# Software Requirements Specification for WAIV

**Version 1.3**

**Last Updated:** July 2, 2025

**Table of Content**

1. Introduction 2

1.1. Purpose 2

1.2. Intended Audience 2

1.3. Intended Use 2

1.4. Scope 2

1.5. Definitions and Acronyms 2

2. Overall Description 3

2.1. Product Perspective 3

2.2. Product Functions 3

2.3. User Classes and Characteristics 3

2.4. Operating Environment 4

2.5. Design and Implementation Constraints 4

2.6. Assumptions and Dependencies 4

3. System Features 5

3.1. User Authentication 5

3.2. Staff Management 5

3.3. Student Management 5

3.4. Counseling Session Management 6

3.5. Simplicity Data Import & Viewing 6

3.6. Monthly CO-OP Listing Import & Viewing 6

3.7. Monthly Report Generation 6

4. External Interface Requirements 7

4.1. User Interfaces 7

4.2. Hardware Interfaces 7

4.3. Software Interfaces 7

4.4. Communications Interfaces 7

5. Non-functional Requirements 8

5.1. Performance Requirements 8

5.2. Security Requirements 8

5.3. Scalability 8

5.4. Portability 8

5.5. Reliability 8

5.6. Maintainability 8

5.7. Usability 9

## 1. Introduction

### 1.1. Purpose

The purpose of this document is to provide a detailed description of the requirements for the WAIV web application. It will serve as a guide for the development team and a point of reference for all stakeholders.

### 1.2. Intended Audience

This document is intended for:

* **Project Managers:** To understand the scope and plan the project.
* **Developers:** To understand the features and functionalities to be implemented.
* **QA/Testers:** To create test cases and ensure the software meets the specified requirements.
* **Stakeholders:** To have a clear understanding of the final product.

### 1.3. Intended Use

This SRS will be used to:

* Clarify the project's vision for all involved parties.
* Provide a basis for design and development.
* Serve as a foundation for testing and validation.
* Manage and control changes to the project scope.

### 1.4. Scope

The WAIV application will be an internal web-based platform designed to streamline the management and reporting processes for the WAIV program. The system will manage staff and student information, track student counseling sessions, and facilitate the generation of mandatory monthly reports for the Department of Rehabilitation (DOR). Key features include importing student activity data from external CSV files (Simplicity reports, CO-OP listings) to consolidate information and automate the creation of detailed monthly progress reports for each student.

### 1.5. Definitions and Acronyms

|  |  |
| --- | --- |
| **Term/Acronym** | **Definition** |
| **SRS** | Software Requirements Specification |
| **WAIV** | Workability IV program |
| **Django** | A high-level Python web framework. |
| **MSSQL** | Microsoft SQL Server |
| **DOR** | Department of Rehabilitation |
| **CO-OP** | Cooperative Education (referring to a student program/listing) |
| **CSV** | Comma-Separated Values |
| **UI** | User Interface |
| **DB** | Database |

## 2. Overall Description

### 2.1. Product Perspective

WAIV will be a self-contained, monolithic web application built using the Django framework. It will not be a component of a larger system.

### 2.2. Product Functions

The primary functions of the WAIV application are:

* User authentication and role-based access control.
* Staff credential and information management (add, edit).
* Student information management for WAIV program intakes (add, edit, view).
* Creation and tracking of student counseling sessions.
* Importing, storing, and viewing of monthly Simplicity reports from CSV files.
* Importing, storing, and viewing of monthly CO-OP listings from CSV files.
* Generation of standardized monthly reports for DOR counselors based on aggregated data.

### 2.3. User Classes and Characteristics

|  |  |
| --- | --- |
| **User Class** | **Description** |
| **Administrator** | A high-level user with full control over the application. Can manage all user accounts (staff, managers), student data, and has access to all system features. |
| **Case Manager / Program Coordinator** | A role with permissions similar to an Administrator. They can add and manage staff, add and manage students, and are responsible for importing system-wide data (Simplicity, CO-OP) and overseeing the report generation process. |
| **WAIV Counselor** | A user focused on student interaction. They can view and edit student information, add and view counseling sessions for their assigned students, and generate monthly reports. They cannot add new students or manage staff accounts. |
| **Anonymous User** | Any individual who is not logged in. Access to the system will be completely restricted. |

### 2.4. Operating Environment

* **Server-side:** The application will be hosted on a server environment capable of running Python/Django, with a WSGI-compliant application server (e.g., Gunicorn) and a reverse proxy (e.g., Nginx).
* **Database:** The primary production database will be **Microsoft SQL Server (MSSQL)**. The application must use a Django-compatible MSSQL database backend (e.g., mssql-django).
* **Client-side:** The application must be accessible on modern web browsers (Chrome, Firefox, Safari, Edge) on desktop and mobile devices.

### 2.5. Design and Implementation Constraints

* The application **must** be developed using the Django framework.
* The application will follow a monolithic architecture.
* All code must adhere to the PEP 8 style guide for Python.
* Sensitive data (e.g., passwords) must be securely hashed before being stored in the database.

### 2.6. Assumptions and Dependencies

* The development team has expertise in Python, Django, and connecting Django to an MSSQL database.
* The necessary hardware and hosting environment will be available.
* Monthly reports (Simplicity, CO-OP listings) will be provided in a consistent CSV format.
* Users will access the application over a secure internal network or VPN.

## 3. System Features

### 3.1. User Authentication

* **3.1.1. Description and Priority:** Users must be able to log in and log out of the application to access role-specific features. This is a high-priority feature.
* **3.1.2. Functional Requirements:**
  + The system shall allow registered users to log in with their email and password.
  + Password resets will be handled manually by an Administrator or Case Manager / Program Coordinator.
  + The system shall allow logged-in users to log out.

### 3.2. Staff Management

* **3.2.1. Description and Priority:** Administrators and Case Managers / Program Coordinators need the ability to create and manage accounts for staff members. This is a high-priority feature.
* **3.2.2. Functional Requirements:**
  + The system shall provide a form for users with appropriate permissions (Administrator, Case Manager / Program Coordinator) to add a new staff member.
  + The form shall include fields for Email, Username, Password, First Name, Last Name, and Job Title.
  + The system shall allow authorized users to view a list of all staff members.
  + The system shall allow authorized users to edit the information of an existing staff member.

### 3.3. Student Management

* **3.3.1. Description and Priority:** Administrators and Case Managers / Program Coordinators need to add new students to the program. Counselors can view and manage the information of existing students. This is a high-priority feature.
* **3.3.2. Functional Requirements:**
  + The system shall provide a form to add a new intake student, accessible only by Administrators and Case Managers / Program Coordinators.
  + The system shall display all students in a paginated, searchable, and sortable table, accessible to all authenticated roles.
  + Each row in the student table shall have an "Edit" button.
  + Clicking the "Edit" button shall allow a user with appropriate permissions (Administrator, Case Manager / Program Coordinator, WAIV Counselor) to modify the student's information.

### 3.4. Counseling Session Management

* **3.4.1. Description and Priority:** Counselors need to log the services they provide to students. This is a high-priority feature.
* **3.4.2. Functional Requirements:**
  + The system shall allow a WAIV Counselor to add a new counseling session for a specific student.
  + The session form must include the type of service provided (e.g., Employment Preparation, Internship, Job Development and Retention).
  + The system shall allow a WAIV Counselor to search for a student and view a chronological list of all previously logged counseling sessions for that student.

### 3.5. Simplicity Data Import & Viewing

* **3.5.1. Description and Priority:** Case Managers / Program Coordinators need to import monthly student activity reports from an external system ("Simplicity"). This is a high-priority feature.
* **3.5.2. Functional Requirements:**
  + The system shall provide a feature to upload a CSV file.
  + The system shall parse the CSV file and import the data into the database.
  + The system shall provide an interface to view the imported Simplicity data.
  + The viewing interface shall allow users to filter the data based on the date it was imported.

### 3.6. Monthly CO-OP Listing Import & Viewing

* **3.6.1. Description and Priority:** Case Managers / Program Coordinators need to import monthly CO-OP listings sent by the DOR. This is a high-priority feature.
* **3.6.2. Functional Requirements:**
  + The system shall provide a feature to upload a CSV file containing the CO-OP listing.
  + The system shall parse the data, which includes information on which students have been serviced, which cases are closed, and student-DOR counselor assignments.
  + The system shall provide an interface to view the imported CO-OP listing data.
  + The viewing interface shall allow users to filter the data based on the date it was imported.

### 3.7. Monthly Report Generation

* **3.7.1. Description and Priority:** Counselors and Case Managers / Program Coordinators need to generate standardized monthly reports for the DOR. This is a high-priority feature.
* **3.7.2. Functional Requirements:**
  + The system shall allow a user to select a DOR counselor from the imported CO-OP listing data.
  + The system shall display a list of all students assigned to the selected counselor.
  + Upon selecting a student, the system shall generate a standardized report form.
  + The report shall be pre-populated with the student's counseling session history for the month.
  + The report shall include a section for the user to add comments (e.g., "Describe step toward goals").
  + The report shall include fields for the user to input their name (as a signature) and the date.
  + The system shall provide an option to export the completed report form, preferably in PDF or another non-editable format.

## 4. External Interface Requirements

### 4.1. User Interfaces

The application will have a clean, modern, and responsive user interface that is intuitive and easy to navigate. The UI should be designed for internal use, prioritizing functionality and clarity over complex aesthetics.

### 4.2. Hardware Interfaces

No direct hardware interfaces are required.

### 4.3. Software Interfaces

The system will not directly interface with other software via APIs. However, it is dependent on receiving data from external sources:

* **Microsoft Outlook (or other email client):** This will be the delivery mechanism for the CSV files (Simplicity, CO-OP) that need to be imported. This is a manual process, not an automated integration.

### 4.4. Communications Interfaces

The application will communicate with users' web browsers via the HTTP/S protocol. All communication must be encrypted using TLS/SSL.

## 5. Non-functional Requirements

### 5.1. Performance Requirements

* The application should load key pages in under 3 seconds on a standard internal network connection.
* Data import processes for large CSV files should be handled asynchronously to avoid blocking the user interface and should complete within a reasonable time frame (e.g., under 2 minutes for a typical monthly file).
* The server should be able to handle at least 100 concurrent users without significant degradation in performance.

### 5.2. Security Requirements

* The application must be protected against common web vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). Django's built-in security features must be utilized and correctly configured.
* User passwords must be securely hashed using a strong algorithm (e.g., Argon2, bcrypt).
* Role-based access control must be strictly enforced to ensure users can only access data and features appropriate for their role.
* The administrative interface should be accessible only to authorized personnel.

### 5.3. Scalability

The application should be designed in a way that allows for future scaling. While it is a monolith, the code should be modular to facilitate potential future transitions to a microservices architecture if needed. Database queries should be optimized for performance as the volume of data grows.

### 5.4. Portability

The application should be easily deployable on different environments (development, staging, production) with minimal configuration changes, managed via environment variables.

### 5.5. Reliability

The application should have an uptime of at least 99.5%. Backups of the MSSQL database should be performed regularly according to workplace policy to prevent data loss.

### 5.6. Maintainability

The code should be well-documented, with clear comments and a logical structure to facilitate future maintenance and updates.

### 5.7. Usability

The application should be easy to learn and use for all user classes. Navigation should be intuitive, and error messages should be clear and helpful, especially for data import failures (e.g., "Row 15: Invalid date format").