CoderSTAT Testing Documentation  
Version: 1.0  
Last Updated: April 20, 2025

## 1. Testing Strategy Overview

We conducted 5 layers of testing to ensure reliability, security, and scalability:

1. Unit Tests - Core components & utilities
2. Integration Tests - API endpoints & data flows
3. End-to-End (E2E) Tests - User workflows
4. Performance Tests - Load handling
5. User Acceptance Testing (UAT) - Real-world validation

## 2. Backend API Testing

Tools: Jest, Supertest, MSW (Mock Service Worker)

## **Tested Endpoints**

| **Endpoint** | **Test Cases** | **Status** |
| --- | --- | --- |
| /api/profile | Auth validation, CRUD operations | ✅ |
| /api/stats | Data scraping, aggregation | ✅ |
| /api/questions/import | CSV parsing, DB insertion | ✅ |
| /api/leaderboard | Sorting, platform filters | ✅ |

## 

## **Sample Test - Profile API**

javascript

describe('GET /api/profile', () => {

it('rejects unauthenticated requests', async () => {

const res = await request(app).get('/api/profile');

expect(res.statusCode).toBe(401);

});

it('returns profile data for valid Clerk session', async () => {

*// Mock Clerk authentication*

jest.mock('@clerk/nextjs', () => ({

getAuth: () => ({ userId: 'user\_123' })

}));

*// Mock database response*

jest.spyOn(db, 'query').mockResolvedValue({

leetCode: 'test\_user',

isPublic: true

});

const res = await request(app)

.get('/api/profile')

.set('Authorization', 'Bearer valid\_token');

expect(res.statusCode).toBe(200);

expect(res.body).toHaveProperty('leetCode', 'test\_user');

});

});

Execution:

npm run test:api *# Runs 45 API test cases*

## 3. Frontend Component Testing

Tools: Jest, React Testing Library

## **Test Coverage**

| **Component** | **Test Cases** | **Status** |
| --- | --- | --- |
| ProfileTracker | Loading states, visibility toggle | ✅ |
| QuestionTracker | Marking solved, progress calculation | ✅ |
| EventTracker | Contest fetching, reminder setup | ✅ |
| Leaderboard | Sorting, platform filters | ✅ |

## **Sample Test - Profile Visibility Toggle**

describe('Profile Visibility', () => {

it('toggles between public/private states', async () => {

*// Mock Clerk user*

useUser.mockReturnValue({

user: { id: 'user\_123' },

isLoaded: true

});

*// Mock API response*

global.fetch = jest.fn()

.mockResolvedValueOnce({ ok: true })

.mockResolvedValueOnce({

ok: true,

json: () => ({ isPublic: false })

});

render(<ProfileTracker />);

const toggle = await screen.findByRole('checkbox');

await userEvent.click(toggle);

expect(fetch).toHaveBeenCalledWith('/api/toggle-profile-visibility', {

method: 'POST',

body: JSON.stringify({ clerkId: 'user\_123', isPublic: false })

});

expect(await screen.findByText('Private')).toBeInTheDocument();

});

});

Execution:

npm run test:components *# Runs 32 component tests*

## 4. End-to-End Testing

Tools: Cypress

## **Key User Flows Validated**

1. Clerk authentication flow (signup/login)
2. Profile setup with platform connections
3. Question tracking & progress visualization
4. Contest reminder setup
5. Public profile sharing

Sample Test - Full User Journey

javascript

describe('Complete User Flow', () => {

it('signs up, connects LeetCode, solves questions', () => {

cy.visit('/');

cy.loginViaClerk('test@coderstat.com');

cy.connectPlatform('LeetCode', 'test\_leetcode');

cy.markQuestionSolved('Two Sum');

cy.get('[data-testid="solved-count"]').should('contain', '1');

});

});

Execution:

npm run test:e2e *# Runs 12 E2E scenarios*

## 5. Performance Testing

Tools: k6

## **Load Test Results**

| **Metric** | **Value** |
| --- | --- |
| API Requests/sec | 1,200 |
| 95th Percentile Latency | 220ms |
| Error Rate | 0.12% |

Sample Test Script:

javascript

import http from 'k6/http';

export const options = {

stages: [

{ duration: '1m', target: 100 }, *// Ramp-up*

{ duration: '3m', target: 500 }, *// Peak load*

{ duration: '1m', target: 0 }, *// Ramp-down*

],

};

export default function () {

http.get('http://localhost:3000/api/stats');

http.get('http://localhost:3000/api/leaderboard');

}

## 6. User Acceptance Testing (UAT)

12 users validated core functionality:

| **Test Scenario** | **Success Rate** | **Feedback Incorporated** |
| --- | --- | --- |
| Profile Setup | 100% | Added username validation |
| Stats Refresh | 91.6% | Improved loading indicators |
| Question Tracking | 100% | Add url in start button |
| Contest tracker | 100% | Fixed timezone handling |

Key Improvements from UAT:

* Added CSV export for question progress
* Implemented rate limiting on stats refresh
* Enhanced error messages for invalid platform usernames

## 7. Test Coverage & CI/CD

Coverage Report:

-----------------------|---------|----------|---------|---------|-------------------

File | % Stmts| % Branch | % Funcs | % Lines | Uncovered Line #s

-----------------------|---------|----------|---------|---------|-------------------

All files | 92.3 | 85.11 | 88.23 | 91.56 |

GitHub Actions Pipeline:

name: CI

on: [push]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- uses: actions/setup-node@v3

- run: npm ci

- run: npm run test:ci

- uses: codecov/codecov-action@v3

## 8. How to Run Tests

1. Install dependencies:npm install
2. Run all test suites:

npm test *# Runs 89 tests in parallel*

1. Generate coverage report:
2. **npm run coverage**