Software Requirements Specification

for

“Data Visualiser”

Version 1.0 approved

Prepared by Hanna Yershova

17/02/2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

Data Visualiser project is a web-based application enabling users to **upload datasets** in various formats (CSV, JSON, Excel) and generate **interactive visualizations** to analyze trends and patterns efficiently.

This Software Requirements Specification (SRS) will serve as a **reference for developers, testers, and stakeholders**, ensuring clarity in the project's scope, objectives, and constraints. The primary goals of the system include:

- Providing an **intuitive** and **user-friendly** interface for data visualization.
- Supporting **multiple chart types** such as bar charts, line graphs, scatter plots, and pie charts.
- Ensuring **fast and secure** data processing with no permanent data storage.
- Enabling **customization** and **exporting** of visualizations for external use.

The system is intended for **students, researchers, analysts, and small businesses** who require a lightweight and accessible data visualization tool without the complexity of professional analytics software.

1.2 Document Conventions

This **Software Requirements Specification (SRS)** follows standard documentation conventions to ensure clarity, consistency, and readability. The following conventions are used throughout this document:

1.2.1 Formatting Conventions

- **Bold Text** – Used for section headings, key terms, and important concepts.
- *Italic Text* – Used for emphasis or to highlight specific terms.
- Monospace Font – Used for code snippets, file formats, and system commands.

1.2.2 Requirement Numbering

- Requirements are labeled using a structured numbering system for easy reference:
    - **FR-x.x** → Functional Requirements (e.g., FR-1.1 for File Upload)
    - **NFR-x.x** → Non-Functional Requirements (e.g., NFR-2.1 for Performance)
    - **UI-x.x** → User Interface Requirements
    - **SEC-x.x** → Security Requirements

1.2.3 Priority Levels

Each requirement is assigned a priority level to indicate its importance:

- **[High]** – Essential features that must be implemented.
- **[Medium]** – Important features that enhance functionality but are not critical.
- **[Low]** – Optional features that can be implemented in future versions.

1.2.4 Assumptions and Dependencies

- Higher-level requirements **inherit priorities** unless explicitly stated otherwise.

- Each requirement statement is **self-contained**, and dependencies on other requirements are explicitly mentioned.

1.3 Intended Audience and Reading Suggestions

1.3.1 Intended Audience

This Software Requirements Specification (SRS) is intended for the following stakeholders involved in the **Data Visualiser** project:

- **Developers** – To understand functional and non-functional requirements for system implementation.
- **Project Managers** – To track project scope, timeline, and deliverables.
- **Testers / QA Engineers** – To design test cases based on system requirements.
- **End Users** – To gain insights into the expected functionality and features of the system.
- **Technical Writers / Documentation Team** – To create user manuals and support documents.

1.3.2 Document Structure

This SRS document is organized into the following sections:

1. **Introduction** – Provides the purpose, conventions, audience, and system scope.
2. **Overall Description** – Outlines system functionality, constraints, and dependencies.
3. **Specific Requirements** – Lists functional and non-functional requirements.
4. **External Interface Requirements** – Describes UI, hardware, and software interfaces.
5. **Other Requirements** – Covers security, performance, and scalability factors.

1.3.3 Reading Recommendations

- **New team members** should start with the **Introduction** and **Overall Description** to understand project objectives.
- **Developers** should focus on **Specific Requirements** and **External Interface Requirements** for implementation details.
- **Testers** should concentrate on **Functional and Non-Functional Requirements** to design test cases.
- **Project Managers** should review the **Scope, Constraints, and Assumptions** to ensure project feasibility.

1.4 Project Scope

1.4.1 Overview

The **Data Visualiser** is a web-based application designed to help users **upload datasets** and generate **interactive visualizations** in various formats. The software aims to provide a simple yet powerful tool for students, researchers, analysts, and small businesses to analyze and interpret data without requiring advanced technical knowledge.

1.4.2 Purpose and Objectives

The primary objectives of the **Data Visualiser** project are:

- To enable users to **upload and process** data files in CSV, JSON, and Excel formats.
- To provide a variety of **data visualization options**, including bar charts, line graphs, scatter plots, and pie charts.
- To offer **interactive features** such as zooming, filtering, and data customization.
- To ensure **fast and secure** data handling without permanently storing user files.
- To maintain a **user-friendly interface** that requires minimal technical expertise.

1.4.3 Benefits and Business Value

The system provides several key benefits:

- **Enhanced Data Interpretation** – Users can quickly transform raw data into meaningful insights.
- **Accessibility** – The application is web-based and does not require installation.
- **Time Efficiency** – Automates data visualization, reducing the need for manual graph creation.
- **Cost-Effective** – Uses open-source technologies, minimizing financial investment.

1.4.4 Alignment with Business Strategies

The project aligns with modern software development strategies by:

- Utilizing **modern web technologies** (React.js, D3.js, Chart.js, etc.).
- Supporting **cloud-based deployment** for scalability and ease of access.
- Ensuring **compliance with usability and security best practices**.

1.5 References

2. Overall Description

2.1 Product Perspective

The **Data Visualiser** is a **new, self-contained web application** designed to provide an intuitive and interactive way to visualize data. It is not a follow-on product or a replacement for any existing system but rather an independent tool that simplifies data analysis for users without requiring complex software like Excel, Tableau, or Python-based solutions.

This product is part of the broader trend of **data-driven decision-making**. It aims to support **students, researchers, analysts, and small businesses** by offering a simple yet powerful data visualization tool.

2.2 Product Features

The **Data Visualiser** provides a streamlined, user-friendly approach to **data visualization** by enabling users to upload datasets and generate meaningful insights through interactive charts. Below is a high-level summary of its key features:

1. File Upload & Data Processing

- Supports **CSV, JSON, and Excel** file formats.
- Parses and validates uploaded data for correctness.
- Provides error messages for unsupported or incorrectly formatted files.

2. Data Visualization

- Offers multiple **chart types**:
  - **Bar charts, Line graphs, Pie charts, Scatter plots, Histograms, Heatmaps**
- Enables users to **switch between visualization types** dynamically.
- Supports **real-time interactivity**, such as zooming, filtering, and hovering over data points.

3. Data Customization & Transformation

- Allows **sorting, filtering, and grouping** of data before visualization.
- Provides customization options such as **color selection, axis labels, and legends**.
- Enables **chart annotations** for additional data insights.

4. Export & Sharing

- Users can download visualizations as **PNG, SVG, or PDF** formats.
- Provides **embedding options** for sharing visualizations in documents or websites.

5. User-Friendly Interface & Accessibility

- Responsive **web-based UI** that works across **desktop and mobile devices**.
- Provides **tooltips and step-by-step guidance** for new users.
- Ensures compatibility with major browsers (**Chrome, Firefox, Edge, Safari**).

6. Performance & Security

- Ensures **fast rendering** for datasets up to **10,000 rows**.
- Implements **security measures** to prevent malicious file uploads.
- No permanent storage of uploaded files, ensuring **user privacy**.

2.3 User Classes and Characteristics

1. General Users (Casual Users)

- **Description**: Users who need to visualize data occasionally without deep technical expertise.
- **Frequency of Use**: Occasional or one-time users.
- **Functions Used**:
  - File upload
  - Basic visualizations (bar charts, line graphs, scatter plots)
  - Download/export of visualizations
- **Technical Expertise**: Low to moderate; may not have experience with data visualization tools.
- **Security/Privileges**: Limited; can upload and visualize their own data but cannot manage system settings.
- **Educational Level/Experience**: Varies; could be students, professionals, or hobbyists with minimal data science knowledge.
- **Priority**: Moderate – ensuring ease of use and accessibility is important.

2. Data Analysts & Researchers

- **Description**: Users who need detailed insights from their datasets for research, reports, or business analysis.
- **Frequency of Use**: Regular users.
- **Functions Used**:
    - Advanced visualization options (heatmaps, histograms, time-series plots)
    - Data filtering and transformation
    - Comparison of multiple datasets
- **Technical Expertise**: Moderate to high; familiarity with data tools such as Excel, Tableau, or Python-based visualization libraries.
- **Security/Privileges**: Can access advanced settings, customize visualizations, and possibly share reports.
- **Educational Level/Experience**: Likely to have experience with data handling and analysis; could be university researchers or business analysts.
- **Priority**: High – primary target users; they require powerful yet intuitive features.

3. Developers & Data Scientists

- **Description**: Users integrating the tool into workflows or customizing visualizations.
- **Frequency of Use**: Regular to heavy users.
- **Functions Used**:
    - API access (if provided)
    - Custom script execution (if supported)
    - Exporting processed data
- **Technical Expertise**: High; comfortable with programming, scripting, and working with large datasets.
- **Security/Privileges**: May require advanced settings, ability to handle large datasets, and customization options.
- **Educational Level/Experience**: Typically professionals with experience in data science, machine learning, or software development.
- **Priority**: Medium – valuable but may not be the primary audience.

4. System Administrators

- **Description**: Users responsible for maintaining and managing the platform.
- **Frequency of Use**: Occasional, mostly for system setup and troubleshooting.
- **Functions Used**:
    - User management
    - Data storage management
    - Security and access control
- **Technical Expertise**: High; familiarity with web applications, databases, and security protocols.
- **Security/Privileges**: Full control over the platform, including user management and system configurations.
- **Educational Level/Experience**: IT professionals with system administration expertise.
- **Priority**: Low – only relevant for internal platform management.

Favored vs. Less Important User Classes

1. **Favored Users**:
    - Data Analysts & Researchers (High priority)

- General Users (Moderate priority)

2. **Less Important Users**:
   - Developers & Data Scientists (Secondary focus)
   - System Administrators (Only for maintenance)

2.4 Operating Environment

Operating Environment

1. Hardware Platform

- The software will primarily run on **personal computers (PCs), laptops, and mobile devices**.
- Minimum system requirements:
  - **Processor**: Intel Core i3 (or equivalent) and above
  - **RAM**: At least **4GB** (8GB recommended for handling larger datasets)
  - **Storage**: At least **200MB** of available disk space
  - **Graphics**: Integrated or dedicated GPU for rendering complex visualizations

2. Operating System & Browser Support

- The software will be a **web-based application**, making it **OS-independent**.
- Compatible with:
  - **Windows** (Windows 10, Windows 11)
  - **macOS**
  - **Linux** (Ubuntu, Fedora, etc.)
  - **Mobile OS**: Android & iOS (through mobile browsers, possibly a responsive web app)
- Supported browsers (latest stable versions):
  - **Google Chrome**
  - **Mozilla Firefox**
  - **Microsoft Edge**
  - **Safari**

3. Software Dependencies

- **Backend Framework**: Likely **Node.js**, Python (Flask/Django), or a similar backend service
- **Frontend Framework**: JavaScript-based frameworks such as **React.js, Vue.js, or Angular**
- **Database**: PostgreSQL, MySQL, or MongoDB for data storage (if required)
- **Data Processing Libraries**: Pandas, NumPy (if using Python for data handling)
- **Visualization Libraries**: D3.js, Chart.js, Plotly, or similar
- **Cloud & Storage**: Integration with **AWS S3, Google Drive, or Firebase** for file handling (if applicable)

4. Network & Connectivity

- Requires an **active internet connection** for full functionality
- May support **offline mode (limited functionality)** in the future

5. Security & Compliance

- **HTTPS encryption** for secure data transmission
- **User authentication** (OAuth, JWT, or session-based)
- Compliance with **GDPR/CCPA** for data privacy if handling user data

2.5 Design and Implementation Constraints

Design and Implementation Constraints

1. Regulatory & Corporate Policies

- Must comply with **GDPR** (General Data Protection Regulation) and **CCPA** (California Consumer Privacy Act) for handling user data.
- Data storage policies must align with **institutional or corporate user privacy and retention policies**.
- If deployed in an educational or corporate environment, it needs to comply with **ISO 27001 (Information Security Management)**.

2. Hardware & Performance Limitations

- **Memory Requirements**: Should efficiently handle datasets within **4GB RAM** constraints on lower-end devices.
- **Performance**: Must process data and generate visualizations within **2-5 seconds** for optimal user experience.
- **Mobile Responsiveness**: The UI must be optimized for touch interactions since the application should run on mobile devices.

3. Software & Technology Constraints

- **Frontend Technologies**: Must use **React.js** (or another JavaScript framework) for a modern and dynamic user interface.
- **Backend Framework**: Must be developed using **Node.js** or **Python (Flask/Django)** to ensure flexibility and scalability.
- **Database**: If persistent storage is needed, options are limited to **PostgreSQL, MySQL, or MongoDB**.
- **Data Visualization Libraries**: Should use **D3.js, Chart.js, or Plotly** for creating interactive charts and graphs.
- **Hosting & Deployment**: Must be compatible with **AWS, Google Cloud, or Firebase** for scalability.

4. Security Considerations

- **User Authentication**: Requires secure login via **OAuth 2.0, JWT, or session-based authentication**.
- **Data Encryption**: All data transmissions must be secured via **HTTPS and TLS encryption**.
- **Access Control**: Users should only have access to their datasets unless explicitly shared.

5. Interfaces to Other Applications

- Should support file uploads in standard formats: **CSV, JSON, XLSX**.
- Potential API integration with third-party data sources (e.g., Google Sheets, APIs for real-time data).

- If required, it may support exporting data in **PDF, PNG, or interactive HTML formats**.

6. Development & Maintenance Constraints

- **Programming Standards**: The code should follow **industry best practices** and be well-documented for future maintenance.
- **Version Control**: Must use **GitHub/GitLab** for collaborative development and version control.
- **Testing Requirements**: Should include **unit testing (Jest, PyTest, Mocha)** and **UI testing (Selenium, Cypress)**.