Diploma Project: Testing Strategy for Automaspec

1. Project Overview & Testing Approach

This project focuses on the comprehensive testing of **Automaspec**, a web application designed to manage test specifications and requirements. The testing approach is **Mixed (Qualitative and Quantitative)**, aiming to ensure both the functional reliability of the system and a high-quality user experience.

- **System Under Test (SUT)**: Automaspec (Next.js 15, React 19, Drizzle ORM, Better Auth).
- **Testing Type**: Structured Functional Testing (Quantitative) and Usability/UX Testing (Qualitative).

2. Scope and Goals

Test Scope

The following components are within the scope of testing: - **Authentication**: User registration, login, session management (via Better Auth). - **Organization Management**: Creating organizations, inviting members, managing roles. - **Dashboard UI**: Navigation, data visualization, and interactive elements. - **Database Operations**: Schema validation (Drizzle), data persistence (LibSQL/Turso). - **API/RPC Layer**: oRPC integration and request handling.

Testing Goals & Objectives

- **Goal**: Validate that Automaspec is reliable, secure, and user-friendly.
- **Objectives**:
  - Achieve >80% unit test pass rate for critical components.
  - Verify data integrity for all CRUD operations on Organizations.
  - Identify and document at least 3 usability improvements for the Dashboard.
  - Ensure zero critical defects in the Authentication flow.
- **Entry Criteria**: App builds successfully via `pnpm build`; Local DB is reachable.
- **Exit Criteria**: All localized Critical and High priority defects are resolved.

3. Requirements Analysis

The testing process is based on the following feature breakdown: 1. **Auth System**: Users must be able to sign up, sign in, and maintain sessions securely. 2. **Multi-Tenancy**: Users can belong to multiple organizations; data must be isolated. 3. **Spec Management**: Users can create, update, and delete test specifications (core business logic).

4. Manual Test Cases (Sample Artifacts)

The project includes a suite of manual test cases. A sample set includes:

ID	Title	Preconditions	Steps	Expected Result	Priority
TC-01	User Login	User exists in DB	1. Navigate to `/login`2. Enter credentials3. Click Submit	Redirect to Dashboard; Session active	Critical
TC-02	Create Org	Logged in	1. Click "New Org"2. Enter Name "TestOrg"3. Save	"TestOrg" appears in switcher	High
TC-03	Invite Member	Org Admin	1. Go to Settings > Members2. Invite email3. Send	Invite email triggered (mocked)	Medium

5. Quantitative Metrics

For the quantitative aspect, the following metrics are defined and measured: - **Test Execution Time**: Total time for `pnpm test` (Target: <30s for unit tests). - **Defect Density**: Number of bugs found per feature. - **Pass Rate**: Percentage of passed automated tests (Vitest). - **Code Coverage**: Lines of code covered by unit tests (Target: High coverage for `lib/utils` and API hooks).

6. Qualitative Findings (Usability & UX)

Focus areas for qualitative testing: - **Consistency**: Verifying usage of Design System (Tailwind v4) and typography. - **Flow Analysis**: "Create Specification" flow analysis to ensure minimal clicks. - **Feedback**: Error messages for failed DB operations must be user-friendly.

Documented Findings: - Initial review suggests the "Organization Switcher" needs clearer visual indication of the active state. - Mobile responsiveness for the main Dashboard table needs optimization.

7. Test Execution & Reporting

- **Defect Tracking**: All defects are logged with steps to reproduce and severity.
- **Execution Report**: A final report summarizes the test run, including:
  - Environment: Local Dev (Next.js Turbo), LibSQL (Local).
  - Status: Passed/Failed count.
  - References: Links to Github Issues/Commit IDs.

8. Professional Testing Tools

The following professional tools are integrated into the workflow: - **Vitest**: Used for Unit and Component testing. Configuration in `vitest.config.mts`. - *Purpose*: Fast, headless testing of React components and logic. - **Postman / Insomnia**: (Optional) For manual API verification of oRPC endpoints. - **Lefthook**: Manages pre-commit hooks to enforce testing before code push.

9. Version Control & Reproducibility

- **Git Repository**: The project is hosted in a Git repository with a meaningful commit history.
- **Reproducibility**:
  - `README.md` details setup instructions (`pnpm install`, `pnpm db:local`).
  - Environment variables documented in `.env.example`.

10. Final Technical Report Summary

The final diploma report will consolidate: 1. **Tested System**: Automaspec v1.0. 2. **Strategy**: Hybrid manual/automated approach. 3. **Conclusion**: Summary of stability and readiness for release based on metrics.