EM Interview Prep - Comprehensive Testing Framework Documentation

Overview

This document provides complete instructions for running the comprehensive testing framework for the EM Interview Prep application. The testing framework includes unit tests, integration tests, and end-to-end browser automation tests.

Table of Contents

Prerequisites

System Requirements

- Node.js 18+
- npm or yarn package manager
- Modern web browser (Chrome, Firefox, Safari)
- Internet connection for E2E tests

Application Dependencies

- Next.js 14
- React 18
- PostgreSQL database
- Prisma ORM

Test Types

1. Unit Tests (Jest + React Testing Library)

- **Purpose**: Test individual components and functions in isolation
- **Framework**: Jest with React Testing Library
- **Coverage**: Components, utilities, authentication functions
- **Speed**: Fast (< 30 seconds)

2. Integration Tests (Jest)

- **Purpose**: Test API endpoints and database interactions

- **Framework**: Jest with mocked dependencies
- **Coverage**: API routes, database operations, service integrations
- **Speed**: Medium (30-60 seconds)

3. End-to-End Tests (Playwright)

- **Purpose**: Test complete user workflows with browser automation
- **Framework**: Playwright
- **Coverage**: User authentication, navigation, form submissions, search/filtering
- **Speed**: Slow (2-5 minutes)

Setup Instructions

1. Install Dependencies

```
cd /home/ubuntu/em-interview-prep
npm install
```

2. Environment Setup

Ensure the following environment variables are set:

```
DATABASE_URL="postgresql://username:password@localhost:5432/database"
JWT_SECRET="your-jwt-secret"
NEXTAUTH_SECRET="your-nextauth-secret"
```

3. Database Setup

```
# Generate Prisma client
npx prisma generate

# Run database migrations
npx prisma db push

# Seed the database
npm run seed
```

4. Start the Application

```
# Start the development server
npm run dev
```

The application should be running on `http://localhost:3000`

Running Tests

Quick Test Commands

```
# Run all tests (unit + E2E)
npm test

# Run only unit tests
npm run test:unit

# Run only integration tests
npm run test:integration

# Run only E2E tests
npm run test:e2e

# Run tests with coverage report
npm run test:unit:coverage

# Run E2E tests with browser UI
npm run test:e2e:headed

# Run E2E tests with Playwright UI
npm run test:e2e:ui

# Debug E2E tests
npm run test:debug
```

Detailed Test Execution

Unit Tests

```
# Run unit tests with watch mode
npm run test:unit:watch

# Run specific test file
npx jest __tests__/components/login-form.test.tsx

# Run tests matching pattern
npx jest --testNamePattern="login"
```

E2E Tests

```
# Run E2E tests in headed mode (visible browser)
npm run test:e2e:headed

# Run specific E2E test file
npx playwright test tests/auth.spec.ts

# Run tests in specific browser
npx playwright test --project=chromium
```

Test Categories

Authentication Tests

- **Location**: `__tests__/auth.test.ts` , `tests/auth.spec.ts`
- **Coverage**: Login/logout flows, session management, protected routes
- **Key Scenarios**:
- Valid credential login
- Invalid credential rejection
- Session persistence
- Logout functionality

Navigation Tests

- **Location**: `tests/navigation.spec.ts`
- **Coverage**: Header navigation, page routing, URL validation
- **Key Scenarios**:
- All navigation links work
- Pages load without errors
- Proper redirects for protected routes

Component Tests

- **Location**: `__tests__/components/`
- **Coverage**: React component rendering, user interactions, props handling
- **Key Components**:
- LoginForm
- Header
- Question Bank
- System Design components

API Tests

- **Location**: `__tests__/api/`
- **Coverage**: API endpoint responses, error handling, data validation
- **Key Endpoints**:
- `/api/auth/login`
- `/api/companies`
- `/api/questions`
- `/api/system-design-questions`

User Workflow Tests

- **Location**: `tests/comprehensive-e2e.spec.ts`
- **Coverage**: Complete user journeys, form submissions, search functionality
- **Key Workflows**:
- User registration and login
- Question browsing and filtering
- Story creation and management
- System design question exploration

Interpreting Results

Unit Test Results

```
PASS __tests__/components/login-form.test.tsx
✓ should render login form with demo buttons (25ms)
✓ should handle successful demo user login (15ms)
✓ should handle login errors (10ms)

Test Suites: 1 passed, 1 total
Tests: 3 passed, 3 total
Coverage: 85% statements, 80% branches, 90% functions, 85% lines
```

E2E Test Results

```
Running 15 tests using 1 worker

✓ [chromium] › auth.spec.ts:3:1 › Authentication › should login successfully (2s)
✓ [chromium] › navigation.spec.ts:3:1 › Navigation › should navigate to all pages (3s
)
✗ [chromium] › question-bank.spec.ts:3:1 › Question Bank › should filter questions
(5s)

15 passed (30s)
```

Coverage Reports

- **Location**: `coverage/lcov-report/index.html`
- **Metrics**: Statements, Branches, Functions, Lines
- **Thresholds**: Minimum 40% coverage required

Troubleshooting

Common Issues

1. Tests Timeout

Problem: E2E tests timeout waiting for elements
Solution:
- Ensure application is running on correct port
- Check network connectivity
- Increase timeout in `playwright.config.ts`

2. Database Connection Errors

Problem: Tests fail with database connection errors
Solution:
- Verify DATABASE_URL environment variable
- Ensure PostgreSQL is running
- Run `npx prisma db push` to sync schema

3. Module Not Found Errors

Problem: Jest cannot find modules or dependencies
Solution:

- Run `npm install` to ensure all dependencies are installed
- Check `jest.config.js` moduleNameMapper configuration
- Verify import paths use correct aliases

4. Authentication Failures

Problem: E2E tests fail during login
Solution:
- Verify admin user exists in database
- Check login credentials in test files
- Ensure JWT_SECRET is set correctly

Debug Commands

```
# Debug specific test with verbose output
npx jest --verbose __tests__/auth.test.ts

# Debug E2E test with browser inspector
npx playwright test --debug tests/auth.spec.ts

# Generate test report
npx playwright show-report

# Check test coverage details
npm run test:unit:coverage && open coverage/lcov-report/index.html
```

Test Configuration Files

Jest Configuration (`jest.config.js`)

```
module.exports = {
  testEnvironment: 'jest-environment-jsdom',
  setupFilesAfterEnv: ['<rootDir>/jest.setup.js'],
  moduleNameMapper: {
    '^@/(.*)$': '<rootDir>/$1',
  },
  collectCoverageFrom: [
    'components/**/*.{js,jsx,ts,tsx}',
    'lib/**/*.{js,jsx,ts,tsx}',
    'app/**/*.{js,jsx,ts,tsx}',
  ],
  coverageThreshold: {
    global: {
      branches: 40,
      functions: 40,
      lines: 40,
      statements: 40,
    },
  },
}
```

Playwright Configuration (`playwright.config.ts`)

```
export default defineConfig({
  testDir: './tests',
  use: {
    baseURL: 'https://your-preview-url.com',
    trace: 'on-first-retry',
  },
  projects: [
    { name: 'chromium', use: { ...devices['Desktop Chrome'] } },
  ],
})
```

Continuous Integration

GitHub Actions Example

```
name: Tests
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
      - run: npm install
      - run: npm run test:unit:coverage
      - run: npx playwright install
      - run: npm run test:e2e
```

Best Practices

Writing Tests

1. **Descriptive Names**: Use clear, descriptive test names
2. **Arrange-Act-Assert**: Structure tests with clear setup, action, and verification
3. **Mock External Dependencies**: Mock APIs, databases, and external services
4. **Test User Behavior**: Focus on user interactions rather than implementation details

Maintaining Tests

1. **Regular Updates**: Update tests when features change
2. **Clean Test Data**: Ensure tests clean up after themselves
3. **Parallel Execution**: Design tests to run independently
4. **Performance**: Keep tests fast and focused

Support

For issues with the testing framework:
1. Check this documentation first

2. Review error messages and logs
3. Check GitHub issues for known problems
4. Contact the development team

Last Updated: June 16, 2025
Version: 1.0.0