SOFTWARE REQUIREMENTS SPECIFICATION

AURA: Automated User Revenue Assistant

Date: 30th January 2026

1. Introduction

1.1 Purpose

This document defines the software requirements for the **AURA (Automated User Revenue Assistant)**. The purpose of this SRS is to clearly describe system functionality, behavior, constraints, and acceptance conditions to ensure development and validation are performed without ambiguity. It serves as the formal reference for implementation; any feature not explicitly described herein is considered out of scope.

1.2 Scope

AURA is a web-based financial monitoring system designed for freelancers and independent professionals.

- **Core Functions:** Records contracts, milestones, delivery dates, invoice eligibility, and payment timelines.
- **Logic:** Evaluates stored data against the current date to determine payment status and provide contract level financial visibility.
- **Exclusions:** The system does not process payments, compute taxes, integrate with accounting platforms, or perform legal enforcement.

1.3 Intended Recipients and Readers

This document is intended for:

- **Project Evaluators & Faculty Reviewers:** For academic validation.
- **Developers:** For system implementation.
- **Testers:** For validating system behavior against requirements.

1.4 Abbreviations

Abbreviation	Full Form
SRS	Software Requirements Specification
FR	Functional Requirement
NFR	Non Functional Requirement
TLS	Transport Layer Security
ERP	Enterprise Resource Planning

---

2. Functional Requirements

The following functional requirements define the core operational behavior of AURA:

- **FR 01:** The system shall allow the user to create a contract record containing client name, contract start date, total contract value, and payment term (in days).
- **FR 02:** The system shall allow the user to create one or more milestones under a contract, including milestone name, planned delivery date, and payment amount.
- **FR 03:** The system shall allow the user to record the actual delivery date of a milestone.
- **FR 04:** Upon recording the actual delivery date, the system shall mark the milestone as **invoice eligible**.
- **FR 05:** The system shall calculate the payment due date by adding the contract payment term to the recorded delivery date.
- **FR 06:** The system shall allow the user to record the payment received date for each milestone.
- **FR 07:** If the current date exceeds the calculated due date and no payment date is recorded, the system shall automatically mark the milestone as **overdue**.
- **FR 08:** The system shall display financial summaries per contract: total received, total pending, and total overdue amounts.

---

3. Non Functional Requirements

These requirements define quality attributes and operational constraints:

- **NFR 01 (Accessibility):** Accessible via modern web browsers (Chrome, Firefox, Edge).
- **NFR 02 (Authentication):** Requires username/password authentication for access.
- **NFR 03 (Security):** User passwords shall be stored using **SHA-256 hashing**.
- **NFR 04 (Encryption):** All communication shall use HTTPS with **TLS 1.2** or higher.
- **NFR 05 (Environment):** The system shall be deployed on a **Linux-based** server.

- **NFR 06 (Performance):** Dashboard/summary pages shall load within **3 seconds** for up to 100 contracts.
- **NFR 07 (Concurrency):** The system shall operate as a single-user application in this version.

---

4. System Requirements

- **Backend:** Implemented using **Python**.
- **Framework:** Use Flask web application framework compatible with Python.
- **Database:** Data shall be stored in a **Relational Database Management System (RDBMS)**.
- **Deployment:** Must be deployable on a Linux server environment.
- **Data Integrity:** Support storage and retrieval of all records without data loss.

---

5. Interface Requirements

5.1 User Interface

- The system shall provide structured form based input screens for contracts and milestones.
- A dashboard shall display project status and financial summaries in a **tabular format**.
- Overdue payments must be visually highlighted (e.g., color coded) on the dashboard.

5.2 Software Interface

- **Database:** Interaction with an RDBMS for persistent storage.
- **Components:** Integration with a PDF generation component for payment notices.

5.3 Communication Interface

- Client-server communication shall use **HTTP/HTTPS** protocols over **TCP/IP**.

---

6. Use Case

Use Case Name: Record Delivery and Track Payment Status

Actor: Freelancer

Precondition: User is authenticated and logged into the system.

Main Flow:

1. The user selects an existing contract.
2. The user selects a specific milestone.
3. The user records the **actual delivery date**.

4. The system marks the milestone as **invoice eligible**.
5. The system calculates the **payment due date**.
6. The system updates financial summaries and status.

Postcondition: The contract reflects updated delivery and payment status.

---

7. Assumptions

- Users will enter accurate and truthful contract and milestone data.
- The system is used by one primary account holder in the current version.
- Stable internet connectivity is available during system use.

---

8. Constraints

- No integration with external payment gateways (e.g., Stripe, PayPal).
- No automatic parsing of uploaded contract documents.
- No multi user collaboration or role based access control (RBAC).
- No accounting ledger entries or double entry bookkeeping.

---

9. Acceptance Criteria

- All Functional Requirements (Section 2) are implemented and verifiable.
- Due dates are calculated correctly based on logic defined in FR 05.
- Overdue status triggers accurately based on the system clock.
- Financial summaries (Total/Pending/Overdue) match stored milestone data.
- The application operates without critical defects in the core workflow.

---

10. Appendix

- **Future Roadmap:** Potential for multi-user access control.
- **Future Roadmap:** Integration with ERP or accounting software (e.g., Tally, QuickBooks).
