Session 34: Angular Development Deep Dive

This session covers key aspects of Angular development, including UI components, testing methodologies, deployment strategies, and performance optimization.-----Angular Material & UI Components

Angular Material, built and maintained by the Angular team, adheres to Google's Material Design principles. It offers a comprehensive suite of responsive, accessible, and themeable UI components.

Benefits:

- Speeds up UI development with consistent styling.
- Mobile-friendly out-of-the-box.

Testing in Angular

Testing is crucial for ensuring application quality and maintaining developer confidence.

Importance of Testing:

- Ensures features function as expected.
- Catches bugs early in the development cycle.
- Supports refactoring and continuous integration/continuous delivery (CI/CD) pipelines.
- Builds user confidence in the application.

Types of Testing:

- **Unit Testing:** Focuses on testing individual components or services in isolation.
- **End-to-End (E2E) Testing:** Simulates real user interactions across the entire application.

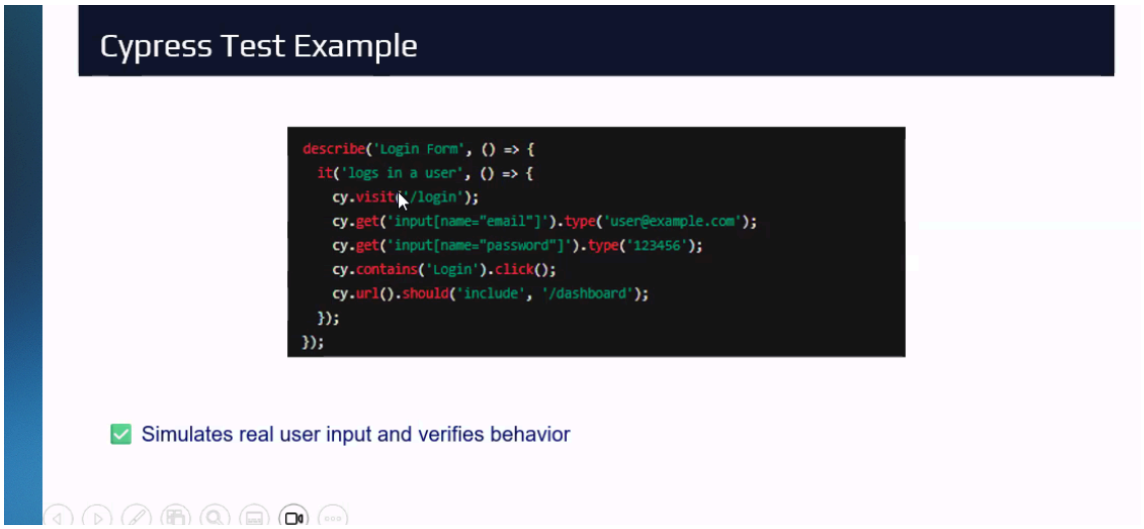
Unit Testing Tools & Practices:

- **Tools:** Jasmine (testing framework) and Karma (test runner). Both are included by default with the Angular CLI.
- **File Naming:** Test files are auto-generated with a `.spec.ts` extension.
- **Running Tests:** Use the `ng test` command, which launches the Karma test runner in a browser for fast feedback during development.

Jasmine Basics:

- `describe()`: Defines a test suite (a group of related tests).
- `it()`: Represents a single test or "spec" and describes an expected behavior.
- `expect()`: Used to define an assertion or expectation for the test.
- `beforeEach()`: A hook that runs before each test within a suite.
- `spyOn()`: Used to track and mock function calls.

E2E Testing Tools & Practices:

- **Cypress:** A modern, widely used E2E testing tool that allows you to observe the application being tested in a browser.
    - *Note: Cypress testing for this session was successful in Edge, but not Chrome.*
- **Protractor (Legacy):**
    - An older, Angular-specific E2E tool.
    - Utilizes WebDriver (Selenium-based).
    - Currently being deprecated in favor of Cypress and Playwright.
- 

Testing Best Practices:

- Write one `it()` block per distinct behavior.
- Avoid testing private methods as they are internal implementation details.
- Use `spyOn()` to mock services for isolated testing.

Deployment and Optimization

Deployment Optimizations:

- **AOT (Ahead-of-Time) Compilation:** Converts Angular templates to efficient JavaScript before the browser loads, leading to faster application loading times.
- **Minification and Bundling:** Removes unnecessary code, whitespace, and comments, and combines files to reduce application size.
- **Tree-shaking:** Eliminates unused code ("deadweight") from the final bundle.
- **Lazy Loading:** Breaks the application into smaller chunks, loading only what's needed for the initial screen, improving perceived performance.

Deployment Output:

- The optimized output is generated in the `/dist` folder, containing static files ready for deployment.

Common Hosting Platforms:

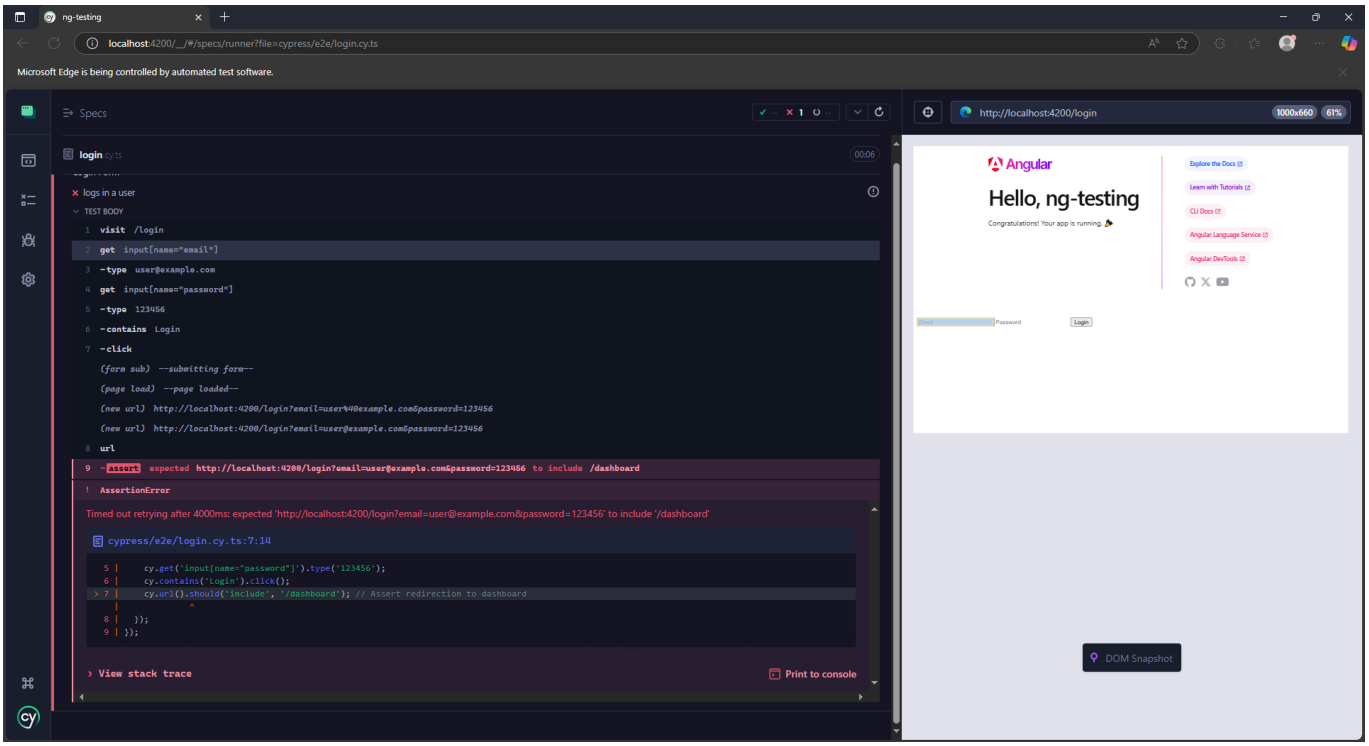
- Firebase Hosting
- Netlify / Vercel
- AWS S3 + CloudFront
- GitHub Pages
- Custom servers (e.g., Node.js, Nginx)

Performance and Optimization Tips:

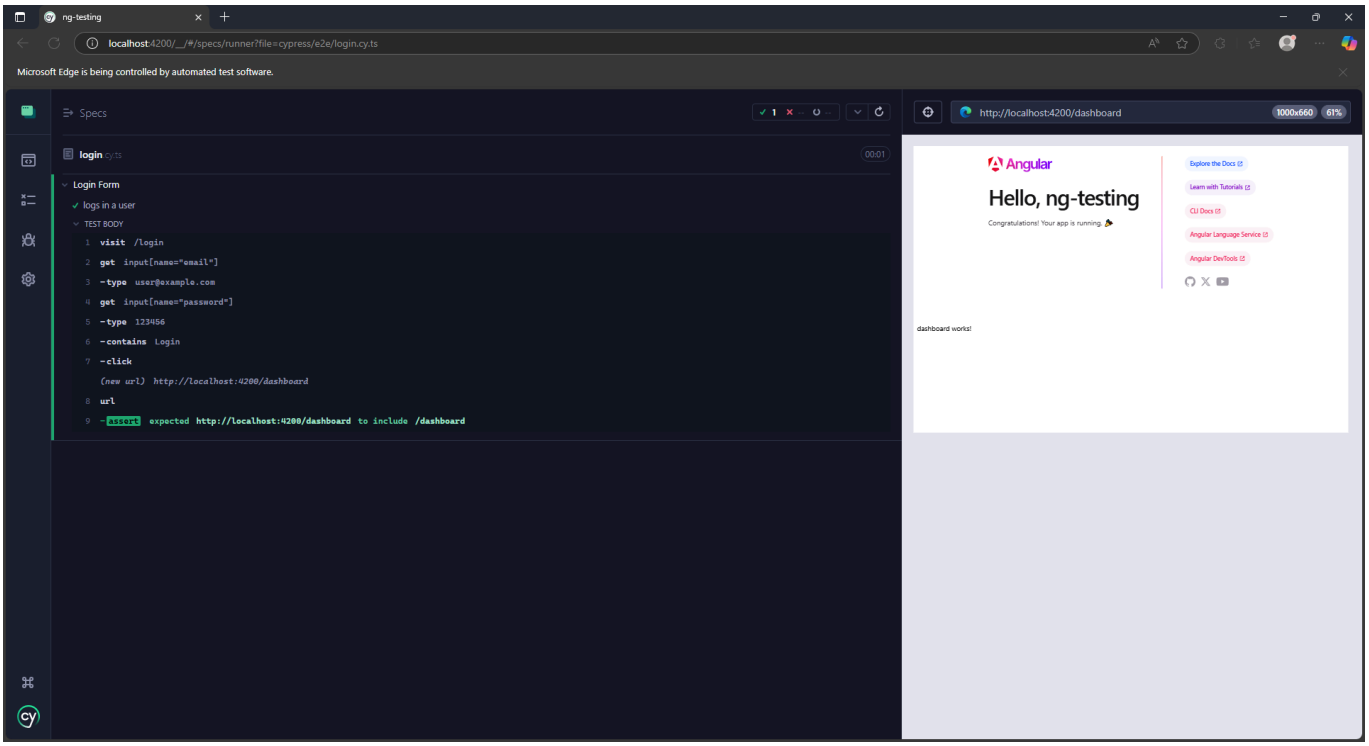
- Implement lazy loading for modules.
- Utilize preloading strategies for anticipated routes.
- Compress images and other assets.
- Use `trackBy` in `*ngFor` directives to optimize list rendering.
- Minimize the use of third-party scripts.

E2E Log

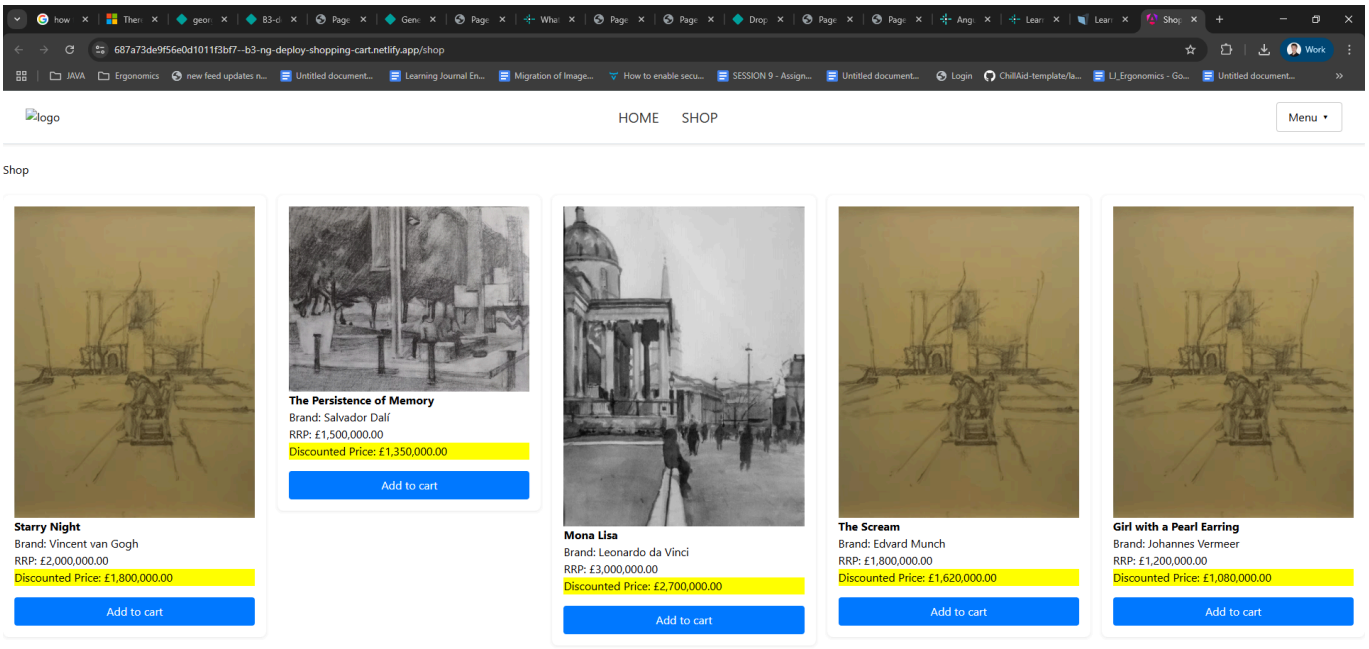
Cyprus Testing



Successful logs:
Note this only worked in Edge, not Chrome



Successful Deploy to Netlify



Live Site URL:

https://687a73de9f56e0d1011f3bf7--b3-ng-deploy-shopping-cart.netlify.app/shop

GitHub Repos:

https://github.com/grs-se/B3-week-34