Software Requirements Specification (SRS)

Project Name: CS Library Nuc Project

Team Members: - Amilcar Armmand - Project Lead, Backend, Database, Testing - armmanda1@southernct.edu - Jose Gaspar Marin - Frontend, UI/UX, Testing - gasparmarij1@southernct.edu - Kenny Molina - Frontend, Database, Testing - molinak4@southernct.edu

Document Version: Draft v1.1 **Last Updated:** 2/12/26

---

1. Executive Summary

1.1 Project Overview

The CS Library Kiosk System is an automated, self-service checkout station designed for the Computer Science Department student lounge library. It replaces the current manual, paper-based system with a touchscreen kiosk that allows students to borrow and return books using barcode scanning and student ID card authentication.

1.2 Problem Statement

Computer Science students currently use a manual library system that uses paper checkout slips, provides no real-time inventory tracking, and offers no automated due date management. This leads to lost books, time wasted searching for unavailable resources, and having to manually track checkouts and returns.

1.3 Solution Overview

Our solution is a standalone Raspberry Pi-based kiosk with integrated barcode and RFID card scanners that automates the entire library management process. The system provides instant book checkout/return, real-time inventory status, automatic ISBN lookup for cataloging, and a searchable digital catalog accessible through an intuitive touchscreen interface.

1.4 Client Context

Client Organization: Computer Science Department, Southern Connecticut State University Client Contact: IT Coordinator, Computer Science Department Faculty Advisor Client Mission/Business: Providing resources and study space for computer science students Project Sponsor: Faculty Advisor

The Computer Science Department needs an automated library system that reduces administrative burden, improves resource tracking, and provides better service to students. Success means having a reliable, easy-to-use system

that operates independently in the student lounge with minimal maintenance requirements.

2. User Research & Personas

2.1 Research Methods

Research conducted:

- Observation or shadowing of current CS Lounge library usage
- Discussions with students using the CS Lounge
- Review the existing manual checkout workflow

Key Findings:

1. The CS Library shelf has a limited selection of books. There is room to add more books. We can add more books through donations which we plan to implement a way for students to give away CS books that they don't need anymore.
2. The only way to check in and out of books is to fill out a paper with the book name, the check in date and check out date.
3. There is more than enough room in the space to set up the device. There is also space to connect the NUC via Ethernet.

2.2 User Personas

Primary User Persona

Name & Role: Taylor, Computer Science Junior Demographics: 21, Tech-savvy, comfortable with self-service systems, heavy library user Organization Context: Full-time student, uses library resources for projects and exam preparation

Needs:

- Quick access to programming reference books[Another need]
- Ability to check availability without searching shelves
- Simple checkout/return process

Pain Points:

- Can't tell if book is available without physically searching
- Manual checkout slips easily lost

Goals:

- Efficiently borrow and return books independently
- Access up-to-date catalog

Secondary User Persona Name & Role: Professor James, CS Faculty Administrator Demographics: 43, Faculty Member, strong technical proficiency

Needs: - The ability to monitor inventory and checkout activity - A easy way to add and remove books from the system

Pain Points: - Manual tracking of books means theres the chance of missing items - Can't see who has borrowed books - The overhead is time-consuming

Goals: - Reduce overhead an manual work of managing said library - Ensure books are accounted for - Improve the accessibility of books for students

Third User Persona

Name & Role: Kevin Rivera, Student Book Donor

Demographics: 22, CS Student or Student Alumni

Organization Context: Wants to donate technical books to help current students, interacting with kiosk to register books

Needs: - Simple and guided way to donate books to CS Lounge using kiosk - Scan ISBN and have details filled in - Confirmation that books were donated/checked in to library database

Pain Points: - No tracking of book donations

Goals: - Donate books quick and efficiently - Confirmation of books being donated in the system

---

3. User Stories

Core User Stories (Must Have)

Authentication & User Management **US001:** As a student, I want to authenticate by scanning my student ID card so that I can use the system without remembering credentials.

Acceptance Criteria: - [ ] Scanner reader successfully detects student ID card - [ ] System validates student is authorized to use library - [ ] User profile is created automatically - [ ] User is redirected to main dashboard after login

---

US002: As an administrator, I want to log in with username and password so that I can manage the system settings and data.

Acceptance Criteria: - [ ] Secure admin login screen accessible from main menu - [ ] Failed login attempts limited to 5 before lockout - [ ] Role-based access to different admin functions

---

Book Management **US003:** As a student, I want to check out a book by scanning its ISBN barcode so that I can borrow it quickly without paperwork.

Acceptance Criteria: - [ ] Barcode scanner reads ISBN successfully on first try 95% of time - [ ] System displays book details (title, author, cover image) - [

] Due date automatically set to 14 days from checkout - [ ] Transaction recorded with timestamp and user ID - [ ] Clear confirmation message with return date

---

US004: As a student, I want to return a book by scanning its barcode so that I can complete returns in seconds.

Acceptance Criteria: - [ ] System recognizes book as currently checked out - [ ] Return transaction recorded with timestamp - [ ] Book status updated to "available" - [ ] Confirmation message displayed

---

US005: As a student, I want to search the library catalog so that I can find books without browsing shelves.

Acceptance Criteria: - [ ] Search by title, author, or keyword - [ ] Real-time search results as user types - [ ] Results show availability status clearly - [ ] Detailed book view accessible from results

User Account Features **US006:** As a student, I want to view my borrowed books and due dates so that I can manage my returns.

Acceptance Criteria: - [ ] Display list of currently checked out books - [ ] Show due dates with overdue items highlighted - [ ] Option to renew books (if no holds) - [ ] Show borrowing history (last 6 months) - [ ] Total books checked out counter

Administration Features **US007:** As an administrator, I want to add new books to the system so that the catalog stays current.

Acceptance Criteria: - [ ] Manual entry form for books without ISBN - [ ] ISBN scan auto-populates metadata from API - [ ] Option to categorize books (programming, theory, etc.) - [ ] Upload cover images manually if needed - [ ] Confirmation message after successful addition

US008: As an administrator, I want to view all checked-out books so that I can track library usage.

Acceptance Criteria: - [ ] Dashboard showing all active checkouts - [ ] Filter by due date (today, overdue, upcoming) - [ ] Sort by borrower name or book title - [ ] Export list to CSV for record keeping - [ ] Overdue items highlighted with days overdue

US009: As an administrator, I want to manage user accounts so that I can manage library access.

Acceptance Criteria:

- ☐ Set borrowing limits per user
- ☐ View borrowing history per student

☐ Reset system for new semester

US010: As an administrator, I want to generate usage reports so that I can make data-driven decisions.

Acceptance Criteria: - [ ] Monthly checkout statistics - [ ] Most popular books report - [ ] Usage patterns by time of day/day of week - [ ] Export reports to CSV format - [ ] Print summary reports

US011: As a Book Donor, I want to be able to donate a book using the kiosk by scanning it's ISBN, which should add the book to the CS Lounge Library Database.

Acceptance Criteria: - [ ] Donor scans book ISBN at kiosk - [ ] System fills in book metadata from ISBN - [ ] Donor can also manually enter details if ISBN is invalid or missing - [ ] Confirmation from system that book has been added to the database successfully - [ ] Donation is logged with timestamp

---

Should Have (Post-MVP Enhancements)

US012: As a student, I want to reserve books that are currently checked out so that I can get them when returned.

Acceptance Criteria: - [ ] Track books that have been checked out - [ ] View currently checked out books marked in the system - [ ] Schedule a pickup date, if pre-existing days are scheduled from other users - [ ] Notify the student when the book has been returned, based on scheduled time, for pickup

US013: As a student, I want to receive email reminders about due dates so that I don't forget to return books.

Acceptance Criteria:

- ☐ Set the system to have a pre-due reminder to notify students that their book is almost due
- ☐ System emails them based on the pre-due reminder period set
- ☐ If the student does not return the book in time, set the system to create an overdue email to the user. Establish a set fee if overdue.

Could Have (Future Considerations)

US014: As a department chair, I want to access library statistics from a web dashboard so that I can monitor usage remotely.

Acceptance Criteria: - [ ] System must track every action created by the end user (book check ins, book check outs, etc.) - [ ] Dashboard is created to view statistics - [ ] Department chair is given admin credentials and access

US015: As a student, I want to suggest books for purchase so that the collection meets student needs.

Acceptance Criteria: - [ ] User fills out a form on the page detailing what books can be purchased - [ ] System collects every form filled out - [ ] Administrators view the form and review it for further consideration

---

4. Features & Requirements

4.1 Core Features

- Touchscreen Kiosk Interface - Responsive, intuitive interface designed for quick transactions with large touch targets and clear visual feedback.
- Student ID Authentication - Barcode scanning of student ID cards for instant student identification without manual input.
- Barcode Book Scanning - Quick ISBN scanning for checkout/return with automatic metadata retrieval, using the same scanner as student authentication.
- Real-time Inventory Management - Instant status updates showing book availability, checkout history, and due dates.
- ISBN API Integration - Automatic population of book details (title, author, cover) from online databases.
- Admin Management Dashboard - Comprehensive backend for managing books, users, and generating reports.
- Offline Operation Mode - Core functionality continues during network outages with sync when reconnected.
- Automated Due Date Tracking - Automatic calculation of return dates with overdue detection and reporting.

4.2 Technical Requirements

Authentication & Authorization

- Barcode scanner authentication for students via student ID cards
- Username/password authentication for administrators
- Session management with automatic timeout
- Role-based access control (student vs. admin vs. super-admin)

Data Management (CRUD Operations)

- Create: Add new books, register new users, record transactions
- Read: Search catalog, view borrowing history, generate reports
- Update: Checkout/return status, user information, book details
- Delete: Archive old records, remove lost books (soft delete)

Database & Storage

- Primary Database: SQLite (lightweight, suitable for Raspberry Pi)
- Key Data Entities:
  - Users: student_id, name, email, department, is_active, date_joined
  - Books: isbn, title, author, publisher, year, category, status, date_added
  - Transactions: transaction_id, user_id, book_id, checkout_date, due_date, return_date, is_overdue
  - SystemLogs: log_id, action, timestamp, user_id, details

4.3 Non-Functional Requirements

Performance

- Checkout/return transaction completes within 3 seconds
- Search results display within 2 seconds
- Support for 5+ concurrent users during peak hours
- System boots to ready state within 60 seconds from power on

Security

- Student ID numbers stored encrypted at rest
- No sensitive personal information stored in system
- Input validation on all user inputs
- Protection against SQL injection and XSS attacks
- Physical security measures for Raspberry Pi unit

Usability

- Touchscreen-optimized interface with 44px minimum touch targets
- High contrast display readable in various lighting conditions

- Clear audio feedback for successful scans
- Intuitive icon-based navigation
- Minimal training required for new users
- Single barcode scanner for both student IDs and books to simplify user interaction

Reliability

- 99% uptime during library operating hours (8 AM - 10 PM)
- Daily automated database backups
- Graceful recovery from power interruptions
- Error logging with timestamp and user context

Accessibility

- Screen reader compatibility for visually impaired users
- Keyboard navigation support for motor-impaired users
- High contrast mode option
- Adjustable text size (within limits of fixed display)

Hardware Requirements

- Single USB barcode scanner compatible with both ISBN barcodes and student ID barcodes
- Raspberry Pi 4 (4GB RAM minimum)
- 7-inch touchscreen display
- Optional receipt printer for transaction records
- Uninterruptible power supply (UPS) for graceful shutdown

---

5. System Design

5.1 Technology Stack

Frontend:

- Terminal-based UI: ncurses (python-ncurses library)
- Python Frameworks: urwid (alternative to ncurses for more complex UIs)
- Graphical Options (if needed): PyGame, Tkinter, or Kivy for simple GUIs

- Touchscreen Support: evdev for touch input handling
- Display Management: fbcp (Framebuffer copy) for secondary display support

Backend:

- Runtime: Python 3.11+ (optimized for Raspberry Pi 5)
- Framework: Custom service architecture with asyncio
- Authentication: Custom barcode authentication system
- Validation: Pydantic for data validation
- Web Framework (Optional): FastAPI for admin web interface

Database:

- Primary Database: MySQL (MariaDB) with aiomysql for async
- Alternative: PostgreSQL with asyncpg for larger deployments
- Cache Layer: Redis (optional for performance)

Hardware Integration:

- Raspberry Pi GPIO: RPi.GPIO or gpiozero library
- USB Device Management: pyusb / libusb for scanner control
- Barcode Scanning: python-barcode / pyzbar for barcode decoding
- Touchscreen: evdev for touch input
- Power Management: python-raspi for system monitoring

Deployment:

- Platform: Raspberry Pi 5 (on-premise)
- OS: Raspberry Pi OS (64-bit) Lite
- Process Management: Systemd for daemon management
- Service Orchestration: Supervisor or systemd services
- Logging: journald (systemd journal) with Python logging integration
- Backup: rsync + cron for automated backups

Development Tools:

- Version Control: Git + GitHub
- Package Management: pip + uv (faster Python package installer)
- Environment Management: pipenv or poetry
- Testing: pytest + pytest-asyncio

- IDE: VS Code with SSH remote development
- Debugging: pdb++, remote debugging with VS Code
- Documentation: Sphinx for API documentation

Third-Party Services:

- ISBN API: Open Library API via aiohttp
- Fallback API: Google Books API
- No external authentication (self-contained system)

5.2 User Interface Design

Design Principles:

- Terminal-First Design: Optimized for 7-inch touchscreen with terminal interface
- Keyboard/Touch Navigation: Support for both keyboard shortcuts and touch taps
- High Contrast: Black/white or custom color schemes for readability
- Minimal Information: Display only essential information at each step
- Progressive Disclosure: Advanced options hidden until needed

Key Screens:

1. Welcome/Login Screen

Purpose: Initial screen for user authentication

Key Elements:

- System header with library name
- "Scan Student ID" prompt with blinking cursor
- Admin login shortcut (Ctrl+A)
- System status indicators (network, time)

User Actions:

- Scan student ID barcode
- Press Ctrl+A for admin login
- Tap screen to focus scanner input

2. Main Menu Screen

Purpose: Central navigation hub

Key Elements:

- User greeting line
- Numbered menu options (1. Checkout, 2. Return, 3. Search, 4. My Books)
- Status bar with current time and logout option
- Quick stats (books checked out)

User Actions:

- Type menu number or tap corresponding area
- Press 'L' to logout
- Press 'Q' to quit to terminal

3. Checkout Screen

Purpose: Complete book checkout transaction

Key Elements:

- Current step indicator (Step 1/2: Scan Book)
- Book details panel (appears after scan)
- Confirmation prompt with (Y/N)
- Transaction summary

User Actions:

- Scan book ISBN barcode
- Press Y to confirm, N to cancel
- Press ESC to return to main menu

4. Return Screen

Purpose: Complete book return transaction

Key Elements:

- Scan prompt with animation
- Return confirmation display
- Receipt option (print or display)
- Success/failure message

User Actions:

- Scan book barcode
- Confirm return
- Choose receipt option

5. Search Screen

Purpose: Find books in library catalog

Key Elements:

- Search input field
- Filter toggles (Title/Author/Category)
- Results list with scrollable window
- Book details pane (on selection)

User Actions:

- Type search query
- Navigate results with arrow keys or touch
- Select book for details
- Press B to check out selected book

6. My Books Screen

Purpose: View personal borrowing status

Key Elements:

- Current checkouts list with due dates
- Overdue indicator (!)
- Renew option (R) for eligible books
- Borrowing history (last 10 transactions)

User Actions:

- View current checkouts
- Press R to renew eligible books
- Scroll through history

7. Admin Dashboard

Purpose: System management (accessed via Ctrl+A)

Key Elements:

- Admin menu (1. Manage Books, 2. Manage Users, 3. Reports, 4. System)
- Quick stats (total books, active users, today's checkouts)
- System status indicators
- Activity log window

User Actions:

- Select admin function
- View system logs
- Generate reports
- Manage system settings

Terminal UI Mockup:

```
  CS LIBRARY KIOSK v1.0
=====================================

Please scan your student ID

[Scanner ready...]

Press Ctrl+A for Admin Login

Network:     Time: 14:30   Date: 3/12
```

5.3 Database Schema

```
User          Transaction          Book




Logs              Reports              Category
```

Entity: User

```
{
  "user_id": "String (unique, required, primary key)",
  "student_id": "String (unique, required)",
  "first_name": "String (required)",
  "last_name": "String (required)",
  "email": "String (unique, required)",
  "department": "String (optional)",
  "user_type": "String (enum: ['student', 'admin', 'super_admin'], default: 'student')",
  "borrowing_limit": "Integer (default: 5)",
  "books_checked_out": "Integer (default: 0)",
  "is_active": "Boolean (default: true)",
```

```
  "date_joined": "Date (required)",
  "last_login": "Date (optional)",
  "total_checkouts": "Integer (default: 0)"
}
```

Entity: Book

```
{
  "book_id": "String (unique, required, primary key)",
  "isbn": "String (unique, required)", // 10 or 13 digit ISBN
  "title": "String (required)",
  "author": "String (required)",
  "publisher": "String (optional)",
  "publication_year": "Integer (optional)",
  "category_id": "Integer (ref: Category, optional)",
  "book_status": "String (enum: ['available', 'checked_out', 'lost', 'damaged', 'donated'],
  "location": "String (optional)", // Shelf location
  "cover_image_url": "String (optional)",
  "description": "Text (optional)",
  "donor_student_id": "String (ref: User, optional)", // If donated by student
  "date_added": "Date (required)",
  "total_checkouts": "Integer (default: 0)",
  "last_checkout_date": "Date (optional)"
}
```

Entity: Transaction

```
{
  "transaction_id": "String (unique, required, primary key)",
  "user_id": "String (ref: User, required)",
  "book_id": "String (ref: Book, required)",
  "transaction_type": "String (enum: ['checkout', 'return', 'renewal', 'donation'], required
  "checkout_date": "Date (required for checkout)",
  "due_date": "Date (required for checkout)",
  "return_date": "Date (optional)",
  "actual_return_date": "Date (optional)", // For tracking lateness
  "renewal_count": "Integer (default: 0)",
  "is_overdue": "Boolean (default: false)",
  "overdue_days": "Integer (default: 0)",
  "late_fee": "Decimal(5,2) (default: 0.00)",
  "fee_paid": "Boolean (default: false)",
  "notes": "Text (optional)",
  "created_at": "Date (required)"
}
```

Entity: Category

```
{
  "category_id": "Integer (unique, required, primary key)",
```

```
  "category_name": "String (required, unique)",
  "description": "String (optional)",
  "book_count": "Integer (default: 0)",
  "created_at": "Date (required)"
}
```

Entity: SystemLog

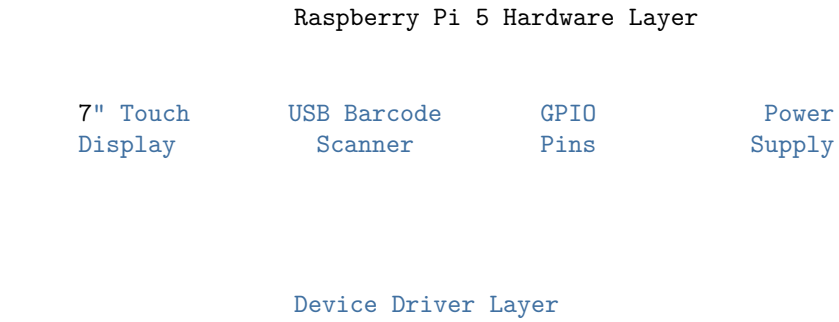
```
{
  "log_id": "String (unique, required, primary key)",
  "user_id": "String (ref: User, optional)", // Null for system events
  "action": "String (required)", // e.g., 'login', 'checkout', 'system_start'
  "log_level": "String (enum: ['info', 'warning', 'error', 'critical'])",
  "details": "Text (optional)",
  "ip_address": "String (optional)",
  "user_agent": "String (optional)",
  "timestamp": "Date (required)"
}
```

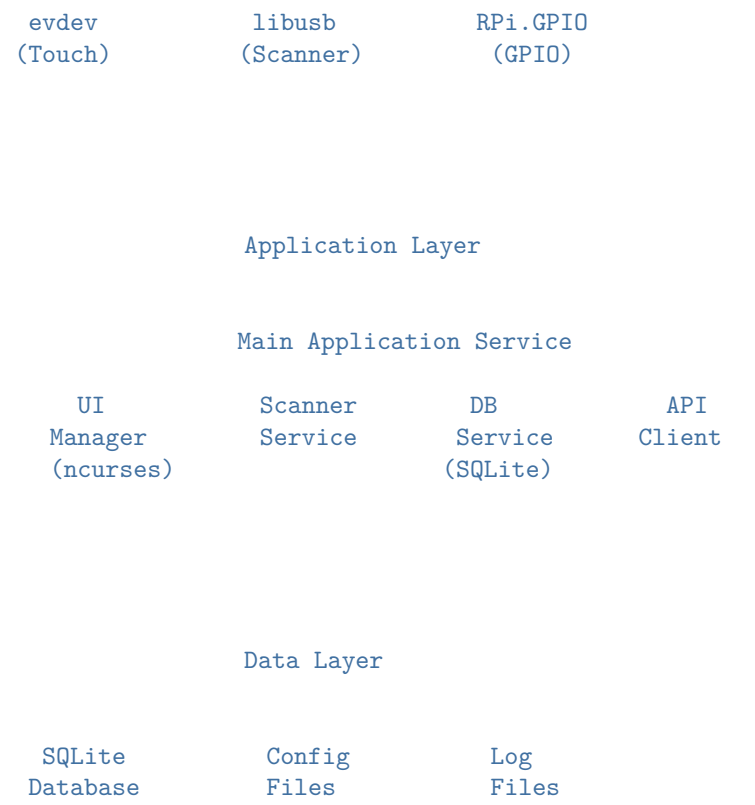
Entity: Reservation

```
{
  "reservation_id": "String (unique, required, primary key)",
  "user_id": "String (ref: User, required)",
  "book_id": "String (ref: Book, required)",
  "reservation_date": "Date (required)",
  "status": "String (enum: ['pending', 'ready', 'cancelled'], default: 'pending')",
  "notified": "Boolean (default: false)",
  "pickup_by_date": "Date (optional)",
  "created_at": "Date (required)"
}
```

5.4 System Architecture

Component Diagram:





Request/Response Flow:

Hardware Event: Student scans barcode or touches screen

Driver Processing: evdev (touch) or libusb (scanner) captures input

Event Dispatch: Main application service receives hardware event

Business Logic Processing:

- Scanner Service decodes barcode
- DB Service queries database
- FastAPI Service fetches external data (if needed)
- Transaction Service processes checkout/return logic

```
UI Update: UI Manager updates terminal display via ncurses

Data Persistence: DB Service commits transaction to SQLite

Logging: System logs event to both database and file system

Response: Visual feedback shown on terminal display
```

Security Layer:

```
Authentication:

    Student: Barcode validation against database

    Admin: Password authentication with bcrypt hashing

    Session timeout after 15 minutes of inactivity

    Failed attempt logging and lockout (5 attempts)

Authorization:

    Role-based permissions (student vs. admin)

    Function-level access control

    Command authorization checks

Input Validation:

    Barcode format validation (ISBN, student ID patterns)

    SQL injection prevention via parameterized queries

    Input sanitization for all user inputs

    Boundary checking for all numeric inputs

Data Protection:

    MySQL database encryption

    Configuration file encryption

    Secure logging (no sensitive data in logs)

    Automated encrypted backups
```

```
File system permissions (restricted access to data files)
```

---

6. Implementation Plan

6.1 Sprint Breakdown (4 Sprints)

Sprint 1: Foundation & Authentication Timeline: Week 3-6 **Goal:** Hardware setup and User Authentication

Tasks: - Amilcar: Set up Raspberry Pi OS and configure database with WAL mode - Jose/Kenny: Frontend either NiceGui or urwid - Jose/Kenny: Write script for barcode scanner evdev

Deliverables: - Working authentication system - User dashboard (basic version)

Sprint 2: Core Feature Development Timeline: Week 7-9 **Goal:** Book Checkout, Return, Inventory

Tasks: - Amilcar: Implement the Library API for metadata - Jose/Kenny: Checkout UI - Jose/Kenny: Populate database and add in logic for scanner

Deliverables: - Functional Checkout/Return flow —

Sprint 3: Feature Completion & Enhancement Timeline: Week 10-12 **Goal:** [Complete remaining features and polish]

User Stories: - US006-US008: [Remaining features] - UI/UX improvements - Error handling

Deliverables: - All core features complete - Responsive design implementation - Improved user experience

---

Sprint 4: Testing, Polish & Deployment Timeline: Week 13-14 **Goal:** Production-ready application

Tasks: - User acceptance testing - Bug fixes and refinements - Performance optimization - Final deployment and documentation

Deliverables: - Fully tested application - Production deployment - User documentation - Presentation materials

---

=======

7. Risk Assessment

Technical Risks

Risk 1: SD Card Corruption - Impact:** High - **Likelihood:** Medium - **Mitigation:** Implementation of a WAL mode and automated USB backups

Risk 2: API Rate Limiting - Impact:** Low - **Likelihood:** Low - **Mitigation:** Possibly caching book metadata locally in SQL and implement a manual entry fallback.

8. Success Metrics

- System uptime of 99% during testing phase specifically in school hours
- Book checkout transaction time under 20 seconds
- Have all available books in CS Library cataloged into the system

---

9. Appendix

A. Glossary

- **ISBN:** International Standard Book Number - A unique numeric commercial book identifier
- **WAL:** Write-Ahead Logging - SQLite Feature
- **GPIO:** General Purpose Input/Output - Pins on the Raspberry Pi that can be programmed

B. References

Technical References - Raspberry Pi Foundation. (2024). Raspberry Pi Documentation. https://www.raspberrypi.com/documentation/ - Open Library API Documentation. https://openlibrary.org/developers/api - Google Books API Documentation. https://developers.google.com/books

C. Change Log

Date	Version	Changes	Author
[1/28]	v1.0	Initial draft	[Team]
[2/6]	v1.1	Revision	[Jose]
[2/12]	v1.2	Revision	[Kenny]

Document Status: Draft / Review / Final **Next Review Date:** TBA

Prepared by: CS Library Nuc Project Course: CSC400 - Computer Science Project Seminar Semester: Spring 2026