Test Implementation – Automated Test Suite (Cypress)

Project Overview

This document outlines the test implementation for the video streaming application using Cypress. The focus is on automating tests for critical user flows, edge cases, error scenarios, and accessibility compliance.

Tools & Frameworks

- Cypress (for end-to-end UI testing)
- Node.js (environment)
- Optional: `cypress-axe` for accessibility checks

Test Structure

```
cypress/
├── e2e/
│   ├── login.cy.js            # Login and authentication tests
│   ├── video.cy.js            # Video playback and control tests
│   ├── bookmarks.cy.js          # Bookmark edge case scenarios
│   ├── accessibility.cy.js      # Accessibility tests using keyboard and contrast checks
```

Covered Test Scenarios

1. Video Playback (Play button, playback speed, quality)
2. Login with valid and invalid credentials
3. Edge cases (empty search, long comments)
4. Error handling (broken video link, bookmarking issues)
5. Accessibility (keyboard navigation, color contrast)

Sample Test Snippet – Login Error Handling

```
describe('Login Error Handling', () => {
  it('should show error for invalid credentials', () => {
    cy.visit('/login');
    cy.get('#email').type('wrong@example.com');
    cy.get('#password').type('wrongpass');
    cy.get('button[type=submit]').click();
    cy.contains('Invalid credentials').should('be.visible');
  });
});
```

Notes & Best Practices

- Use meaningful selectors for stable tests.
- Use `beforeEach` for setting up test context.
- Keep tests short and focused (one assertion per test ideally).
- Use plugins like `cypress-axe` to enhance accessibility testing.
- Organize tests by feature or functionality.

Time	**Task**
30 mins	Setup Cypress & project structure
1.5 hours	Write automated tests (critical flows, edge cases, errors, a11y)
30 mins	Document & comment tests
30 mins	Run, debug, and clean up test suite

Step 1: Setup Cypress

bash

CopyEdit

```
npm install cypress --save-dev

npx cypress open
```

Create your test file inside:

bash

CopyEdit

```
cypress/e2e/video_app_spec.cy.js
```

Step 2: Write Cypress Test Cases

1. Critical User Flow – Video Playback

js

CopyEdit

```
describe('Video Playback Flow', () => {
  it('should play a selected video', () => {
    cy.visit('/');
    cy.contains('Play').click(); // adjust selector
    cy.get('video').should('have.prop', 'paused', false);
  });
});
```

2. Login Error Scenario

js

CopyEdit

```
describe('Login Error Handling', () => {
  it('should show error for invalid credentials', () => {
    cy.visit('/login');
    cy.get('#email').type('wrong@example.com');
    cy.get('#password').type('wrongpass');
    cy.get('button[type=submit]').click();
    cy.contains('Invalid credentials').should('be.visible');
  });
});
```

3. Edge Case – Empty Search Input

js

CopyEdit

```
describe('Search Input Edge Case', () => {
  it('should warn on empty search', () => {
    cy.visit('/');
    cy.get('#searchButton').click(); // adjust selector
    cy.contains('Please enter a search term').should('be.visible');
  });
});
```

4. Bookmark Without Loaded Video

js

CopyEdit

```
describe('Bookmark Edge Case', () => {
  it('should not allow bookmarking without video', () => {
    cy.visit('/');
    cy.get('#bookmarkButton').click();
    cy.contains('Load video to bookmark').should('be.visible');
  });
});
```

5. Accessibility Test – Keyboard Navigation

js

CopyEdit

```
describe('Accessibility Test', () => {
  it('should allow tab navigation through buttons', () => {
    cy.visit('/');
    cy.get('body').tab(); // needs cypress-plugin-tab
    cy.focused().should('have.attr', 'role');
  });
});
```

Organize Tests

bash

CopyEdit

```
cypress/
  e2e/
    login.cy.js
    video.cy.js
    bookmarks.cy.js
```

accessibility.cy.js

Each test file:

- Tests one feature
- Uses beforeEach() for clean setup
- Has comments to explain logic

Step 3: Add Comments & Documentation

js

CopyEdit

```
// Test: Should play a video when "Play" button is clicked
// Priority: Critical user flow
// Author: Mahalakshmi R
```

Include this on top of each test case.

Run Tests

bash

CopyEdit

```
npx cypress run
```

Or with UI:

bash

CopyEdit

```
npx cypress open
```

Add Accessibility Plugin

bash

CopyEdit

```
npm install cypress-axe --save-dev
```

Then in your test:

js

CopyEdit

```
import 'cypress-axe';
```

```
describe('Accessibility', () => {
  it('should pass WCAG checks', () => {
    cy.visit('/');
    cy.injectAxe();
    cy.checkA11y();
  });
});
```