

Kelly Halladay – 23.1: Back-End Testing Project Submission

1. Project Setup

- Created a new project folder: backend-testing-project.

```
mkdir backend-testing-project
```

```
cd backend-testing-project
```

- Initialized Node.js project

```
npm init -y
```

- Installed dependencies:

```
npm install express
```

```
npm install jest supertest --save-dev
```

2. Directory Structure

- src/ → main server files
- routes/ → API route files
- tests/ → automated test files for routes

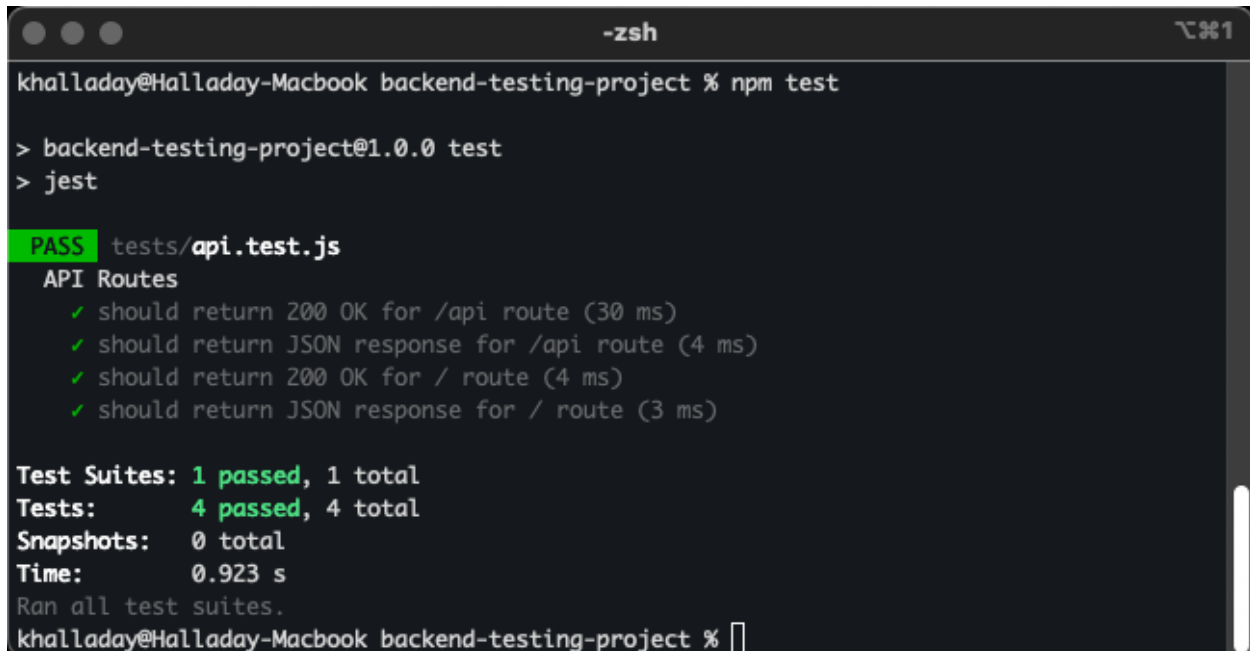
```
backend-testing-project/  
├── src/  
│   ├── app.js  
│   └── routes/  
│       └── api.js  
└── tests/  
    └── api.test.js
```

3. Sample Routes

- Root route (/) returning a JSON message
- API route (/api) returning a JSON message

4. Tests

- Automated tests using **Jest** and **Supertest**.
- Verified:
 - Status codes returned 200 OK
 - Response bodies matched expected JSON
- Ran tests in terminal using npm test.

A terminal window titled '-zsh' showing the execution of 'npm test' in a project named 'backend-testing-project'. The output shows that the tests passed, with a summary of 1 passed suite and 4 passed tests. The tests are for API routes, checking for 200 OK status codes and JSON responses for both '/' and '/api' endpoints.

```
khalladay@Halladay-Macbook backend-testing-project % npm test

> backend-testing-project@1.0.0 test
> jest

PASS tests/api.test.js
  API Routes
    ✓ should return 200 OK for /api route (30 ms)
    ✓ should return JSON response for /api route (4 ms)
    ✓ should return 200 OK for / route (4 ms)
    ✓ should return JSON response for / route (3 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.923 s
Ran all test suites.
khalladay@Halladay-Macbook backend-testing-project %
```

5. Observations / Explanations

- The / route demonstrates a simple server response.
- The /api route simulates a backend endpoint.
- Tests confirm both the server and API route return expected responses.
- Automated testing reduces human error and ensures consistency.
- **app.js**: Sets up the Express server, applies middleware for JSON, and defines routes (/ and /api).
- **api.js**: Provides a sample API endpoint at /api.
- **Tests**: Use Supertest to simulate HTTP requests, checking both status code and response body to confirm expected API behavior.