTest)
Frontend unit test (Jest)
Frontend unit test (Jest)

3. System Testing (Cypress)

3.1 Setup for End-to-End testing

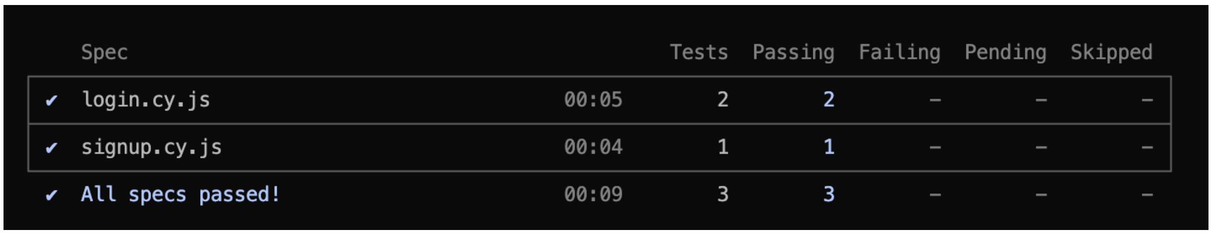
- Tests Located in: cypress/e2e/
- Location: 2025-csc1097-Hiretrack
- Install requirements: pip install -r requirements.txt
- Installation Command: npm install cypress --save-dev
- Run Command: npx cypress run

3.2 Authorisation System Testing

Tests Located in: cypress/e2e/auth/

File	Description
login.cy.js	Verifies the login flow including form validation, error handling, and successful login
signup.cy.js	Test the user registration process with input validation, API interaction, and success/error scenarios

All 3 system tests passed successfully as of April 17.

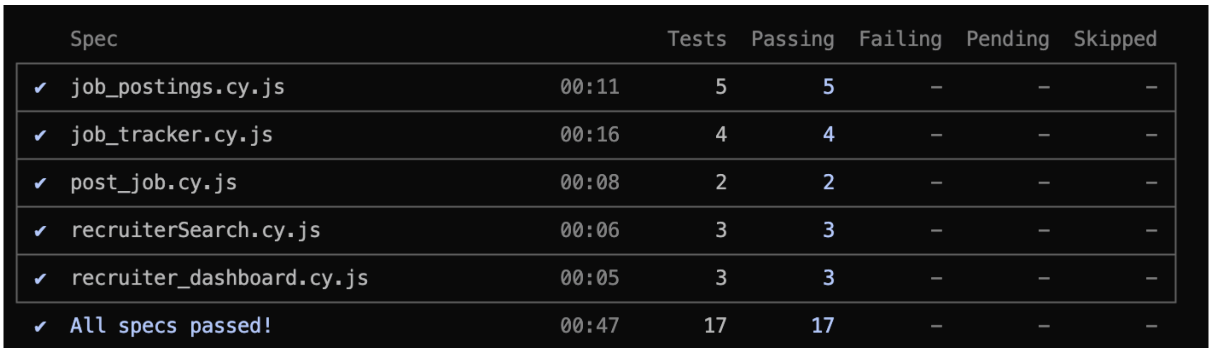


```
     Spec                                        Tests  Passing  Failing  Pending  Skipped
  ┌─────────────────────────────────────────────────────────────────────────────────────┐
  │ ✔  login.cy.js                      00:05        2        2        -        -        - │
  ├─────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  signup.cy.js                     00:04        1        1        -        -        - │
  └─────────────────────────────────────────────────────────────────────────────────────┘
    ✔  All specs passed!                00:09        3        3        -        -        -
```

3.3 Recruiter System Testing

Tests Located in: cypress/e2e/recruiter/

File	Description
post_job.cy.js	Posts a job, verifies redirect and success confirmation
job_postings.cy.js	View, edit, and delete job postings
job_tracker.cy.js	Chart input updates and job trend logic
recruiter_dashboard.cy.js	Dashboard navigation and component render checks
recruiterSearch.cy.js	View job seekers, all applications, and individual applicants



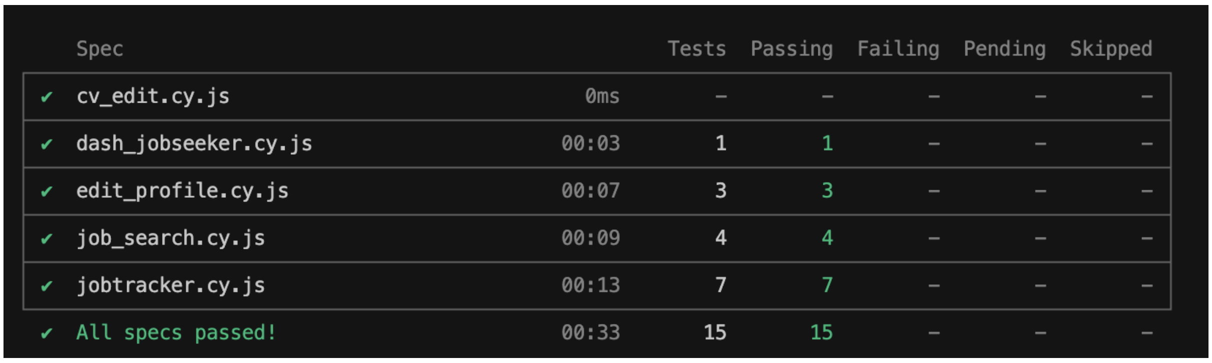
```
     Spec                                          Tests  Passing  Failing  Pending  Skipped
  ✔  job_postings.cy.js                  00:11         5        5        -        -        -
  ✔  job_tracker.cy.js                   00:16         4        4        -        -        -
  ✔  post_job.cy.js                      00:08         2        2        -        -        -
  ✔  recruiterSearch.cy.js               00:06         3        3        -        -        -
  ✔  recruiter_dashboard.cy.js           00:05         3        3        -        -        -
  ✔  All specs passed!                   00:47        17       17        -        -        -
```

All 17 system tests passed successfully as of April 17.

3.4 Jobseeker System Testing

Tests Located in: cypress/e2e/jobseeker/

File	Description
dash_jobseeker.cy.js	Verifies the Jobseeker Dashboard loads correctly and that all sidebar buttons and job columns are visible and functional.
edit_profile.cy.js	Tests the Edit Profile page's ability to display, update, and cancel user profile information.
job_search.cy.js	Checks that job listings load with a loading indicator, can be filtered by search, and that the pagination and More Info buttons work.
jobtracker.cy.js	Ensures the Job Tracker page renders the dashboard UI, allows input interactions, and displays job charts and calendar integration options.


```
       Spec                                     Tests  Passing  Failing  Pending  Skipped
  ┌──────────────────────────────────────────────────────────────────────────────────────┐
  │ ✔  cv_edit.cy.js                       0ms      -        -        -        -        - │
  ├──────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  dash_jobseeker.cy.js              00:03      1        1        -        -        - │
  ├──────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  edit_profile.cy.js                00:07      3        3        -        -        - │
  ├──────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  job_search.cy.js                  00:09      4        4        -        -        - │
  ├──────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  jobtracker.cy.js                  00:13      7        7        -        -        - │
  └──────────────────────────────────────────────────────────────────────────────────────┘
    ✔  All specs passed!                 00:33     15       15        -        -        -
```
All 15 system tests passed successfully as of April 17.

4. Backend Integration Testing (Pytest)

Tests Located in: backend/tests/integration/

These tests validate multi-step backend flows involving route ↔ database ↔ external service interactions.

File	Flow Covered
test_auth_flow.py	User registration, login, and token verification
test_chat_flow.py	Recruiter ↔ Jobseeker messaging, chat creation, chat history
test_cv_flow.py	Resume upload, parsing, and structured saving
test_google_calendar_flow.py	Interview scheduling, calendar integration, video link creation
test_job_lifecycle_flow.py	Job creation, fetch, and update lifecycle
test_jobseeker_dashboard_flow.py	Retrieval of jobseeker dashboard jobs and job data
test_recruiter_dashboard_flow.py	Recruiter-side analytics, job tracking, and dashboard metrics

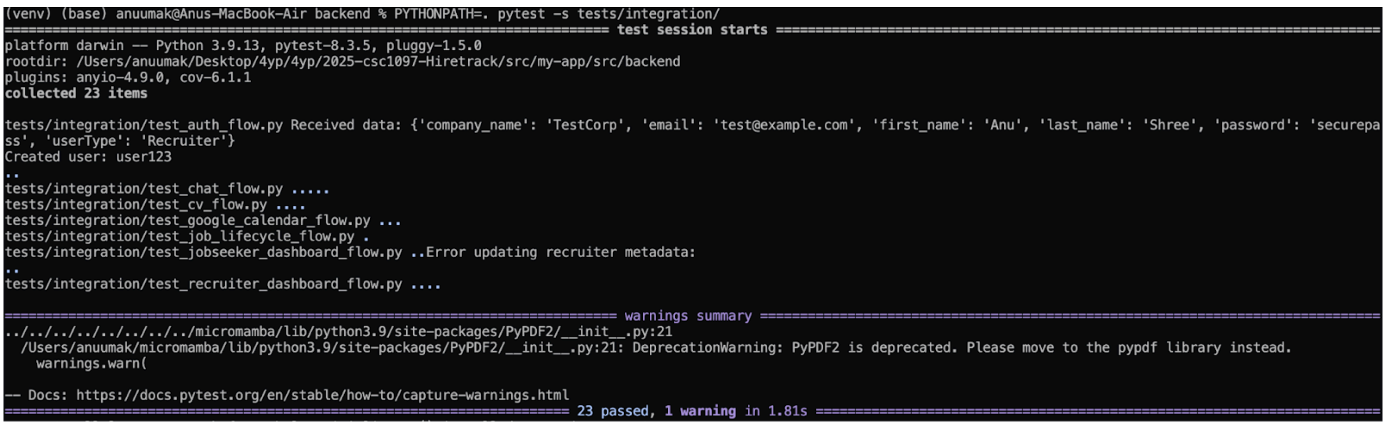
To run these tests

Location: 2025-csc1097-Hiretrack/src/my-app/src/backend

Installation Command: pip install pytest pytest-cov

Run Command: PYTHONPATH=. pytest -s tests/integration/

All 23 integration tests passed successfully as of April 17.



```
(venv) (base) anuumak@Anus-MacBook-Air backend % PYTHONPATH=. pytest -s tests/integration/
============================================================================ test session starts ============================================================================
platform darwin -- Python 3.9.13, pytest-8.3.5, pluggy-1.5.0
rootdir: /Users/anuumak/Desktop/4yp/4yp/2025-csc1097-Hiretrack/src/my-app/src/backend
plugins: anyio-4.9.0, cov-6.1.1
collected 23 items

tests/integration/test_auth_flow.py Received data: {'company_name': 'TestCorp', 'email': 'test@example.com', 'first_name': 'Anu', 'last_name': 'Shree', 'password': 'securepa
ss', 'userType': 'Recruiter'}
Created user: user123
..
tests/integration/test_chat_flow.py .....
tests/integration/test_cv_flow.py ....
tests/integration/test_google_calendar_flow.py ...
tests/integration/test_job_lifecycle_flow.py .
tests/integration/test_jobseeker_dashboard_flow.py ..Error updating recruiter metadata:
..
tests/integration/test_recruiter_dashboard_flow.py ....

============================================================================= warnings summary ==============================================================================
../../../../../../../../micromamba/lib/python3.9/site-packages/PyPDF2/__init__.py:21
  /Users/anuumak/micromamba/lib/python3.9/site-packages/PyPDF2/__init__.py:21: DeprecationWarning: PyPDF2 is deprecated. Please move to the pypdf library instead.
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================================================================= 23 passed, 1 warning in 1.81s ========================================================================
```

5. Unit Tests

5.1 Backend Testing

Tests Located in**:** backend/tests/unit/

The following backend unit tests were successfully implemented and passed:

File	Description
test_auth.py	Token creation and decoding
test_chat.py	Messaging endpoints and response handling
test_create_job.py	Validation and creation of job entries
test_cv_generate.py	CV file generation using extracted fields
test_cv.py	CV parsing utility logic
test_cv_suggestions.py	GPT-powered suggestion generation
test_edit_profile.py	Profile update endpoint and validation
test_google_cal.py	Google Calendar route functionality
test_notifications.py	Notification logic (email/message triggers)
test_recruiterdash.py	Dashboard data aggregation
test_recruitersearch.py	Seeker search route logic
test_seeker_dashboard.py	Jobseeker dashboard endpoint
test_seekeractions.py	Application and interaction logic
test_seekersearch.py	Candidate filtering, tag search
test_view_jobpostings.py	Fetching recruiter job list
test_viewapplicants.py	Retrieving list of applicants per job

Unexecuted Tests:

Due to environment mismatches or legacy code restructuring, the following unit test files were not executed or removed:

- test_cors.py: CORS functionality now handled via Flask extension, test deemed redundant.
- test_edit_job.py: Logic merged into test_create_job.py and test_view_jobpostings.py.
- test_train_resume.py: Refactored CV parsing pipeline replaced this legacy script.

Each test was initially designed and scaffolded, but as the backend evolved, some were deprecated for clarity and consolidated into more robust test files. Time was spent troubleshooting test failures and dependency conflicts during local execution.

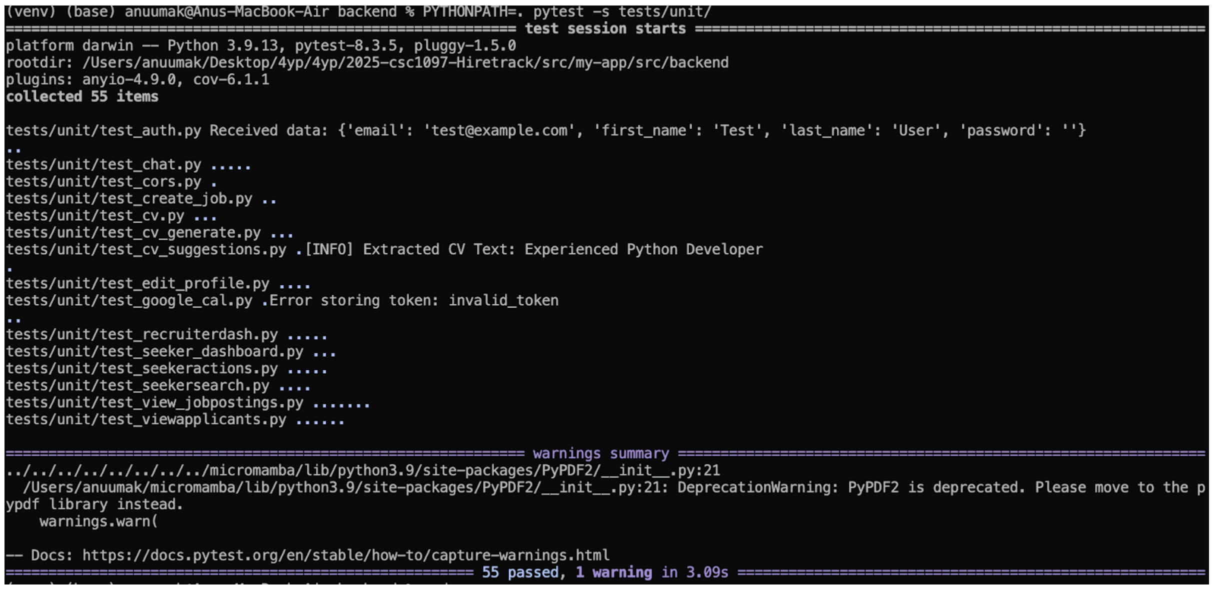
To run these tests

Location: 2025-csc1097-Hiretrack/src/my-app/src/backend

Installation Command: pip install pytest pytest-cov

Run Command: PYTHONPATH=. pytest -s tests/unit/

All 55 unit tests passed successfully as of April 17.



```
(venv) (base) anuumak@Anus-MacBook-Air backend % PYTHONPATH=. pytest -s tests/unit/
============================================================ test session starts ============================================================
platform darwin -- Python 3.9.13, pytest-8.3.5, pluggy-1.5.0
rootdir: /Users/anuumak/Desktop/4yp/4yp/2025-csc1097-Hiretrack/src/my-app/src/backend
plugins: anyio-4.9.0, cov-6.1.1
collected 55 items

tests/unit/test_auth.py Received data: {'email': 'test@example.com', 'first_name': 'Test', 'last_name': 'User', 'password': ''}
..
tests/unit/test_chat.py .....
tests/unit/test_cors.py .
tests/unit/test_create_job.py ..
tests/unit/test_cv.py ...
tests/unit/test_cv_generate.py ...
tests/unit/test_cv_suggestions.py .[INFO] Extracted CV Text: Experienced Python Developer
.
tests/unit/test_edit_profile.py ....
tests/unit/test_google_cal.py .Error storing token: invalid_token
..
tests/unit/test_recruiterdash.py .....
tests/unit/test_seeker_dashboard.py ...
tests/unit/test_seekeractions.py .....
tests/unit/test_seekersearch.py ....
tests/unit/test_view_jobpostings.py .......
tests/unit/test_viewapplicants.py ......

============================================================= warnings summary ==============================================================
../../../../../../../../micromamba/lib/python3.9/site-packages/PyPDF2/__init__.py:21
  /Users/anuumak/micromamba/lib/python3.9/site-packages/PyPDF2/__init__.py:21: DeprecationWarning: PyPDF2 is deprecated. Please move to the p
ypdf library instead.
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
====================================================== 55 passed, 1 warning in 3.09s ======================================================
```

5.2 Frontend Unit Testing

5.2.1 Recruiter Unit Testing (Jest)

Tests Located in**:** my-app/__tests__/

File	Description
post-job.test.js	Input field validation and submit button logic
view-job-postings.test.js	Table rendering, edit button logic
recruiter-search.test.js	View of all job seeker profiles
dashboard-recruiter.test.js	Dashboard components rendered correctly
jobtracker-recruiter.test.js	Trend input updates and chart state

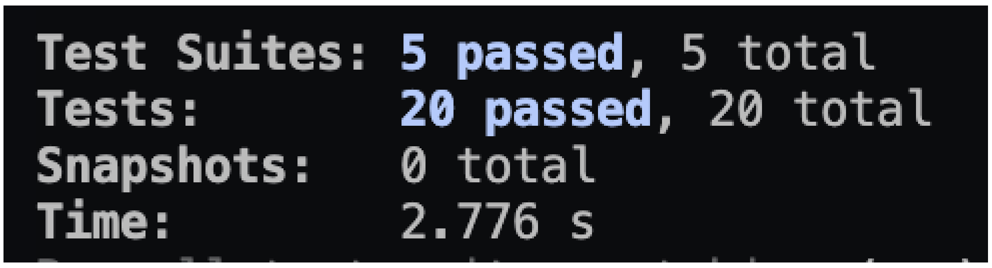
To run these tests

Location: 2025-csc1097-Hiretrack/

Installation Command: npm install --save-dev jest @testing-library/react

Run Command: npx jest src/my-app/__tests__/

All 20 unit tests passed successfully as of April 17.



```
Test Suites: 5 passed, 5 total
Tests:       20 passed, 20 total
Snapshots:   0 total
Time:        2.776 s
```

6. CI/CD Pipeline

Although the project includes thorough backend unit and integration tests using Pytest, these tests could not be executed within the GitLab CI/CD pipeline due to technical constraints. The amount of dependencies required , including nltk, spaCy, and downloading the language model en_core_web_sm which made the setup process too large and resource-heavy. As a result, the pipeline exceeded GitLab's memory, time, or resource limits during installation, preventing successful execution. To ensure quality, all backend unit and integration tests were run locally, and their results were manually verified outside of the CI/CD environment.

The Cypress end-to-end tests were another area that presented challenges. While they worked reliably in local development, running them inside GitLab's CI pipeline proved unstable and inconsistent. Managing the simultaneous startup of the React frontend and Flask backend using background execution (&) and sleep delays was unreliable. Additionally, using Cypress's official Docker image (cypress/included:14.1.0) introduced conflicts related to notification settings: with or without the --no-notify flag, the test runner would either fail to launch properly or hang indefinitely. Because of these persistent issues, Cypress testing was disabled in the pipeline, though all end-to-end tests continue to function correctly when run locally.

On the other hand, the frontend unit tests built with React Testing Library and Jest are fully functional and stable. These tests consistently pass both locally and within GitLab CI, providing reliable coverage for critical frontend components and user interactions. They form the primary automated testing layer of the project. The frontend unit tests were implemented using react-scripts test and have been successfully integrated into the CI/CD pipeline without any issues.

7. User Testing

User testing has been conducted throughout the later stages of the project. This involved giving a small sample of users a version of the application and making them fill in a Google Forms document to document their experience while using the website.

With the user testing we were able to identify some bugs in the system that needed addressing. It was a useful tool to have a better understanding of what decisions a user would make within the app and if certain series of actions lead to any issues. We took the user feedback and made changes accordingly which involved fixing bugs, making UI changes to signpost better to the users on where certain information is located and how to access it.

Some notable changes we made that were flagged due to user testing was the incorporation of custom alerts which appeared when users performed an action, this was used to show the user if their action was successful or not. This makes the engagement through the application less confusing for the user as stated by users who tested after the change was made.

More information on user testing results can be found here.

Question Asked	Average Rating from 6 user testers on a scale of 1-5
How intuitive was the process of logging in	5.00
How intuitive was the process of signing up to the website	4.83
How easy was it to upload and view your own cv on the dashboard	4.67
How easy was it to upload and view your own cv on the dashboard	4.50
How easy was it to search for a job and view its details	4.50
How easy was it to apply for a job	4.67
How easy was it to save a job	4.17
How easy was it to find the match score for a job	4.50
How easy was it to access the inbox and send a message to another user	4.00
How would you rate your experience n the website	4.83

8. Known Issues and Future Improvements

While the testing framework covers a wide range of flows across both the recruiter and jobseeker sides, several areas remain for future improvement:

CI/CD Pipeline Limitations

Although Cypress tests passed reliably in local development, persistent issues were encountered when attempting to run them inside the GitLab CI/CD pipeline. These challenges were primarily related to environment mismatches, instability launching the frontend and backend services within the Docker-based pipeline, and incompatibilities with Cypress's notification settings. As a result, Cypress end-to-end tests are not currently included in CI automation and remain limited to local verification. Additionally, backend unit and integration tests using Pytest could not be run in CI due to excessive dependency setup, which caused the job to exceed GitLab's resource limits. Backend tests were manually verified outside the pipeline instead. Future improvements may involve splitting test jobs, optimising dependency installation, or using a more scalable CI infrastructure.

Frontend Integration Testing

While full-system Cypress tests were designed to validate jobseeker and recruiter flows, their reliance on end-to-end environment stability limited their usefulness in CI/CD. A future improvement would be to incorporate mid-level integration tests using React Testing Library with mocked backend APIs. This would allow key frontend flows (such as job application submission and resume editing) to be validated independently of backend or deployment issues.

Backend Validation Enhancements

Certain backend routes would benefit from stronger form validation and stricter error handling, particularly for edge cases or invalid user input. Enhancing API responses with more informative error messages and additional input checks would improve overall system resilience and user experience.

Collaborative Feature Testing

Due to time constraints, the Collaborative Resume Editing feature; which allows users to edit and optimise their CVs, was not included in automated or manual test cycles. Future work should prioritise thorough testing of this feature to ensure its stability, functionality, and seamless user experience across different scenarios.

9. References

[1] -
https://docs.google.com/spreadsheets/d/1uPsECnLXdx1rVJsCQPndd4PXO07FAJKRiYF2JfF6A6s/edit?usp=sharing