Software Requirements Specification

for

ARIA

Advanced Requirements Intelligence & Analytics

Version 1.0 approved

Prepared by Crackers Team

15.10.25

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Initialization	15.10.25	-	1.0
Data model and UML case update	08.11.25	1) Backend model architecture is decided 2) Finalized case for features	1.1

1. Introduction

ARIA is an AI-powered tool designed to automate and enhance requirements prioritization for mid-sized software development, product management, and project management projects. This document serves as a foundational agreement among stakeholders regarding the scope, features, and constraints of the ARIA system.

1.1. Purpose

The purpose of this SRS is to provide a comprehensive and unambiguous description of the software to be developed for ARIA. It aims to:

- Clearly define the capabilities and limitations of the ARIA system.
- Serve as a reference for all development, testing, and project management activities.
- Form the basis for future project phases, including design, implementation, and verification.
- Ensure that the developed software meets the business objectives and user needs outlined in the Vision and Scope Document.

1.2. Document Conventions

SF<number> - *system feature*
FR<number> - *functional requirement (reference in pair with SF, i.e. SF1.FR2)*
NF<number> - *non-functional requirement*
D<number> - *dependency*
A<number> - *assumption*
QA<number> - *quality attribute*
NOT MVP - *not included in MVP, delving into details is not a priority at the current stage*
Optional - *optional for project at the current stage*
TBD - *to be decided (currently cannot provide decision)*
NICE TO HAVE - *optional for MVP*

1.3. Project Scope

The ARIA project aims to deliver a proof-of-concept AI tool that automates requirements prioritization. The initial release will focus on a single-user web application with core prioritization functionalities
See *Vision and Scope for ARIA Project* for comprehensive information.

1.4. References

Milestone 1 / Vision and Scope for ARIA Project available at github
Scenarios / scenarios.excalidraw available at github
Scenarios / sf-cases.excalidraw available at github

2. Overall Description

2.1. Product Perspective

ARIA is a web-based AI tool designed to assist in the complex process of requirements prioritization. It will not initially be integrated with existing requirements management platforms but rather serve as a complementary analytical tool. Users will interact with ARIA by providing requirement data, which the system will then process to generate prioritized outputs. The system's primary function is to offer an objective, data-driven approach to prioritization, addressing the limitations of traditional manual methods which are often time-consuming and subjective, particularly for projects with numerous requirements.

2.2. User Classes and Characteristics

User Characteristics:
- Startups, Medium-size developer teams:
  - Characteristics: Curious, cost-sensitive, focused on quick prototyping and actionable insights.
  - Major Interests: Lightweight prioritization tool for early-stage product development.
  - Constraints: Limited support, MVP scope.
  - Use Case: Rapidly prioritizing features for new product development or iterating on existing product backlogs with limited resources.

Assumptions for User Characteristics:
- Users will have a basic understanding of requirements management concepts.
- Users will be comfortable with web-based applications.
- Users will be working with English-only requirement text.

2.3. Operating Environment

MVP:
- **Deployment Platform:** Docker containers deployed on either Heroku's free tier or university servers (either any other cloud-based provider by agreement)
- **Client Access:** Web browsers, any OS (Windows, Macos, Linux)
- **Resource Constraints:**
  - Limitation on CPU, RAM
  - Third-party provider for LLM

2.4. Design and Implementation Constraints

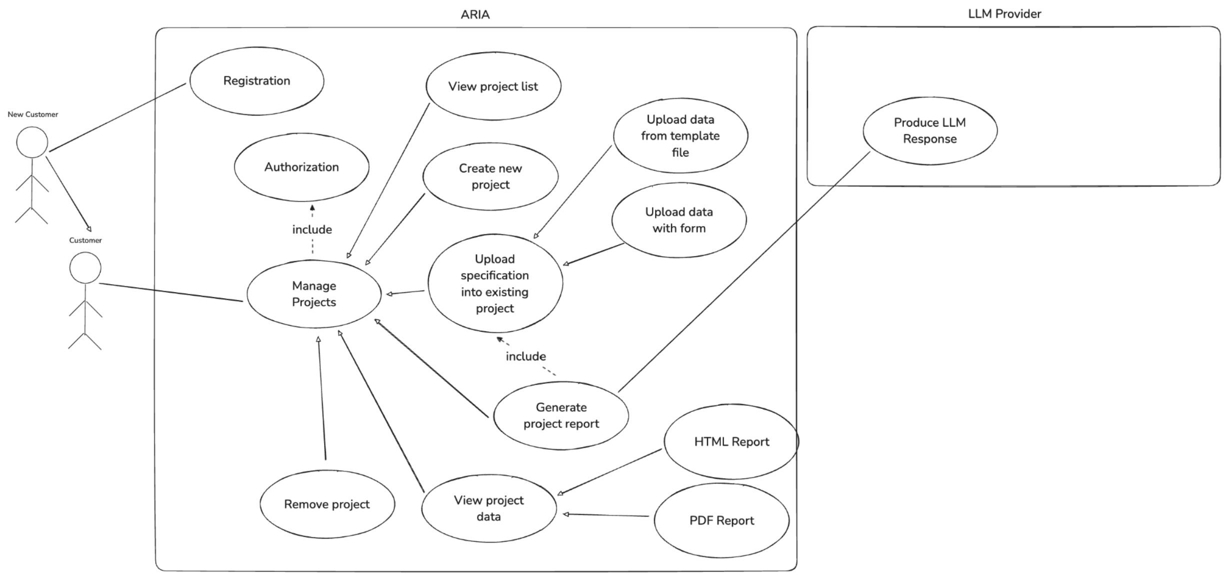
- Technology Stack
  - Docker for deployment
  - Flutter for web interface
  - Open-source LLMs / Third-party LLM providers
  - Python (FastApi) for backend service
  - Postgres as database
- Performance:
  - Processing time for 100 requirements must be ≤5 seconds
  - Must handle 50-100 requirements for the MVP demo
- Quality:
  - Ensure a stable demo with ≤3 critical bugs.
  - Aim for ≥70% correlation with expert ranking on a 30-item test set.
  - Achieve ≥80% positive usability feedback.

2.5. Assumptions and Dependencies

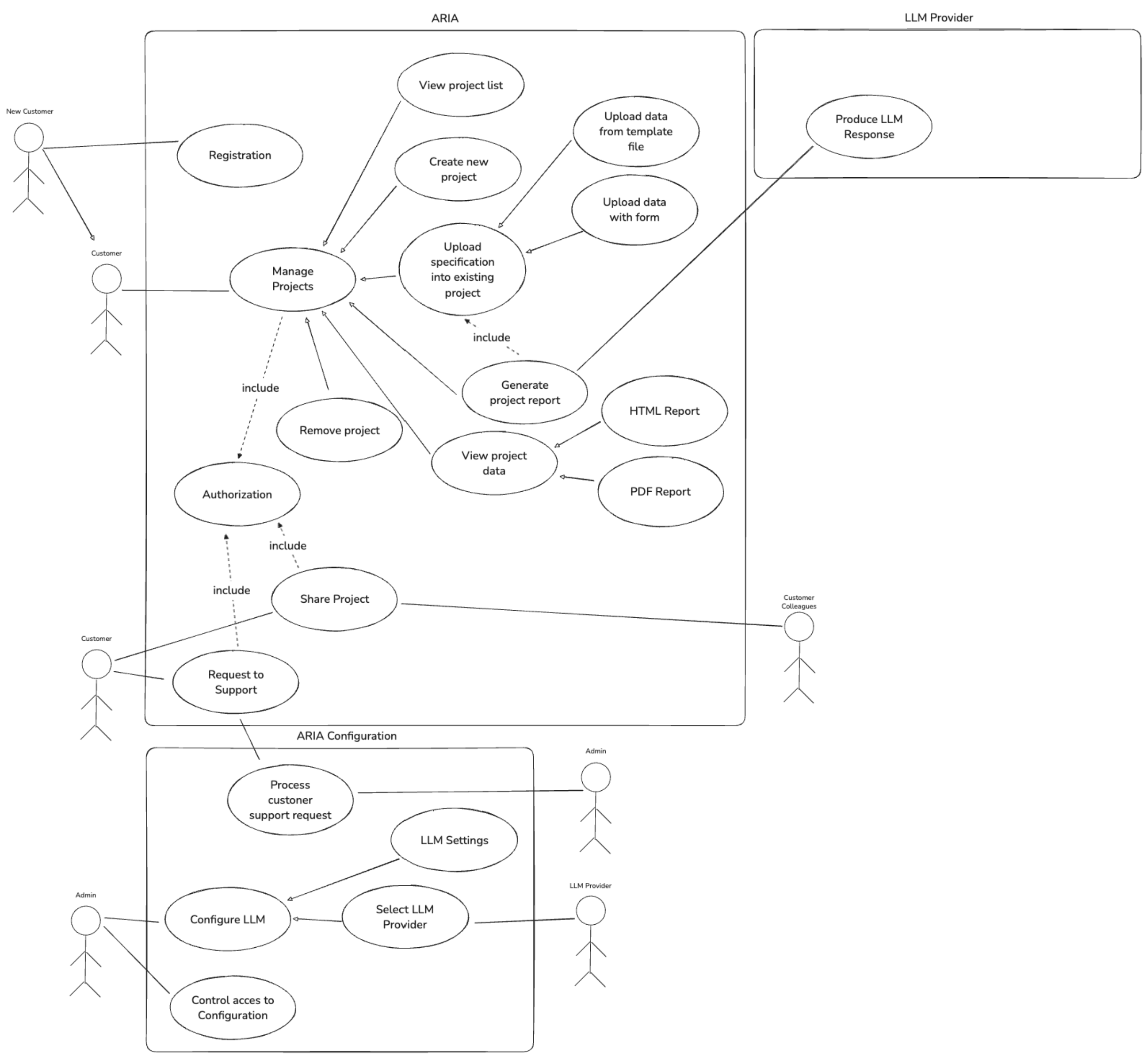
- **D1**: The stability and performance of a product depends on the stability and performance of the LLM provider.
- **D2:** Stability of resources of selected cloud provider

3. System Features

1) UML case diagram for MVP



2) UML case diagram with extras



3.1. SF1: Registration and Authorization

3.1.1. Description

Low priority. For MVP any flow is good. The simplest method is fine - login and password. For clients.

3.1.2. Stimulus/Response Sequences

Case: User would like to get to the his/her projects / profile

Actions: Login / Signup button -> Login / Signup Form contains

- Login:
    - Username field

  - Password field
- Signup:
  - Username field
  - Password field
  - Repeat password field

Result: Projects and Profile pages are available

3.1.3. Functional Requirements

- FR1: User has profile page with personal / billing data
- FR2: User has projects page, which consists his/her project list

3.2. SF2: Project Management

3.2.1. Description

High Priority due to high effects on data and interface. Each user should have access to his/her project list to manage them. For clients.

3.2.2. Stimulus/Response Sequence

Case: User would like to create/view/update/remove project(s)

Prerequirements: User is authorized

Actions:

Create	**View**	**Update**	**Remove**
Open projects page-> Click button create -> Fill basic form (project name, description fields)	Open projects page -> select and click on project	Open projects page -> Click edit button -> Fill form	Open projects page -> Click remove button -> Ensure removal by accepting dialog message
New project is created and added to list	List of projects has been viewed	Project is updated	Project is deleted

3.2.3. Functional Requirements

- FR1: Dedicated page for projects
- FR2: Dedicated page for each clients project
- FR3: CRUD operations over projects via interface

- FR4: TBD how many projects each client can have

3.3. SF3: Project Requirements

3.3.1. Description

High priority. User can setup requirements for the project by filling form manually or by importing text file with specific format (from the template).

3.3.2. Stimulus/Response Sequence

Prerequirements: User went to the project page filled with initial data

Actions:

Download template file	**Upload file with requirements**	**Fill specific form**
Click “Download template”	Click “Upload requirements”-> select file with requirements	Click “Fill requirements” -> type requirements in fields
File with template is downloaded	Server accepts requirements and produces result to the project	

3.3.3. Functional Requirements

- FR1:
    - Input data:
        - Cost (1-10)
        - Rist (1-10)
        - Business (1-10)
        - Dependency (R1 -> R2)
        - Description for requirement
        - Requirement
        - Mandatory / Nice to have (NOT MVP)
    - Output data:
        - HTML Report
            - Chart (requirement / priority score)
            - Chart (cost / value)
            - Priority of each requirement
            - Dependency view graph
        - Export to pdf (NICE TO HAVE)

3.4. SF4: Billing Info

NOT MVP

3.5. SF5: Support

Optional, NOT MVP

3.6. SF6: Administration. Users

NOT MVP

3.6.1. Description

Admin can configure users and their permissions to configure the project

NOT MVP

3.7. SF7: Administration. Configuration LLM provider

NOT MVP

3.7.1. Description

Admin can configure LLM model and provider from selection in admin-panel

3.7.2. Stimulus/Response Sequence

Case: Admin would like to setup available LLMs for customers

Prerequirements: Admin is authorized into admin panel

Actions:

- Configure LLMs providers
    - Find config LLM_PROVIDERS
    - Edit config either through json-text, either through special form
    - Apply

Result: Settings were applied

3.7.3. Functional Requirements

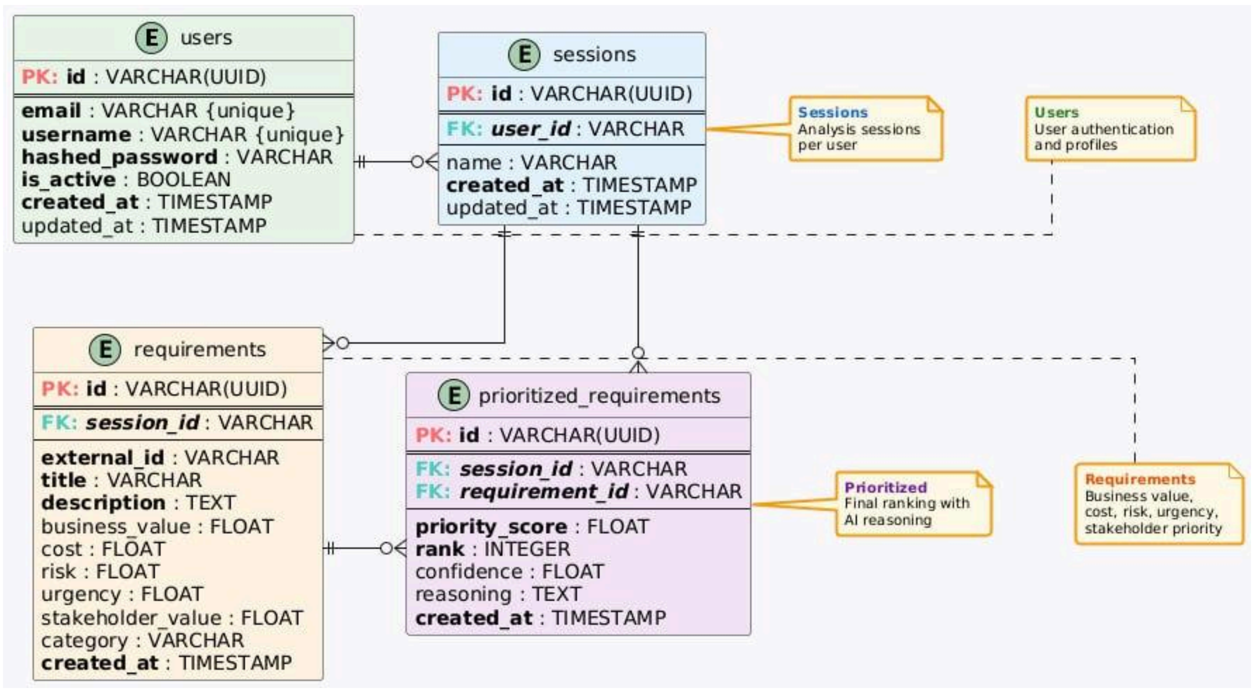
- FR1: Admin has config for LLM providers and models
- FR2: Input data
    - Url
    - Credentials (token)

3.8. SF8: Business. Payments

NOT MVP

4. Data Requirements

4.1. Logical Data Model



4.2. Data Dictionary

-

4.3. Reports

See 3.3.3 for input and output data.

4.4. Data Acquisition, Integrity, Retention, and Disposal

- Postgres

5. External Interface Requirements

5.1. User Interfaces

Interface for Clients
- Landing
  - Hero page
  - Login / Signup Button
- Login / Signup Page
  - Simple form with required fields
- Projects Page
  - List of the projects
- Project Page
  - Input form
  - Upload requirements button
  - Download template button
  - History
  - Reports
- Profile Page
  - Username
  - Requests made

Interface for Administration NOT MVP
- Admin panel
  - LLM Provider page
    - LLM Model and Provider configuration
  - Staff page
    - Users and permissions

5.2. Software Interfaces

- External Data Sources (User Uploads):
  - Purpose: To ingest raw requirements data for prioritization.
  - Components: User's local file system / chosen file upload mechanism.
  - Data Exchange:
    - Input: CSV files (comma-separated values) and Microsoft Excel files (.xlsx, .xls).
    - Format: Tabular data, where each row represents a requirement and columns represent attributes (e.g., ID, Description, Business Value, Risk, Effort).
    - Content: Textual descriptions of requirements, numerical or categorical values for prioritization criteria.
    - Mapping/Translations: The system will parse CSV/Excel data, mapping column headers to internal requirement attributes. Basic data type conversions (e.g., text to numerical for scoring) will be performed. Error handling for malformed files or missing critical columns will be implemented.

- Communication Protocol: Standard HTTP/HTTPS for file upload via the web interface.
- Non-functional Requirements:
  - NF1. Security: Input validation will be performed to prevent malicious file uploads or injection attacks.
  - NF2. Frequency: On-demand, initiated by the user.
  - NF3. Response Time: File upload and initial processing feedback (e.g., "File uploaded successfully") should be near-instantaneous for typical file sizes (<1MB).

- Machine Learning (ML) Models:
  - Purpose: To perform intelligent analysis and scoring of requirements.
  - Components: Open-source LLM (Large Language Model) models, as per business assumptions.
  - Data Exchange:
    - Input: Structured text (requirement descriptions) and associated metadata (priority criteria values) extracted from user uploads.
    - Output: Numerical scores, weights, or classifications relevant to prioritization.
    - Mapping/Translations: Internal requirement objects will be transformed into the input format expected by the specific ML model APIs/libraries. Model outputs will be translated back into standardized internal scoring metrics.
  - Communication Protocol: Internal library calls or potentially RESTful API calls if models are hosted as separate services (e.g., Flask API for LLM).
  - Non-functional Requirements:
    - NF4. Response Time: Processing time for 100 requirements by the ML model must contribute to the overall ≤5 seconds performance target for the system.
    - NF5. Security: If external model APIs are used, appropriate API key management and secure communication (HTTPS) will be employed.
    - NF6. Reliability: The ML model inference engine must be stable and produce consistent results for identical inputs.

- Database (Internal):
  - Purpose: To store uploaded requirements, processing results, and system configuration.
  - Components: An internal database system (e.g., SQLite for simplicity in MVP, or PostgreSQL/MySQL if deployed on Heroku).
  - Data Exchange:
    - Input: Parsed requirement data, prioritization scores, user session data (for single-user context).
    - Output: Retrieved requirement data, scores for visualization and export.
  - Communication Protocol: Standard database client drivers/APIs (e.g., SQLAlchemy in Python).
  - Non-functional Requirements:
    - NF7. Security: Basic data isolation for the single-user MVP. Enterprise-level security, including encryption, is deferred.
    - NF8. Backup: Weekly automated backups of the database will be performed.
    - NF9. Integrity: Data consistency and integrity will be maintained.

5.3. Hardware Interfaces

- Standard Server Hardware:
  - The software will run on standard server hardware provided by the Heroku free tier or university servers.

- Characteristics: These typically include CPUs (for processing), RAM (for memory), and persistent storage (for database and application files).
- Data and Control Interactions: The software interacts with this hardware via the operating system's APIs for CPU scheduling, memory allocation, and disk I/O.
- Inputs/Outputs: These are managed by the operating system; the software consumes CPU cycles, memory, and disk space as needed.
- Timing Issues: The ≤5 seconds processing time for 100 requirements directly constrains the software's efficiency in utilizing available CPU and memory resources on the server hardware.

- Client Devices:
    - Users will access the ARIA web application via their personal computing devices (desktops, laptops, tablets) using standard web browsers.
    - Characteristics: These devices have varying CPU, RAM, and screen resolutions.
    - Data and Control Interactions: Indirect interaction through standard HTTP/HTTPS protocols and browser rendering engines.
    - Inputs/Outputs: User input (keyboard, mouse/touch) translated into HTTP requests; server responses rendered as HTML, CSS, and JavaScript for visual output.
    - Timing Issues: User interface responsiveness and rendering speed will depend on client device capabilities and network latency. The ≥80% positive usability feedback target implies that the web interface must perform adequately across a range of typical client devices.

5.4. Communications Interfaces

- Web Browser Communication (HTTP/HTTPS):
    - Purpose: The primary means for users to access and interact with the ARIA web application.
    - Protocols: HTTP and HTTPS. HTTPS will be used for all production deployments to ensure secure communication, even for the MVP.
    - Message Formatting: Standard HTTP requests (GET, POST) and responses (HTML, CSS, JavaScript, JSON data).
    - Data Transfer Rates: Dependent on the user's internet connection. The system should be optimized for reasonable transfer rates.
    - Security/Encryption:
        - HTTPS: All data transmitted between the client browser and the ARIA server will be encrypted using TLS/SSL to protect against eavesdropping and tampering.
        - Input Validation: As mentioned in System Interfaces, all user inputs will undergo validation to prevent common web vulnerabilities (e.g., cross-site scripting, SQL injection).
    - Handshaking/Synchronization: Standard HTTP/HTTPS request-response mechanisms.
    - Constraints:
        - No specific email communication functions are required for the initial release.
        - No direct integrations with external communication platforms (e.g., Slack, Jira) are planned for the MVP.
        - The web application must function correctly across different modern web browsers (e.g., Chrome, Firefox, Safari, Edge).

6. Quality Attributes

6.1. Usability

The ARIA system must be designed to be user-friendly, efficient, and minimize user effort, particularly for its target audience of course instructors and local startups.
Ease of Use & Learning:
- QA01: The web interface shall be intuitive, allowing users to understand the core prioritization workflow (upload, process, view results) without extensive training.
- QA02: The system shall provide clear, concise labels and instructions for all input fields and actions.
- QA03: The system shall achieve ≥80% positive usability feedback from 10 student/instructor testers during evaluation.

Efficiency:
- QA04: Users shall be able to upload a requirements file and initiate prioritization with a maximum of 3 clicks/actions from the main dashboard.
- QA05: The visualization of results (sorted list, bar chart) shall be presented in a clear and organized manner to facilitate rapid comprehension.

Error Handling and Recovery:
- QA06: The system shall provide informative and actionable error messages for invalid inputs (e.g., incorrect file format, missing critical data).
- QA07: The system shall prevent data loss due to user errors through confirmation prompts for critical actions (e.g., discarding unsaved work, though not applicable in the initial stateless MVP).

Accessibility:
- QA08: The web interface shall adhere to basic web accessibility guidelines (e.g., sufficient color contrast, keyboard navigability) to ensure broad usability, though full WCAG compliance is not an MVP target.

User Interface Design Guidelines:
- QA09: The user interface shall employ a clean, minimalist design to avoid cognitive overload.
- QA10: Consistent visual elements (fonts, colors, button styles) shall be used throughout the application.

6.2. Performance

- QA11: The system shall process 100 requirements and generate prioritized output within a maximum of 5 seconds (except D1, D2 cases) from the moment the user initiates the prioritization command to the display of results.
- QA12: The web application pages shall load within 2 seconds on a standard broadband internet connection.
- QA13: File uploads of up to 1MB (typical for 100 requirements) shall complete within 3 seconds, depending on network conditions.
- QA14: Data export (CSV, HTML report) for 100 requirements shall be generated within 2 seconds.

6.3. Security

- QA15: The system shall implement input validation for all user-provided data (e.g., file uploads, form inputs) to prevent common web vulnerabilities such as SQL injection, cross-site scripting (XSS), and directory traversal attacks.
- QA16: All communication between the client browser and the server shall be encrypted using HTTPS/TLS.
- QA17: The system shall not store any personally identifiable information (PII) or sensitive project data beyond the active user session for the MVP, as there is no user authentication.
- QA18: Access to the deployed application will not require authentication in the initial release, thus implying a shared environment for the MVP demo.
- QA19: Deployment environments (Heroku free tier/university servers) shall leverage their inherent security measures where applicable.

6.4. Safety

- QA20: The system shall not allow the execution of arbitrary code uploaded by users.
- QA21: In the event of an internal processing error, the system shall provide an informative message to the user and attempt to gracefully recover or indicate failure without crashing the application.
- QA22: The system shall clearly indicate when a prioritization process has failed or yielded potentially inconclusive results, to prevent users from making critical decisions based on faulty output.
- QA23: The system shall have robust error handling around ML model failures to prevent incorrect prioritization from being presented as valid.
- QA24: The system shall not delete or modify original user-uploaded files; it will only process and generate new output based on them.

7. Internationalization and Localization Requirements

En for MVP. Other is *NOT MVP*

8. Other Requirements

Appendix A: Glossary

-

Appendix B: Analysis Models

-