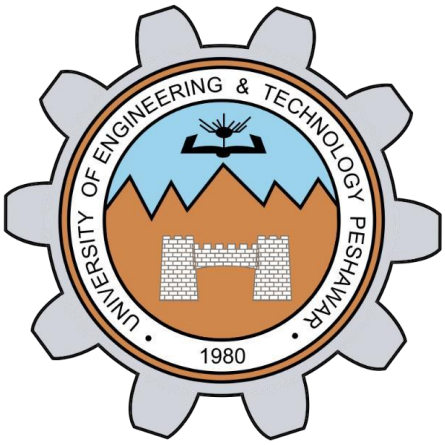
UNIVERSITY OF ENGINEERING AND TECHNOLOGY, PESHAWAR PAKISTAN

Main Campus



Software Engineering Lab

Assignment 4

Name:	**Muhammad Mohsin**
Registration No.	**23PWBCS0973**
Semester:	**BS CS 5th**
Section:	**A**

Submitted To : Miss Kanwal Aneeq

DEPARTMENT OF COMPUTER SCIENCE & IT

UNIVERSITY OF ENGINEERING AND TECHNOLOGY, PESHAWAR, PAKISTAN

Course: Software Engineering
Topic: Requirement Analysis
Group Members:

1. Abdul Baseer
2. Muhammad Mohsin
3. Saad Abdullah
4. Hooriya Altaf

Part 1: Requirement Elicitation

User Requirements:

1. The system should allow users to load a website simply using its URL.
2. The system should scan a given website and report common WCAG 2.1 AA accessibility violations.
3. The system should simulate how a site appears to users with visual impairments (color blindness, low vision, high contrast).
4. The system should provide the ability to navigate using both keyboard and mouse.
5. The system should allow the generation of an exportable PDF containing the findings.
6. The system should allow users save and manage their project data, thus report data would be saved to a database.
7. The system should allow role-based access: Tester, Admin.
8. The system should return scan results within a reasonable time frame.
9. The live simulation should have low response latency so as to keep the experience smooth.
10. The system should provide security for the users' data and reports with secure authentication and HTTPS.
11. The system should be responsive and usable from mobile and desktop devices.
12. The system should include an API to trigger scans and retrieve results programmatically also local websites(in development) should also be allowed to run with custom options for experienced testers .

Part 2: Requirement Categorization

ID	Requirement Description	Type (F/NF)	Source	User / System	Ambiguity / Conflict	Resolution
1	load a website simply using its URL	**F**	Accessibility Tester	User	No ambiguity	**N/A**
2	Scan website URL and report WCAG 2.1 AA violations	**F**	Accessibility Tester	User	Ambiguity: which WCAG level exactly (AA) and which rules included	Specify: WCAG 2.1 AA full rule set; list included rules in help docs.
3	Simulate visual impairments (color blindness, low vision, high contrast)	**F**	Accessibility Tester	User	Ambiguity: which simulation modes and severity levels	Define supported simulations: protanopia, deuteranopia, tritanopia, grayscale, high-contrast; document limitations.
4	Allow interaction using both keyboard and mouse for simulation	**F**	Accessibility Tester	User	Ambiguity: how much interaction is allowed	Define what type of interaction is not allowed due to technical/legal issues.
5	Generate exportable report PDF	**F**	Accessibility Tester	User + System	Conflict: level of detail vs. report size	Offer configurable report verbosity (summary vs. full) and include filters.
6	Save and manage projects (metadata, past scans)	**F**	Accessibility Tester	System	None	Clear.
7	Role-based access: Tester, Developer, Manager	**NF** (security & usability)	IT/Admin	System	Ambiguity: exact permissions per role	Define permission matrix (e.g., Tester: run scans;

						Developer: view reports & code snippets; Manager: export + manage users).
8	Return scan results within reasonable time	**NF** (performance)	Tester/Admin	System	Ambiguity: "reasonable" depends on infra and page complexity	Define targets, e.g., 20 pages/min for typical pages; provide progress indicator and queueing.
9	low response latency for live simulation	**NF** (Performance)	IT Staff	System	Ambiguity: Low latency depends upon system complexity .	Define latecy cap: latency should be less than 500ms. i.e. the user should see the result of their interaction within 500ms.
10	Protect uploaded content and reports with secure auth and HTTPS	**NF** (security)	IT Staff	System	Ambiguity: auth method and encryption standards	Specify: OAuth2 / JWT for auth, TLS 1.2+, encryption-at-rest for sensitive files.
11	Responsive UI for mobile and desktop	**NF** (usability)	Tester	User	No Ambiguity	N/A
12	Provide an API to trigger scans and retrieve results programmatically	**F**	Tester	System	Ambiguity: auth and rate limits	Define API auth (API keys / OAuth2), rate limits and pagination; document endpoints and schemas.

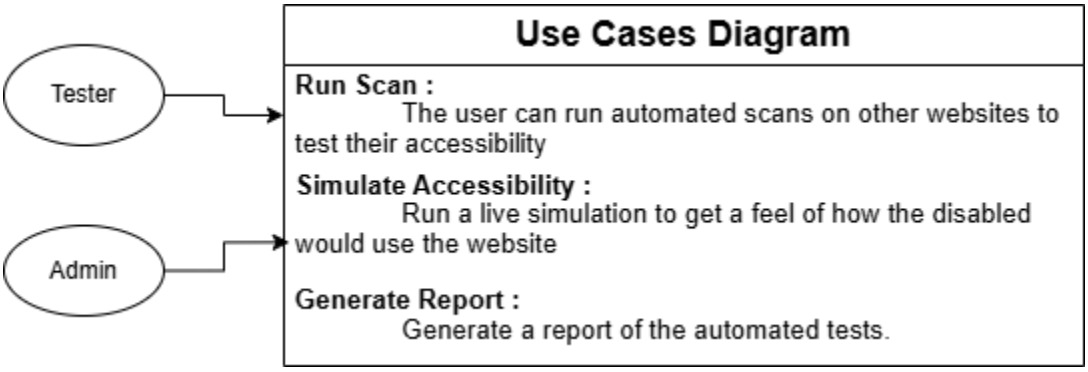
Notes on Ambiguous / Conflicting Requirements & Resolutions

- **Ambiguity around "WCAG coverage":** resolved by explicitly supporting WCAG 2.1 AA rule set and listing supported checks in documentation.
- **Performance target vagueness:** resolved by defining a baseline SLA (e.g., 20 pages/min average) and adding progress/queue indicators. Live simulation to have latency < 500ms.

- **Reporting detail vs. size conflict:** resolved by adding configurable report verbosity (summary vs full) and filters for developers/managers.

Part 3: Requirement Modeling

Use Case Diagram:



Use Case Descriptions (2–3 lines each)

Use Case A — Run Scan
The Tester provides a URL, uploads a page, or selects a saved project and requests a scan. The system crawls the page(s), runs automated accessibility checks against WCAG 2.1 AA rules, and stores results in its database.

Use Case B — Simulate Accessibility
The Tester or Developer selects a simulation mode (color-blindness, keyboard-only, high-contrast) and the system renders an interactive preview showing how the site behaves under the chosen impairment.

Use Case C — Generate Report
The Developer requests an exportable report. The system compiles findings into a PDF. It also allows selecting verbosity level (summary/full).

Part 4: Reflection

The primary challenge we encountered during requirement elicitation was balancing automated checks with manual verification needs. Many accessibility problems cannot be fully validated automatically — users expect the simulator to be "perfect," which requires clarifying scope and limitations.

The hardest requirements to define were performance (scan speed) and coverage (which WCAG rules are checked and how deeply). These depend heavily on implementation choices (crawler depth, JavaScript execution, headless browser configuration) and infrastructure capacity. We resolved this by proposing clear limits and documenting which checks are automated versus those recommended for manual testing.

Poor requirement analysis for this project could cause serious issues: the team might implement incorrect checks, produce misleading reports, or fail to meet usability and security expectations. That would reduce trust in the tool and might lead to noncompliance in sites expected to be accessible. Clear, testable, and prioritized requirements reduce rework and improve adoption.