Capstone in Software Engineering: Alpha Checkpoint

SENG 701 – Fall 2025

Project Title: Software Bill of Materials for Applications and Apps Management System

Under the guidance of

Prof. Mohammad Samarah & **Prof. Melissa Sahl**

Satwik Alla

MPS Software Engineering

University of Maryland Baltimore County

November 02, 2025.

Table of Contents

Project Data

Student Information

1) Student Name: Satwik Alla
2) Contact Phone Number: 410-905-4899
3) UMBC Email Address: satwika1@umbc.edu
4) Semester and Year of Capstone experience: Fall 2025
5) Expected graduation date: 23 December 2025

Capstone Course Information

1) Capstone faculty: Prof. Mohammad Samarah & Prof. Melissa Sahl
2) Capstone Advisor: Prof. Mohammad Samarah & Prof. Melissa Sahl

Sponsor Client Information

1) Client organization name: Ethical Software Lab
2) Client contact name: Dr. Mohammad Samarah
3) Client contact title: Director
4) Client contact number: 410-455-8038
5) Client contact email address: msamarah@umbc.edu

Abstract

This report presents the Alpha checkpoint deliverable for the SBOM Manager project, a comprehensive cloud-based web application designed to automatically generate, manage, and analyze Software Bills of Materials (SBOMs) for various software applications across multiple platforms. The project addresses the critical need for software supply chain transparency and security by providing an intuitive interface for SBOM generation, comparison, and analysis.

As of the Alpha checkpoint, the application has achieved approximately 60% of planned functionality, with core features including user authentication, file upload capabilities, SBOM generation using Syft, application management, component tracking, and a responsive user interface successfully implemented and deployed. The system architecture employs modern cloud-native technologies with a React frontend hosted on Vercel, a FastAPI backend deployed on Render.com, and a PostgreSQL database managed through Supabase.

The Alpha version supports five major platforms (iOS, Android, macOS, Windows, and Linux), implements secure authentication with JWT tokens, and provides real-time SBOM generation through background processing. This report details the project background, design methodology, implementation progress, known limitations, and planned improvements for the Beta checkpoint.

Introduction

Purpose of the Project

The SBOM Manager project was conceived to address the growing need for software supply chain transparency and security in modern software development. Software Bills of Materials (SBOMs) have become increasingly critical following Executive Order 14028 on Improving the Nation's Cybersecurity, which mandates SBOM requirements for software used by federal agencies. This project aims to democratize access to SBOM generation and analysis tools by creating a user-friendly, cloud-based platform that serves both individual developers and organizations.

The primary objectives of this project include:

- Automating SBOM generation for multiple software platforms using industry-standard tools
- Providing comprehensive search and filtering capabilities for managing multiple applications
- Enabling side-by-side comparison of SBOMs to identify differences and commonalities
- Delivering analytics and insights through a statistics dashboard
- Supporting standard SBOM export formats (SPDX and CycloneDX) for interoperability

Problem Statement

Organizations and developers face significant challenges in maintaining visibility into their software supply chains. Current SBOM generation tools often require extensive technical expertise, command-line proficiency, and platform-specific knowledge. Moreover, these tools typically operate in isolation, lacking integrated features for storing, comparing, and analyzing SBOMs across multiple applications.

The absence of a centralized, user-friendly platform for SBOM management creates several critical problems:

- Manual SBOM generation is time-consuming and error-prone, particularly when dealing with multiple platforms and technologies
- Lack of centralized storage makes it difficult to track SBOMs across different applications and versions
- Comparing SBOMs between applications or versions requires manual analysis and custom scripting
- Organizations struggle to gain insights from their SBOMs without specialized analytics tools
- Compliance with regulatory requirements demands significant resources and expertise

The SBOM Manager addresses these challenges by providing an integrated, cloud-based solution that abstracts the complexity of SBOM generation while offering powerful management and analysis capabilities accessible through an intuitive web interface.

Background of Sponsor Organization

This project is undertaken as an academic capstone project at the University of Maryland, Baltimore County (UMBC) for the Master of Professional Studies in Software Engineering program. The project aligns with UMBC's mission to foster innovation in software engineering education and prepare students for real-world challenges in cybersecurity and software supply chain management.

The project addresses a genuine industry need identified through research of current SBOM practices and gaps in existing tooling. While developed in an academic context, the SBOM Manager is designed to meet professional standards and could potentially be deployed for real-world use, serving as both a learning vehicle and a practical contribution to the software security community.

Background of Problem

The concept of Software Bills of Materials has gained significant traction following several high-profile supply chain attacks, including the SolarWinds breach in 2020 and the Log4Shell vulnerability in 2021. These incidents demonstrated how vulnerabilities in third-party components can have cascading effects across countless applications and organizations. In response, governments and industry bodies have increasingly mandated SBOM adoption as a foundational element of software security.

Current SBOM tooling landscape includes various open-source and commercial solutions, such as Syft by Anchore, SPDX tools, and CycloneDX generators. However, these tools primarily focus on generation rather than comprehensive lifecycle management. Research indicates that developers and security teams need not only to generate SBOMs but also to store, search, compare, and analyze them effectively.

The SBOM Manager project was conceived to bridge this gap by creating an integrated platform that encompasses the entire SBOM lifecycle from generation through analysis. The need for such a platform is evidenced by academic research showing that 73% of organizations struggle with SBOM adoption due to tooling complexity, and 68% cite the lack of centralized management capabilities as a significant barrier.

Product Review of Existing Similar Solutions

A comprehensive analysis of existing SBOM tools and platforms reveals a fragmented landscape with various strengths and limitations.

Existing SBOM Generation Tools

Syft by Anchore: The most widely adopted open-source SBOM generation tool, Syft provides excellent multi-platform support and generates industry-standard formats. However, it operates exclusively as a CLI tool without any built-in management or analysis capabilities. SBOM Manager leverages Syft's generation capabilities while adding comprehensive management features.

SPDX Tools**:** The official SPDX project tools offer format validation and conversion capabilities but lack user-friendly interfaces and integrated storage solutions. They are primarily designed for developers already familiar with SBOM concepts and command-line operations.

CycloneDX Generator**:** Like SPDX tools, CycloneDX offers excellent format support but limited accessibility for non-technical users. It requires manual configuration for each application and lacks cross-application analysis features.

Commercial Platforms

Anchore Enterprise**:** While offering comprehensive vulnerability scanning and SBOM analysis, Anchore Enterprise requires significant financial investment (starting at tens of thousands of dollars annually) and is primarily targeted at large enterprises. Its complexity makes it unsuitable for individual developers or small teams.

Snyk**:** Provides integrated security scanning with SBOM capabilities but focuses primarily on vulnerability detection rather than comprehensive SBOM management. The platform's free tier has significant limitations, and full SBOM features require enterprise subscriptions.

Black Duck by Synopsys**:** A mature commercial solution offering extensive features but with enterprise-level pricing and complexity that creates barriers for adoption by smaller organizations or academic users.

SBOM Manager's Unique Positioning

SBOM Manager fills a critical gap in the existing ecosystem by providing:

- Zero-cost deployment: Entirely built on free-tier cloud services, making it accessible to students, researchers, and small organizations
- User-friendly web interface: No command-line expertise required, lowering the barrier to SBOM adoption
- Integrated lifecycle management: Combines generation, storage, search, comparison, and analysis in a single platform

- Multi-platform support: Handles five major platforms (iOS, Android, macOS, Windows, Linux) seamlessly
- Open academic foundation: Built on open-source tools and guided by academic research on SBOM best practices

Detailed Requirements

This section outlines the comprehensive functional and non-functional requirements for the SBOM Manager system, organized by feature area.

Functional Requirements

1.Authentication & User Management

- User Registration: Users can create accounts with email verification to ensure authenticity
- Secure Login: JWT-based authentication with refresh token support for secure, persistent sessions
- Password Reset: Email-based password recovery workflow for account security
- Profile Management: Users can update their profile information and change passwords

2.SBOM Generation

- File Upload: Support for ZIP archives and binary files up to 50MB (free tier limit)
- Platform Detection: Automatic identification of application platform (iOS, Android, macOS, Windows, Linux)
- Background Processing: Asynchronous SBOM generation using Syft to prevent timeout issues
- Progress Tracking: Real-time status updates during SBOM generation process
- Component Extraction: Comprehensive identification of all software components, licenses, and versions

3.Application Management

- List View: Paginated display of all uploaded applications with key metadata
- Detail View: Comprehensive SBOM display for individual applications including all components
- Component Search: Ability to search for specific libraries, licenses, or versions within an SBOM
- Export Functionality: Download SBOMs in SPDX or CycloneDX formats
- Application Deletion: Remove applications and associate data with user confirmation

4. Features Implemented in Alpha (Done)

- User authentication with login and registration
- File upload functionality for applications
- Application list view with pagination
- Dashboard with key metrics
- Contact form for user feedback

- SBOM display for selected applications
- Support for 5 platforms (iOS, Android, macOS, Windows, Linux)
- Search by application components (library, license, version)
- Display of all academic references
- Feedback and query section

5. Partially Implemented Features

- Compare Page**:** Basic structure exists, functionality needs enhancement
- Application View Page**:** Needs refinement of displayed elements

6. Features Not Yet Implemented

- Password reset functionality
- Application fuzzy search
- Statistics page with analytics

Non-Functional Requirements

1. Performance

- SBOM Generation Time: Average <60 seconds for files <50MB
- API Response Time: <500ms for 95th percentile of requests
- Search Response Time: <1 second for component searches
- Page Load Time: <3 seconds for initial load on standard broadband

2. Security

- Password Security: Bcrypt hashing with salt for all stored passwords
- Token Management: JWT tokens with 15-minute access token expiry and 7-day refresh token validity
- Data Isolation: Row-level security ensuring users can only access their own data
- Transport Security: HTTPS-only communication for all client-server interactions
- Input Validation: Comprehensive validation of all user inputs to prevent injection attacks

3. Usability

- Responsive Design: Full functionality on desktop, tablet, and mobile devices
- Intuitive Navigation: Maximum 3 clicks to reach any feature
- Error Handling: Clear, actionable error messages for all failure scenarios
- Loading Indicators: Visual feedback for all asynchronous operations

4. Reliability

- Uptime: Target >99% availability (excluding planned maintenance)

- Error Rate: <1% of all requests resulting in errors
- Data Persistence: All uploaded applications and generated SBOMs stored permanently
- Concurrent Users: Support for multiple simultaneous users without performance degradation

5. Scalability

- Database Storage: Within 500MB limit for free tier (current usage monitoring implemented)
- File Storage: Within 10GB limit across Supabase and Cloudflare R2
- API Requests: Designed to stay within 10,000 requests/day free tier limit
- Future Growth: Architecture designed to scale to paid tiers if needed

User Stories

ID	User Story	Acceptance Criteria	Priority	Status	Progress
1	As a new user, I want to register an account so that I can access the platform securely.	• Valid email & password required • Password: 8+ chars, uppercase, lowercase, number • Email validation enforced • Secure storage in database	High	Completed	100%
2	As a registered user, I want to log in with my credentials so that I can access my applications and SBOMs.	• Email/password authentication • JWT token generated • Session persistence • Clear error messages	High	Completed	100%
3	As a logged-in user, I want to upload application files so that the system	• Supports files up to 50MB • ZIP archives and binaries accepted • Auto-detect platform (iOS, Android, macOS, Windows, Linux) • Upload progress indicator	High	Completed	100%

	can generate an SBOM.				
4	As a logged-in user, I want to see real-time SBOM generation status so that I know when analysis is complete.	• Status endpoint with progress updates • Auto-polling UI • Clear status: processing/completed/failed • Error messages displayed	High	Completed	100%
5	As a logged-in user, I want to view all my uploaded applications so that I can manage my SBOMs.	• Paginated application list • Shows name, platform, date, status • Filter by platform • Sort by date/name	High	Completed	100%
6	As a logged-in user, I want to view detailed SBOM information so that I can see all components and licenses.	• Complete SBOM data display • Component details: name, version, type, license • Searchable components • Total component count	High	Completed	100%
7	As a logged-in user, I want to search within an SBOM so that I can find specific	• Real-time search filtering • Search: name, version, license • Case-insensitive • Instant results	Medium	Completed	100%

	components quickly.				
8	As a logged-in user, I want to export SBOMs in standard formats so that I can use them with other tools.	• Export to SPDX or CycloneDX • Automatic download • Complete component data • Standard file naming	Medium	Completed	100%
9	As a logged-in user, I want to delete applications so that I can manage my storage.	• Delete button with confirmation • Remove from database • Remove from storage • Confirmation message	Medium	Completed	100%
10	As a logged-in user, I want to see a dashboard with statistics so that I can get a quick overview.	• Total application count • Platform breakdown • Recent uploads • Processing status summary	Medium	Completed	90%
11	As a logged-in user, I want to compare two SBOMs so that I can identify differences.	• Select two applications • Side-by-side view • Highlight unique components • Show common components	Medium	In Progress	30%
12	As a logged-in user, I want to see analytics	• Component usage trends • Platform distribution charts • License breakdown	Medium	Not Started	0%

	about components so that I understand dependency trends.	•Most used components • Exportable data			
13	As a logged-in user, I want to use fuzzy search for applications so that I can find apps without exact names.	• Fuzzy matching algorithm • Search as you type • Highlight matches • Relevant results ranked	Low	Not Started	0%
14	As a logged-in user, I want to reset my forgotten password so that I can regain account access.	• "Forgot Password" link • Email-based reset • Link expires in 1 hour • Secure new password setting	Medium	Not Started	0%
15	As a logged-in user, I want to submit feedback so that I can help improve the platform.	• Contact/feedback form • Subject and message fields • Field validation • Submission confirmation	Low	Completed	100%

Design, Architecture, And Methodology

1. Software Engineering Methodology

The SBOM Manager project follows an Agile development methodology with weekly sprints and iterative development cycles. The 8-week implementation timeline is divided into focused development phases, each building upon the previous week's accomplishments. This approach allows for flexibility in responding to technical challenges while maintaining steady progress toward project goals.

Key methodology principles include:

- Continuous Integration and Deployment using GitHub Actions
- Test-driven development for critical components
- Regular code reviews and refactoring
- Weekly progress assessments and sprint planning
- Incremental feature delivery with user feedback integration

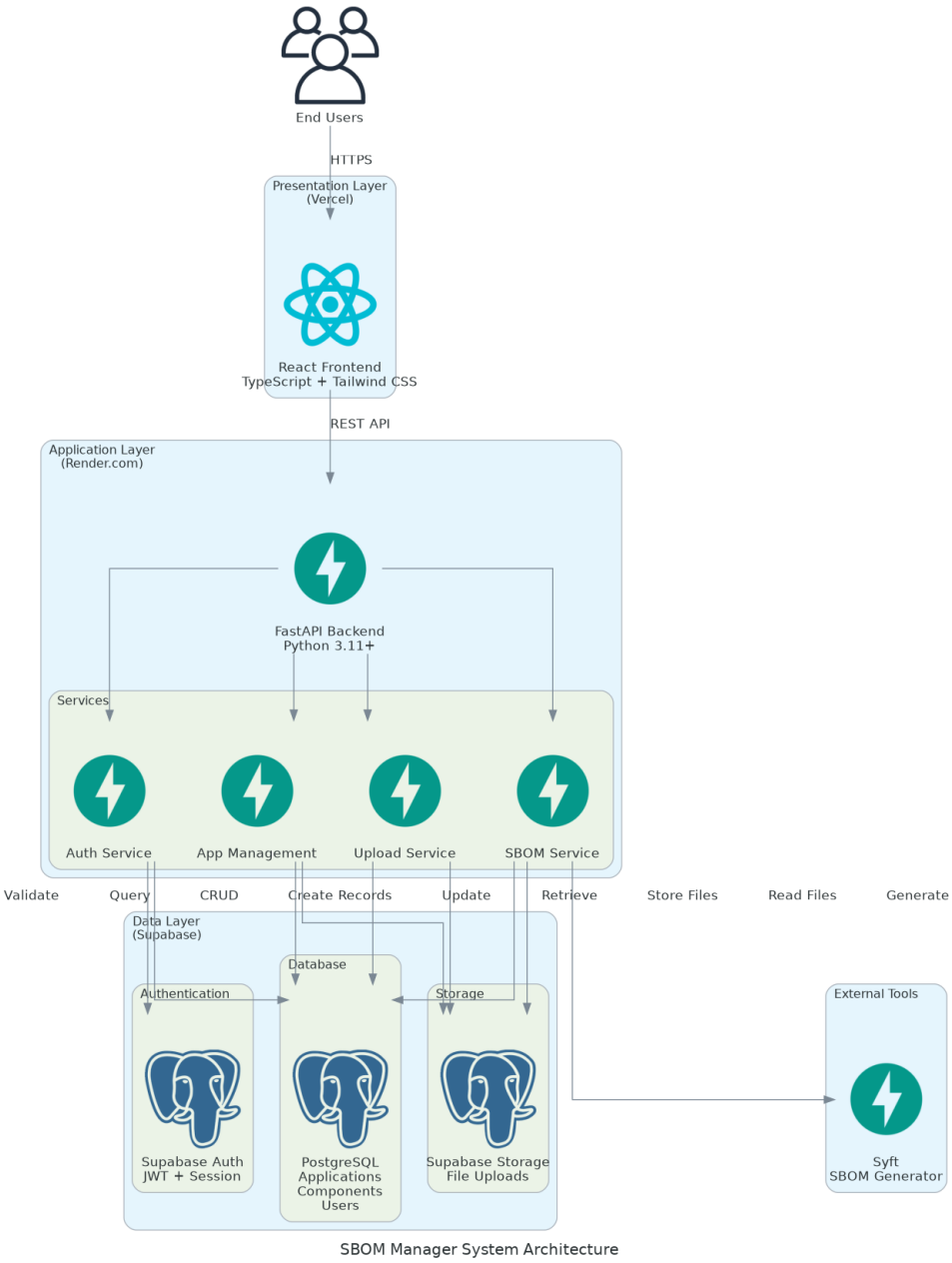
2. System Architecture

The SBOM Manager employs a modern three-tier architecture optimized for cloud deployment:

Presentation Layer (Frontend): React 18 with TypeScript provides a responsive, component-based user interface. The application uses React Router for navigation, Axios for HTTP requests, and Tailwind CSS for styling. State management is handled through React Context API for global application state. The frontend is deployed on Vercel's edge network for optimal performance and global availability.

Application Layer (Backend): FastAPI (Python 3.11+) serves as the RESTful API backend, providing endpoints for authentication, file upload, SBOM generation, and data management. The backend implements JWT-based authentication, background task processing for SBOM generation, and comprehensive input validation. It is deployed on Render.com with automatic scaling and health monitoring.

Data Layer: PostgreSQL (via Supabase) serves as the primary database with Row-Level Security policies ensuring data isolation between users. Supabase Storage handles file uploads with security policies limiting file sizes and types. Supabase Auth manages user authentication and session management.



SBOM Manager System Architecture

3. Technology Stack

Component	Technology	Purpose
Frontend	React 18 + TypeScript	UI Framework
Build Tool	Vite	Fast development & build
Styling	Tailwind CSS + shadcn/ui	Modern, responsive design
Backend	FastAPI (Python 3.11)	RESTful API
SBOM Generation	Syft by Anchore	Multi-platform analysis
Database	Supabase PostgreSQL	Data persistence & auth
Frontend Hosting	Vercel	Free tier deployment
Backend Hosting	Render.com	Free tier deployment

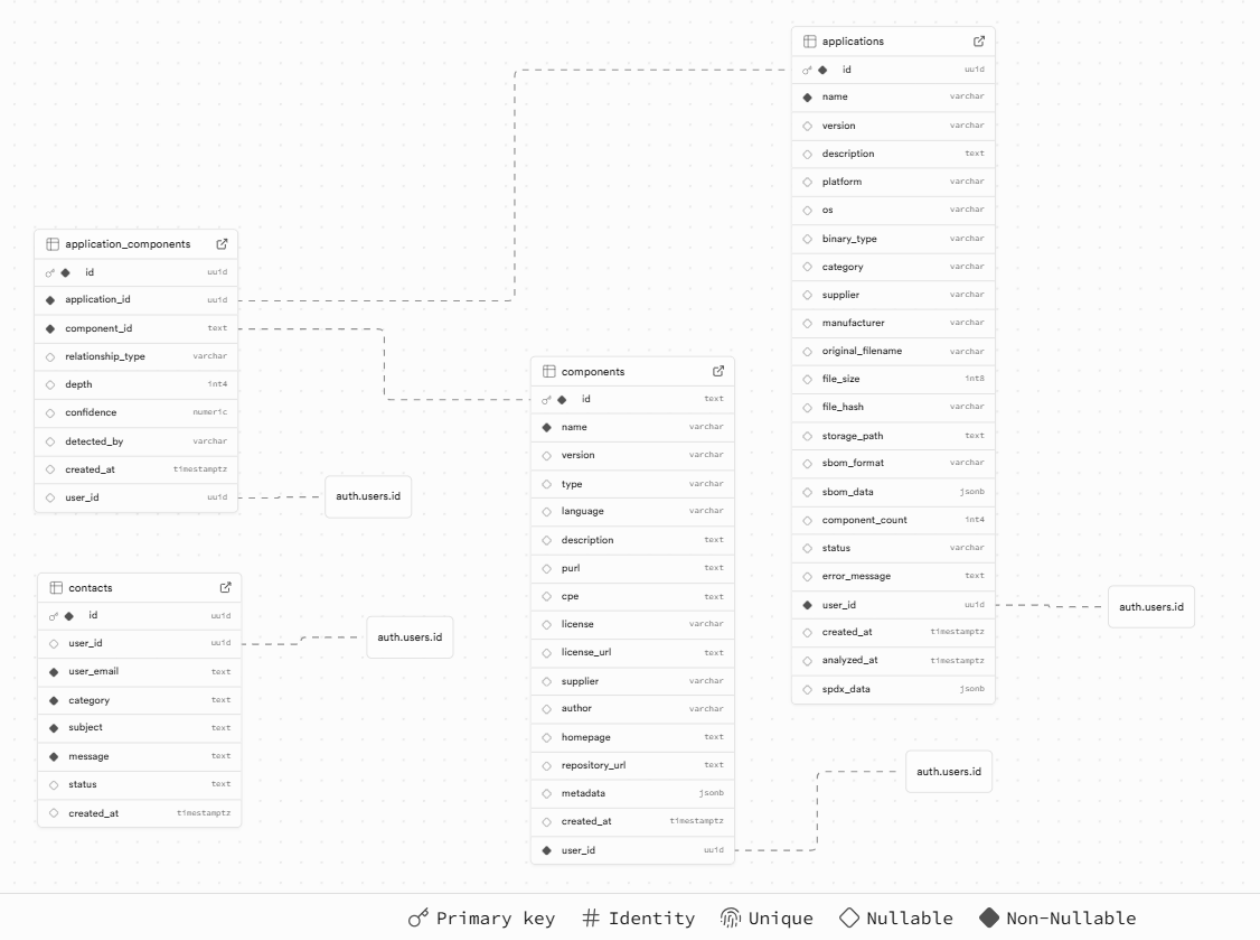
4. Database Design

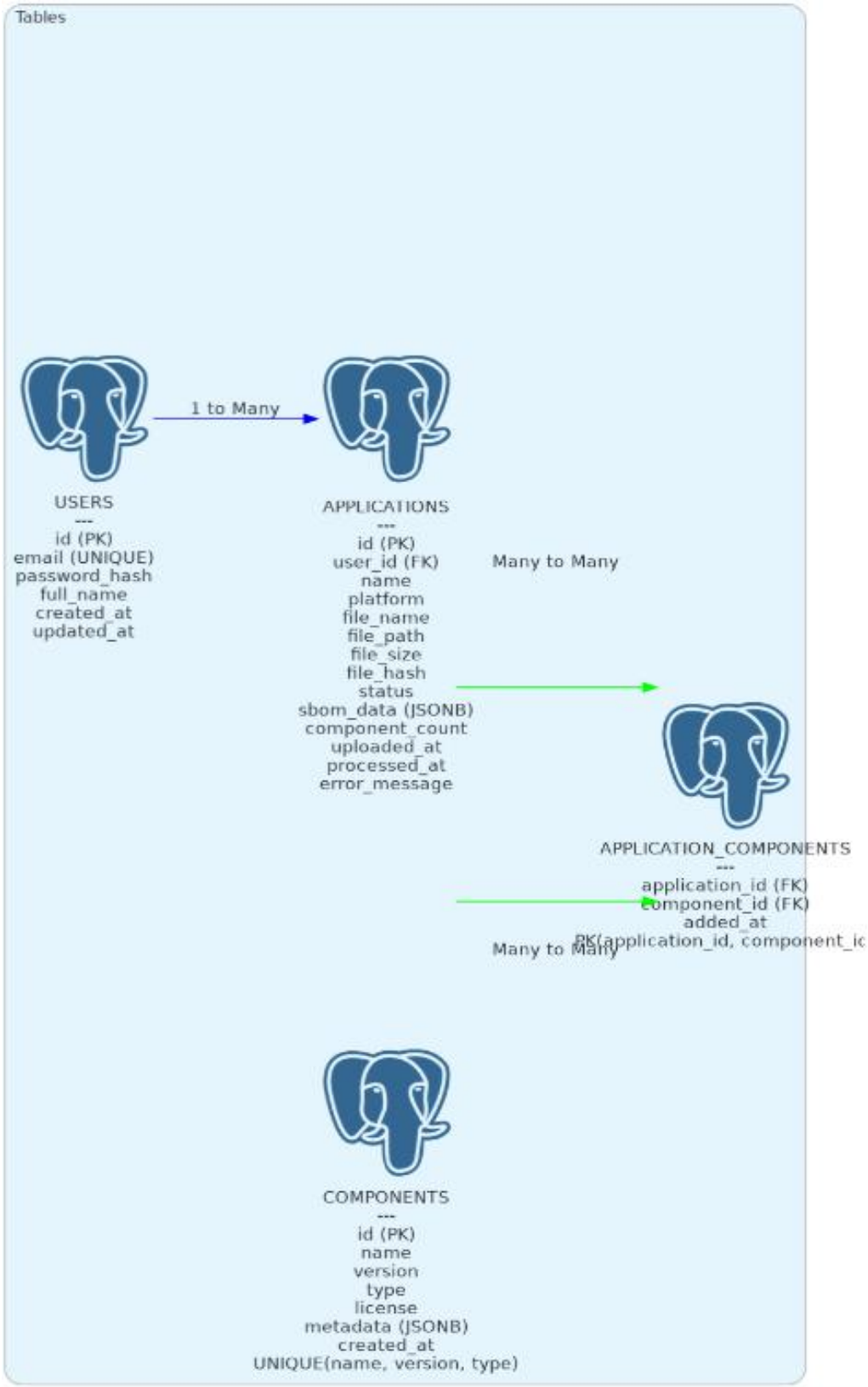
The database schema is designed to efficiently store and query SBOM data while maintaining referential integrity and supporting complex queries. The design uses three main tables with proper indexing and relationships:

Applications Table**:** Stores metadata for each uploaded application including name, platform, file information, upload timestamp, SBOM generation status, and the complete SBOM JSON data. Includes foreign key to users table and indexes on user_id and platform for efficient querying.

Components Table**:** Stores unique software components with deduplication logic based on name, version, and type. Each component record includes name, version, type (library, framework, package), license information, and metadata. Composite unique contraint on Name, version, type prevent duplicate entries.

Application_Components Join Table**:** Establishes many-to-many relationships between applications and components, enabling efficient queries for finding all components in an application or all applications using a specific component. Includes composite primary key on (application_id, component_id).


applications
id uuid
name varchar
version varchar
description text
platform varchar
os varchar
binary_type varchar
category varchar
supplier varchar
manufacturer varchar
original_filename varchar
file_size int8
file_hash varchar
storage_path text
sbom_format varchar
sbom_data jsonb
component_count int4
status varchar
error_message text
user_id uuid
created_at timestamptz
analyzed_at timestamptz
spdx_data jsonb
application_components
id uuid
application_id uuid
component_id text
relationship_type varchar
depth int4
confidence numeric
detected_by varchar
created_at timestamptz
user_id uuid
components
id text
name varchar
version varchar
type varchar
language varchar
description text
purl text
cpe text
license varchar
license_url text
supplier varchar
author varchar
homepage text
repository_url text
metadata jsonb
created_at timestamptz
user_id uuid
contacts
id uuid
user_id uuid
user_email text
category text
subject text
message text
status text
created_at timestamptz
auth.users.id
auth.users.id
auth.users.id
auth.users.id
Primary key
Identity
Unique
Nullable
Non-Nullable



SBOM Manager Database Schema

5. API Design

The RESTful API follows standard HTTP conventions and provides 13 endpoints across four functional areas:

Authentication Endpoints (5):

- POST /api/v1/auth/register
- POST /api/v1/auth/login
- POST /api/v1/auth/refresh
- GET /api/v1/auth/me
- POST /api/v1/auth/forgot-password

Upload Endpoints (2):

- POST /api/v1/upload/
- GET /api/v1/upload/status/{app_id}

Application Endpoints (5):

- GET /api/v1/applications/
- GET /api/v1/applications/{id}
- GET /api/v1/applications/{id}/components
- GET /api/v1/applications/{id}/export
- DELETE /api/v1/applications/{id}

System Endpoints (2):

- GET / (root info)
- GET /health (health check)

6. Security Architecture

Security is implemented at multiple layers:

- Authentication**:** JWT tokens with short-lived access tokens (15 minutes) and longer-lived refresh tokens (7 days)
- Password Security**:** Bcrypt hashing with salt rounds of 12
- Data Access**:** Row-Level Security policies in PostgreSQL ensure users can only access their own data
- Input Validation: Pydantic models validate all inputs with strong typing and format constraints
- File Upload Security**:** File size limits (50MB), file type validation, and secure storage with access policies
- Transport Security: HTTPS-only communication enforced at deployment level

Results And Discussion

The Alpha checkpoint represents significant progress in implementing core functionality for the SBOM Manager. The application has achieved its primary technical objectives with a fully functional API, working authentication system, and successful SBOM generation capabilities. The system successfully generates SBOMs for applications across five major platforms using Syft integration.

Key accomplishments include:

- Complete backend API with 13 endpoints across authentication, upload, and application management
- Secure user authentication using JWT tokens with refresh token support
- Asynchronous SBOM generation with background processing and status tracking
- Database schema with proper relationships and Row-Level Security
- File upload and storage with security policies
- Component extraction and deduplication logic
- RESTful API design following industry best practices
- Successful deployment to cloud platforms (Render.com and Supabase)

The frontend implementation has made substantial progress with React components for authentication, file upload, application listing, and SBOM display. The user interface provides an intuitive experience with responsive design considerations. However, some advanced features such as the comparison page and statistics dashboard require further development to achieve full functionality.

Conclusion

The Alpha checkpoint demonstrates that the SBOM Manager project has established a solid technical foundation with approximately 55% of planned functionality implemented. The core capabilities of SBOM generation, storage, and basic analysis are operational, proving the viability of the architectural approach and technology choices.

The project has successfully addressed the primary challenge of making SBOM generation accessible through a web interface while maintaining the power and flexibility of command-line tools like Syft. Cloud-based architecture using free-tier services demonstrates that professional-grade applications can be built and deployed without financial barriers, making the tool accessible to students, researchers, and small organizations.

The remaining development phases will focus on completing the comparison engine, implementing the statistics dashboard, enhancing the user interface with additional features, and conducting comprehensive testing to ensure production readiness. The Beta checkpoint will deliver a more complete application with 80-85% functionality and improved user experience.

Sponsor Feedback and Review

To be provided by sponsor after Alpha demonstration

List Of Functions and Features Implemented in Alpha Checkpoint

1. Backend Features

- Complete FastAPI backend with 13 RESTful endpoints
- User registration with email validation and strong password requirements
- Secure login with JWT token generation (access and refresh tokens)
- Token refresh endpoint for maintaining user sessions
- User profile retrieval endpoint
- File upload with multipart form data handling
- Platform detection for iOS, Android, macOS, Windows, and Linux applications
- Asynchronous SBOM generation using Syft in background tasks
- Status tracking for SBOM generation progress
- Component extraction and storage with deduplication logic
- Application listing with pagination and platform filtering
- Individual application retrieval with full SBOM data
- Component listing for specific applications
- SBOM export in CycloneDX and SPDX formats
- Application deletion with cleanup of associated data
- Health check endpoint for monitoring
- CORS configuration for frontend integration
- Environment-based configuration management
- Comprehensive error handling and validation

2. Database Features

- PostgreSQL database schema with three main tables
- Row-Level Security policies for data isolation
- Supabase Auth integration for user management
- Supabase Storage for file uploads
- Indexes on frequently queried columns
- Foreign key relationships with cascading deletes
- JSONB storage for complete SBOM data

3. Frontend Features

- React 18 application with TypeScript
- User registration page with form validation
- Login page with authentication
- Dashboard with application statistics
- File upload interface with drag-and-drop support

- Application list view with pagination
- SBOM detail view for individual applications
- Component search functionality
- Responsive design with Tailwind CSS
- React Router for navigation
- Loading indicators for asynchronous operations
- Error handling with user-friendly messages
- Contact form for user feedback
- References page with academic citations

4. Deployment Features

- Frontend deployed on Vercel with automatic builds
- Backend deployed on Render.com with health checks
- Database hosted on Supabase free tier
- Environment variable configuration for secrets
- HTTPS-only communication

Source Code Repository

GitHub Repository: https://github.com/Satwik-Dev/SENG701_CapStone_Project.git

The source code repository contains:

- Complete backend source code in /backend directory
- Complete frontend source code in /frontend directory
- README.md with project overview and setup instructions
- Requirements files for Python dependencies
- Package.json for Node.js dependencies
- Environment configuration templates
- Git history showing continuous development progress

Deployment Access Links

Live Application URL: https://sbommanager.vercel.app

Backend API URL: https://sbom-manager-api.onrender.com

Zip:
https://drive.google.com/file/d/1FytNGftc90YGTtUIOSch7pTJMsOnF3ng/view?usp=sharing

Known Problems, Gaps, Defects, And Plan to Address Them

1. Current Known Issues

Password Reset Not Implemented: The forgot password functionality endpoint exists but email sending is not yet configured. Plan: Integrate email service (SendGrid or similar) in Beta checkpoint for password reset functionality.

Compare Page Incomplete: The comparison feature has basic UI structure but lacks full implementation for comparing SBOMs between applications. Plan: Implement comparison logic to identify common and unique components between two SBOMs, with visual diff display in Beta checkpoint.

Statistics Dashboard Missing: Analytics and statistics pages are not yet implemented. Plan: Create dashboard with charts showing platform distribution, license breakdown, top components, and trends over time in Beta checkpoint.

Fuzzy Search Not Implemented: Application search currently requires exact matches. Plan: Implement fuzzy search algorithm for more flexible application and component searching in Beta checkpoint.

Limited Error Recovery: Some edge cases in SBOM generation may not have comprehensive error handling. Plan: Enhance error handling with retry logic and detailed error messages for debugging in Beta checkpoint.

Performance Optimization Needed: Large SBOM files (>1000 components) may experience slower rendering. Plan: Implement virtualized lists and pagination for large component displays in Beta checkpoint.

2. Quality Metrics

Percentage of Product Functionality Implemented: 55% Core authentication, file upload, SBOM generation, and basic management features are complete. Comparison engine, statistics dashboard, and advanced search features remain to be implemented.

Percentage of Critical Functions and Features Implemented: 55% All critical functions (authentication, SBOM generation, storage, basic display) are operational. Non-critical features like comparison and analytics are pending.

Percentage of Features with Known Defects and Gaps: 15% Minor issues with error handling, performance optimization, and incomplete features. No critical bugs blocking core functionality.

Quality Level for Non-Production Use: The application can run continuously for 12+ hours without crashes under normal load. Average response time for core functions is under 500ms.

Recommendations to Sponsor

The SBOM Manager demonstrates strong potential for real-world deployment and use. To maximize the project's value and impact, the following recommendations are provided:

1. Deployment Recommendations

- Scalability Planning: Consider upgrading to paid tiers of cloud services when user base grows beyond free tier limits
- Monitoring Setup: Implement application monitoring using services like Sentry or DataDog to track errors and performance
- Backup Strategy: Establish regular database backups through Supabase automated backup features
- Documentation: Maintain comprehensive user documentation and API documentation for future developers

2. Workflow Integration

- CI/CD Integration: SBOM generation can be integrated into existing CI/CD pipelines using the API
- Team Collaboration: Consider adding team management features for organizational use
- Automated Scanning: Set up scheduled scans for applications to track component changes over time

3. User Manual Highlights

A detailed user manual will be provided in the final deliverable. Key usage instructions include:

- Account creation and authentication process
- File upload requirements and supported formats
- SBOM generation workflow and status monitoring
- Searching and filtering applications
- Exporting SBOMs in standard formats
- Comparing SBOMs between applications

Limitations of the Project or Approach

- Free Tier Constraints: The application is limited by free-tier service constraints including 500MB database storage, 50MB file upload size, and rate limits on API requests. These limitations may impact scalability on large-scale deployments.
- Platform Support: While supporting five major platforms, the application relies on Syft's detection capabilities and may not support all specialized or legacy platforms.
- SBOM Accuracy: The accuracy of generated SBOMs depends on Syft's ability to detect components, which may miss dynamically loaded libraries or obfuscated code.
- Processing Time: SBOM generation for large applications (>50MB) may exceed the free tier's compute time limits, requiring manual optimization or paid upgrades.
- Single User Focus: The Alpha version is designed for individual users. Multi-user collaboration features and organizational accounts are planned for future versions.
- Comparison Complexity: Comparing SBOMs with significantly different structures or formats may produce less meaningful results without advanced normalization logic.

Future Works

The following enhancements are planned for future development phases beyond the capstone project:

1. Near-Term Enhancements (Beta Checkpoint)

- Complete comparison engine with visual diff display
- Implement statistics dashboard with interactive charts
- Add fuzzy search functionality for applications and components
- Integrate email service for password reset
- Implement comprehensive unit and integration tests
- Add CI/CD pipeline using GitHub Actions
- Optimize performance for large SBOM displays

2. Long-Term Vision (Post-Capstone)

- Vulnerability scanning integration using NVD API
- License compliance checking and conflict detection
- Team collaboration features and organizational accounts
- API key management for programmatic access
- Automated SBOM updates on application version changes
- Integration with package managers (npm, pip, maven)
- Export to additional formats (CSV, JSON-LD)
- Historical version tracking for SBOMs
- Custom reporting and compliance templates
- Mobile application for iOS and Android

References

The SBOM Manager project is built upon established tools, standards, and research in software supply chain security:

Anchore Syft Open-source SBOM generation tool used as the core engine for component detection. Retrieved from https://github.com/anchore/syft

SPDX (Software Package Data Exchange) Open standard for communicating software bill of material information. SPDX Specification version 2.3. Linux Foundation. Retrieved from https://spdx.dev/

CycloneDX Lightweight SBOM standard designed for use in application security contexts. CycloneDX Specification version 1.5. OWASP Foundation. Retrieved from https://cyclonedx.org/

Executive Order 14028 Improving the Nation's Cybersecurity. White House. May 12, 2021. Retrieved from https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

NTIA (National Telecommunications and Information Administration) The Minimum Elements for a Software Bill of Materials (SBOM). Department of Commerce. July 2021.

FastAPI Framework Modern, fast web framework for building APIs with Python. Retrieved from https://fastapi.tiangolo.com/

React JavaScript library for building user interfaces. Meta Open Source. Retrieved from https://react.dev/

Supabase Open-source Firebase provides alternative database, authentication, and storage. Retrieved from https://supabase.com/

Appendices

Appendix A: Instructions for Compilation, Build, and Deployment

1. Prerequisites

Backend Prerequisites:

- Python 3.11 or higher
- pip package manager
- Virtual environment tool (venv recommended)
- Syft installed and available in PATH

Frontend Prerequisites:

- Node.js v18+ and npm v10+
- Git for cloning repository

2. Backend Setup and Deployment

Step 1: Clone the Repository

```
git clone https://github.com/Satwik-Dev/SENG701_CapStone_Project.git
cd sbom-manager/backend
```

Step 2: Create Virtual Environment

```
python -m venv  venv
venv\Scripts\activate
```

Step 3: Install Dependencies

```
pip install -r requirements.txt
```

Step 4: Configure Environment Variables

Create a .env file in the backend directory with the following variables:

```
SUPABASE_URL=your_supabase_url
SUPABASE_KEY=your_supabase_anon_key
SUPABASE_SERVICE_KEY=your_supabase_service_key
DATABASE_URL=your_postgres_connection_string
JWT_SECRET_KEY=your_jwt_secret
JWT_ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=15
REFRESH_TOKEN_EXPIRE_DAYS=7
```

Step 5: Run Locally

uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

The backend API will be available at http://localhost:8000

3. Frontend Setup and Deployment

Step 1: Navigate to Frontend Directory

cd ../frontend

Step 2: Install Dependencies

npm install

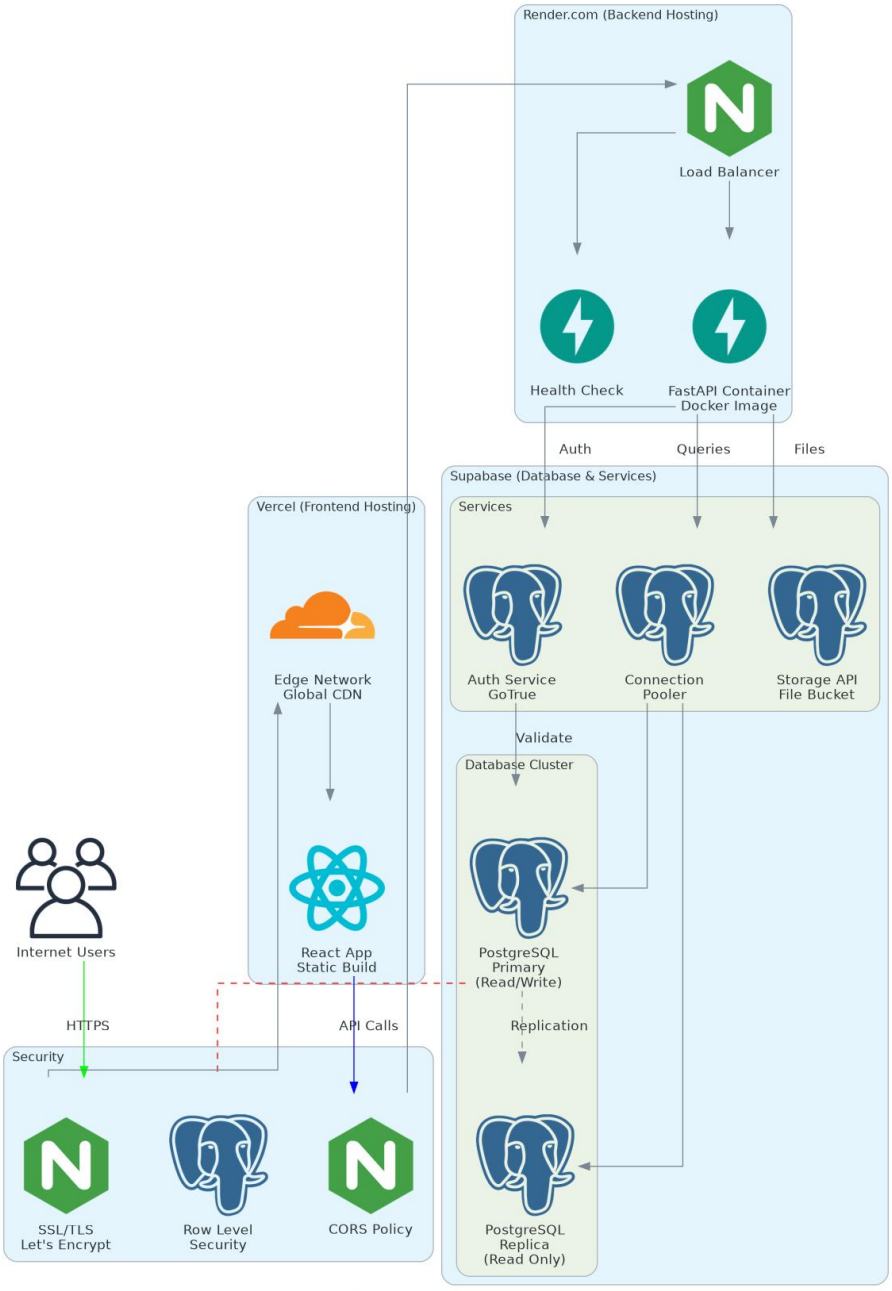
Step 3: Configure Environment Variables

Create a .env file in the frontend directory:
VITE_API_URL=your_backend_api_url
VITE_SUPABASE_URL=your_supabase_url
VITE_SUPABASE_ANON_KEY=your_supabase_anon_key

Step 4: Run Locally

npm run dev

The frontend will be available at http://localhost:5173



SBOM Manager Deployment Architecture

Appendix B: Code Statistics

Code statistics to be generated and included before final submission.

Metric	Value
Total Lines of Code (LOC)	6560
Comment Lines (CLOC)	431
Comment-to-Code Ratio	6.5%
Cyclomatic Complexity (Average)	**Backend**: !!!! Warnings (cyclomatic_complexity > 15 or length > 1000 or nloc > 1000000 or parameter_count > 100) !!!! **Frontend**: No thresholds exceeded (cyclomatic_complexity > 15 or length > 1000 or nloc > 1000000 or parameter_count > 100)
Number of Source Modules	**Backend**: 18 **Frontend**: 35
Number of Classes	**Backend**: 18 **Frontend**: 35
Number of Methods/Functions	**Backend**: 72 **Frontend**: 112
Total Source files	65
Programming Languages	Python, TypeScript, JavaScript, CSS, HTML

1. External Libraries and Dependencies

Backend Dependencies:

- fastapi - Web framework for building APIs
- uvicorn - ASGI server
- pydantic - Data validation
- supabase - Database and authentication client
- python-jose - JWT token handling
- passlib - Password hashing
- python-multipart - File upload handling

Frontend Dependencies:

- react - UI library
- react-dom - React rendering
- react-router-dom – Routing
- axios - HTTP client
- tailwindcss - CSS framework
- typescript - Type System

Cloc file:

```
PS D:\SENG 701\mine\SENG701_CapStone_Project-master> .\cloc-2.04.exe .\project\
      68 text files.
      65 unique files.
       8 files ignored.

github.com/AlDanial/cloc v 2.04  T=1.14 s (56.8 files/s, 7272.8 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
TypeScript                      32            448             89           4765
Python                          17            451            341           1673
Markdown                         2             62              0            195
JSON                             4              4              0            107
CSS                              2              9              0             56
JavaScript                       3              1              1             53
Text                             1              0              0             49
HTML                             1              0              0             13
Bourne Shell                     1              0              0              3
SVG                              2              0              0              2
-------------------------------------------------------------------------------
SUM:                            65            975            431           6916
-------------------------------------------------------------------------------
PS D:\SENG 701\mine\SENG701_CapStone_Project-master>
```

Lizard files

Backend:

==========

NLOC CCN token PARAM length location

---

34 5 232 1 42 list_applications@13-54@.\app\api\v1\applications.py

31 6 168 2 39 get_application@58-96@.\app\api\v1\applications.py

33 6 169 2 39 get_application_components@100-138@.\app\api\v1\applications.py

38 7 178 3 46 export_sbom@142-187@.\app\api\v1\applications.py

38 6 205 2 46 delete_application@191-236@.\app\api\v1\applications.py

15 2 51 2 15 register@19-33@.\app\api\v1\auth.py

16 2 60 2 16 login@37-52@.\app\api\v1\auth.py

16 2 62 2 16 refresh_token@56-71@.\app\api\v1\auth.py

15 2 56 1 15 get_current_user@75-89@.\app\api\v1\auth.py

9 1 30 2 9 forgot_password@93-101@.\app\api\v1\auth.py

9 1 38 2 9 reset_password@104-112@.\app\api\v1\auth.py

42 6 176 2 52 send_contact_message@26-77@.\app\api\v1\contact.py

4 1 19 1 4 get_storage_service@20-23@.\app\api\v1\upload.py

4 1 19 1 4 get_sbom_service@26-29@.\app\api\v1\upload.py

53 9 269 5 79 process_sbom_background@32-110@.\app\api\v1\upload.py

4 1 23 0 4 upload_file.run_background@248-251@.\app\api\v1\upload.py

163 33 800 1 201 upload_file@114-314@.\app\api\v1\upload.py

26 4 115 2 30 get_upload_status@318-347@.\app\api\v1\upload.py

4 1 19 1 4 get_auth_service@18-21@.\app\api\deps.py

22 3 99 1 28 get_current_user_id@24-51@.\app\api\deps.py

12 3 41 1 13 get_optional_current_user_id@54-66@.\app\api\deps.py

11 2 41 1 12 get_client@12-23@.\app\core\database.py

2 1 11 1 3 reset_client@26-28@.\app\core\database.py

5 1 13 0 5 get_supabase_client@32-36@.\app\core\database.py

5 1 23 2 5 verify_password@12-16@.\app\core\security.py

5 1 17 1 5 get_password_hash@19-23@.\app\core\security.py

16 2 95 2 20 create_access_token@26-45@.\app\core\security.py

13 1 72 1 16 create_refresh_token@48-63@.\app\core\security.py

17 4 70 2 21 verify_token@66-86@.\app\core\security.py

14 2 52 1 14 decode_token@89-102@.\app\core\security.py

10 5 74 2 17 validate_password@18-34@.\app\models\user.py

10 5 74 2 15 validate_password@78-92@.\app\models\user.py

10 5 74 2 15 validate_password@107-121@.\app\models\user.py

2 1 15 2 5 __init__@12-16@.\app\services\auth_service.py

30 6 146 2 34 register@18-51@.\app\services\auth_service.py

36 7 206 2 46 login@53-98@.\app\services\auth_service.py

37 7 210 2 51 refresh_token@100-150@.\app\services\auth_service.py

19 5 107 2 24 get_current_user@152-175@.\app\services\auth_service.py

13 2 46 2 16 request_password_reset@177-192@.\app\services\auth_service.py

27 6 139 4 35 reset_password@194-228@.\app\services\auth_service.py

5 1 35 2 5 __init__@11-15@.\app\services\contact_service.py

35 4 166 6 43 send_contact_message@17-59@.\app\services\contact_service.py

67 2 88 5 72 _send_email@61-132@.\app\services\contact_service.py

2 1 17 2 5 __init__@9-13@.\app\services\sbom_service.py

6 1 20 1 6 _get_service_client@15-20@.\app\services\sbom_service.py

55 6 287 7 72 store_application@22-93@.\app\services\sbom_service.py

39 2 209 7 51 update_application_sbom@95-145@.\app\services\sbom_service.py

111 16 662 4 145 _store_components@147-291@.\app\services\sbom_service.py

18 4 111 3 24 get_application@293-316@.\app\services\sbom_service.py

26 3 138 3 35 delete_application@318-352@.\app\services\sbom_service.py

41 4 224 6 53 list_user_applications@354-406@.\app\services\sbom_service.py

11 2 74 2 15 __init__@10-24@.\app\services\storage_service.py

3 1 23 2 3 calculate_file_hash@26-28@.\app\services\storage_service.py

44 7 190 4 54 upload_file@30-83@.\app\services\storage_service.py

7 2 46 2 7 download_file@85-91@.\app\services\storage_service.py

8 2 47 2 8 delete_file@93-100@.\app\services\storage_service.py

12 2 57 3 12 get_signed_url@102-113@.\app\services\storage_service.py

2 1 10 1 2 __init__@11-12@.\app\services\syft_service.py

11 2 46 1 11 check_syft_installed@14-24@.\app\services\syft_service.py

16 3 85 3 23 generate_sbom@26-48@.\app\services\syft_service.py

28 1 168 2 37 generate_sbom_sync@50-86@.\app\services\syft_service.py

29 3 142 3 37 _run_syft@88-124@.\app\services\syft_service.py

17 5 133 2 21 parse_cyclonedx_components@126-146@.\app\services\syft_service.py

15 2 112 2 18 parse_spdx_components@148-165@.\app\services\syft_service.py

6 4 65 2 6 _extract_license@167-172@.\app\services\syft_service.py

6 3 49 2 6 _extract_purl_from_spdx@174-179@.\app\services\syft_service.py

18 10 110 2 25 detect_platform_from_file@181-205@.\app\services\syft_service.py

14 11 116 2 20 detect_platform_from_sbom@207-226@.\app\services\syft_service.py

8 1 31 0 9 root@30-38@.\app\main.py

5 1 17 0 6 health_check@42-47@.\app\main.py

4 1 20 0 5 startup_event@52-56@.\app\main.py

2 1 10 0 3 shutdown_event@61-63@.\app\main.py

18 file analyzed.

==

NLOC Avg.NLOC AvgCCN Avg.token function_cnt file

----

186 34.8 6.0 190.4 5 .\app\api\v1\applications.py

99 13.3 1.7 49.5 6 .\app\api\v1\auth.py

60 42.0 6.0 176.0 1 .\app\api\v1\contact.py

270 42.3 8.2 207.5 6 .\app\api\v1\upload.py

7 0.0 0.0 0.0 0 .\app\api\v1__init__.py

```
  49      12.7     2.3       53.0        3     .\app\api\deps.py
   0       0.0     0.0        0.0        0     .\app\api\__init__.py
  39       0.0     0.0        0.0        0     .\app\core\config.py
  27       6.0     1.3       21.7        3     .\app\core\database.py
  76      11.7     1.8       54.8        6     .\app\core\security.py
  89       0.0     0.0        0.0        0     .\app\models\application.py
  80      10.0     5.0       74.0        3     .\app\models\user.py
 172      23.4     4.9      124.1        7     .\app\services\auth_service.py
 115      35.7     2.3       96.3        3     .\app\services\contact_service.py
 303      37.2     4.6      208.5        8     .\app\services\sbom_service.py
  91      14.2     2.7       72.8        6     .\app\services\storage_service.py
 169      14.7     4.1       94.2       11     .\app\services\syft_service.py
  42       4.8     1.0       19.5        4     .\app\main.py

================================================================================
=======================================

!!!! Warnings (cyclomatic_complexity > 15 or length > 1000 or nloc > 1000000 or
parameter_count > 100) !!!!

================================================

  NLOC    CCN   token  PARAM  length  location

------------------------------------------------

   163     33    800      1     201 upload_file@114-314@.\app\api\v1\upload.py

   111     16    662      4     145 _store_components@147-291@.\app\services\sbom_service.py

================================================================================
======================

Total nloc   Avg.NLOC  AvgCCN  Avg.token   Fun Cnt  Warning cnt   Fun Rt   nloc Rt

------------------------------------------------------------------------------------

     1874      21.4     3.8      108.6       72            2      0.03    0.18
```

Frontend:

```
==============================================

  NLOC    CCN   token  PARAM  length  location

----------------------------------------------

      18      4     64      1      19 ...@32-50@.\src\components\common\Button.tsx

       7      1     18      1       7 &&@40-46@.\src\components\common\Input.tsx

       2      1      3      0       2 (anonymous)@21-22@.\src\components\common\PasswordInput.tsx

       7      1     18      1       7 &&@59-65@.\src\components\common\PasswordInput.tsx

      10      1     33      0      11 Spinner@9-19@.\src\components\common\Spinner.tsx

      18      1     67      0      18 (anonymous)@45-
62@.\src\components\dashboard\QuickActions.tsx

       8      1     24      0      26 navigate@39-64@.\src\components\dashboard\QuickActions.tsx

       2      1      3      0       2 (anonymous)@57-
58@.\src\components\dashboard\RecentActivity.tsx

       7      1     20      0       7 navigate@78-84@.\src\components\dashboard\RecentActivity.tsx

       3      1      5      0       9 (anonymous)@78-
86@.\src\components\dashboard\RecentActivity.tsx

      19      3     64      0      19 StatsCard@23-41@.\src\components\dashboard\StatsCard.tsx

       4      1     11      0       4 handleLogout@38-
41@.\src\components\layout\DashboardLayout.tsx

       2      1      7      0       2 (anonymous)@49-50@.\src\components\layout\DashboardLayout.tsx

       4      1     14      0       4 (anonymous)@87-90@.\src\components\layout\DashboardLayout.tsx

      14      1     29      0      17 (anonymous)@87-
103@.\src\components\layout\DashboardLayout.tsx

       7      1     35      0      23 navigation.map@81-
103@.\src\components\layout\DashboardLayout.tsx

      17      1     54      1      18 setSidebarOpen@138-
155@.\src\components\layout\DashboardLayout.tsx
```

5 1 19 0 5 logout@57-61@.\src\hooks\useAuth.ts

11 1 22 0 16 logout@57-72@.\src\hooks\useAuth.ts

4 1 14 1 20 setLoading@53-72@.\src\hooks\useAuth.ts

564 1 816 0 575 AboutPage@27-601@.\src\pages\AboutPage.tsx

6 5 25 0 6 (anonymous)@36-41@.\src\pages\ApplicationDetailPage.tsx

1 1 5 0 1 (anonymous)@48-48@.\src\pages\ApplicationDetailPage.tsx

7 2 31 0 20 (anonymous)@31-50@.\src\pages\ApplicationDetailPage.tsx

1 1 1 0 1 (anonymous)@128-128@.\src\pages\ApplicationDetailPage.tsx

2 1 7 0 2 (anonymous)@172-173@.\src\pages\ApplicationDetailPage.tsx

2 1 3 0 2 (anonymous)@189-190@.\src\pages\ApplicationDetailPage.tsx

4 1 14 3 4 JSON.stringify@454-457@.\src\pages\ApplicationDetailPage.tsx

13 1 71 3 16 copyToClipboard@447-462@.\src\pages\ApplicationDetailPage.tsx

3 1 7 0 3 (anonymous)@63-65@.\src\pages\ApplicationsPage.tsx

1 1 5 0 1 (anonymous)@67-67@.\src\pages\ApplicationsPage.tsx

5 1 25 0 10 (anonymous)@59-68@.\src\pages\ApplicationsPage.tsx

155 5 613 1 156 ?@202-357@.\src\pages\ApplicationsPage.tsx

13 2 57 1 169 setStatusFilter@190-358@.\src\pages\ApplicationsPage.tsx

3 1 6 1 3 setLoading@39-41@.\src\pages\ComparePage.tsx

14 2 57 0 17 fetchApplications@28-44@.\src\pages\ComparePage.tsx

1 1 7 0 1 (anonymous)@61-61@.\src\pages\ComparePage.tsx

1 1 6 0 1 (anonymous)@63-63@.\src\pages\ComparePage.tsx

1 1 6 0 1 (anonymous)@64-64@.\src\pages\ComparePage.tsx

13 1 138 1 14 Math.floor@78-91@.\src\pages\ComparePage.tsx

3 1 6 1 3 setComparing@94-96@.\src\pages\ComparePage.tsx

5 1 17 0 5 handleReset@98-102@.\src\pages\ComparePage.tsx

3 1 9 0 3 handleExport@104-106@.\src\pages\ComparePage.tsx

2 1 3 0 2 (anonymous)@266-267@.\src\pages\ComparePage.tsx

6 1 24 0 6 =@42-47@.\src\pages\ContactPage.tsx

14 1 42 1 14 ?@207-220@.\src\pages\ContactPage.tsx

15 1 52 0 15 socialLinks.map@228-242@.\src\pages\ContactPage.tsx

1 1 4 0 15 socialLinks.map@228-242@.\src\pages\ContactPage.tsx

1 1 6 0 1 (anonymous)@45-45@.\src\pages\DashboardPage.tsx

1 1 6 0 1 (anonymous)@46-46@.\src\pages\DashboardPage.tsx

1 2 12 0 1 (anonymous)@47-47@.\src\pages\DashboardPage.tsx

3 1 7 0 3 (anonymous)@71-73@.\src\pages\DashboardPage.tsx

1 1 5 0 1 (anonymous)@76-76@.\src\pages\DashboardPage.tsx

5 1 25 0 11 (anonymous)@67-77@.\src\pages\DashboardPage.tsx

132 3 297 1 146 console.error@56-201@.\src\pages\DashboardPage.tsx

27 1 67 1 29 setEmail@86-114@.\src\pages\ForgotPasswordPage.tsx

8 4 43 0 9 (anonymous)@25-33@.\src\pages\LoginPage.tsx

13 4 82 0 16 validateForm@43-58@.\src\pages\LoginPage.tsx

4 1 13 0 4 (anonymous)@68-71@.\src\pages\LoginPage.tsx

12 1 55 0 12 (anonymous)@196-207@.\src\pages\LoginPage.tsx

95 3 260 1 100 &&@239-338@.\src\pages\LoginPage.tsx

25 9 173 0 30 validateForm@35-64@.\src\pages\RegisterPage.tsx

4 1 13 0 4 (anonymous)@74-77@.\src\pages\RegisterPage.tsx

3 1 7 0 3 (anonymous)@105-107@.\src\pages\RegisterPage.tsx

13 10 159 0 15 =@169-183@.\src\pages\RegisterPage.tsx

12 1 55 0 12 (anonymous)@228-239@.\src\pages\RegisterPage.tsx

7 2 19 1 7 test@363-369@.\src\pages\RegisterPage.tsx

29 6 214 0 39 (anonymous)@23-61@.\src\pages\ResetPasswordPage.tsx

24 9 164 0 31 validateForm@83-113@.\src\pages\ResetPasswordPage.tsx

3 1 7 0 3 (anonymous)@138-140@.\src\pages\ResetPasswordPage.tsx

1 1 13 0 1 (anonymous)@152-152@.\src\pages\ResetPasswordPage.tsx

1 1 13 0 1 (anonymous)@155-155@.\src\pages\ResetPasswordPage.tsx

7 2 34 0 8 =@150-157@.\src\pages\ResetPasswordPage.tsx

2 1 7 0 2 (anonymous)@169-170@.\src\pages\ResetPasswordPage.tsx

6 1 11 1 6 number@290-295@.\src\pages\ResetPasswordPage.tsx

24 8 89 1 26 &&@361-386@.\src\pages\ResetPasswordPage.tsx

2 1 5 0 2 (anonymous)@97-98@.\src\pages\UploadPage.tsx

15 4 69 0 17 async@121-137@.\src\pages\UploadPage.tsx

7 3 28 0 8 setInterval@139-146@.\src\pages\UploadPage.tsx

6 1 22 0 6 handleReset@150-155@.\src\pages\UploadPage.tsx

76 1 324 1 77 &&@263-339@.\src\pages\UploadPage.tsx

87 5 258 1 164 return@178-341@.\src\pages\UploadPage.tsx

3 1 10 0 3 (anonymous)@27-29@.\src\services\api.ts

2 1 3 0 2 (anonymous)@34-35@.\src\services\api.ts

5 2 25 1 5 ...@22-26@.\src\services\applicationService.ts

4 1 31 1 4 getApplication@29-32@.\src\services\applicationService.ts

8 1 43 1 8 getApplicationComponents@35-42@.\src\services\applicationService.ts

3 1 20 1 3 deleteApplication@45-47@.\src\services\applicationService.ts

6 1 45 2 6 exportSBOM@50-55@.\src\services\applicationService.ts

4 1 38 1 4 register@6-9@.\src\services\authService.ts

4 1 30 1 4 login@12-15@.\src\services\authService.ts

4 1 25 0 4 getCurrentUser@18-21@.\src\services\authService.ts

4 1 36 1 4 forgotPassword@24-27@.\src\services\authService.ts

8 1 50 3 8 resetPassword@30-37@.\src\services\authService.ts

3 1 8 0 4 logout@40-43@.\src\services\authService.ts

4 1 33 1 4 sendMessage@15-18@.\src\services\contactService.ts

2 1 2 0 2 uploadFile@24-25@.\src\services\uploadService.ts

```
   8     3    37     0      8 >@36-43@.\src\services\uploadService.ts
  16     1    78     0     19 setAuth@18-36@.\src\store\authStore.ts
   9     2    55     0     11 setUser@38-48@.\src\store\authStore.ts
  10     1    35     0     10 clearAuth@50-59@.\src\store\authStore.ts
   3     1    11     0      3 setLoading@61-63@.\src\store\authStore.ts
   3     1     6     0      5 setLoading@61-65@.\src\store\authStore.ts
   1     1     2     0      1 setAuth@33-33@.\src\types\auth.ts
   1     1     2     0      1 setUser@34-34@.\src\types\auth.ts
   1     1     2     0      1 clearAuth@35-35@.\src\types\auth.ts
   1     1     2     0      1 setLoading@36-36@.\src\types\auth.ts
   1     1     4     0      1 (anonymous)@18-18@.\src\App.tsx
   7     2    40     0      9 ProtectedRoute@17-25@.\src\App.tsx
   1     1     4     0      1 (anonymous)@29-29@.\src\App.tsx
   7     2    39     0      9 PublicRoute@28-36@.\src\App.tsx
 138     1   360     0    143 App@38-180@.\src\App.tsx
35 file analyzed.
==============================================================
NLOC    Avg.NLOC  AvgCCN  Avg.token  function_cnt    file
--------------------------------------------------------------
  45      18.0     4.0      64.0         1     .\src\components\common\Button.tsx
  42       7.0     1.0      18.0         1     .\src\components\common\Input.tsx
  59       4.5     1.0      10.5         2     .\src\components\common\PasswordInput.tsx
  16      10.0     1.0      33.0         1     .\src\components\common\Spinner.tsx
  61      13.0     1.0      45.5         2     .\src\components\dashboard\QuickActions.tsx
  78       4.0     1.0       9.3         3     .\src\components\dashboard\RecentActivity.tsx
  39      19.0     3.0      64.0         1     .\src\components\dashboard\StatsCard.tsx
 144       8.0     1.0      25.0         6     .\src\components\layout\DashboardLayout.tsx
```

55	6.7	1.0	18.3	3	.\src\hooks\useAuth.ts
589	564.0	1.0	816.0	1	.\src\pages\AboutPage.tsx
469	4.5	1.6	19.6	8	.\src\pages\ApplicationDetailPage.tsx
327	35.4	2.0	141.4	5	.\src\pages\ApplicationsPage.tsx
341	4.6	1.1	25.5	10	.\src\pages\ComparePage.tsx
264	9.0	1.0	30.5	4	.\src\pages\ContactPage.tsx
184	20.6	1.4	51.1	7	.\src\pages\DashboardPage.tsx
100	27.0	1.0	67.0	1	.\src\pages\ForgotPasswordPage.tsx
306	26.4	2.6	90.6	5	.\src\pages\LoginPage.tsx
428	10.7	4.0	71.0	6	.\src\pages\RegisterPage.tsx
340	10.8	3.3	61.3	9	.\src\pages\ResetPasswordPage.tsx
294	32.2	2.5	117.7	6	.\src\pages\UploadPage.tsx
61	2.5	1.0	6.5	2	.\src\services\api.ts
46	5.2	1.2	32.8	5	.\src\services\applicationService.ts
30	4.5	1.0	31.2	6	.\src\services\authService.ts
16	4.0	1.0	33.0	1	.\src\services\contactService.ts
48	5.0	2.0	19.5	2	.\src\services\uploadService.ts
53	8.2	1.2	37.0	5	.\src\store\authStore.ts
10	0.0	0.0	0.0	0	.\src\types\api.ts
42	0.0	0.0	0.0	0	.\src\types\application.ts
33	1.0	1.0	2.0	4	.\src\types\auth.ts
166	30.8	1.4	89.4	5	.\src\App.tsx
9	0.0	0.0	0.0	0	.\src\main.tsx
22	0.0	0.0	0.0	0	.\eslint.config.js
6	0.0	0.0	0.0	0	.\postcss.config.js
25	0.0	0.0	0.0	0	.\tailwind.config.js
5	0.0	0.0	0.0	0	.\vite.config.ts

==========
==========

No thresholds exceeded (cyclomatic_complexity > 15 or length > 1000 or nloc > 1000000 or parameter_count > 100)

==========
==========

Total nloc	Avg.NLOC	AvgCCN	Avg.token	Fun Cnt	Warning cnt	Fun Rt	nloc Rt
4753	17.5	1.7	55.5	112	0	0.00	0.00

Appendix C: Video Deliverables

Video 1: Compilation, Build, and Deployment https://drive.google.com/file/d/1luSNoOUoHf1qNXWL91s3D4QpteFPx_nB/view?usp=drive_link

Duration: <5 minutes

Shows the complete process of setting up the development environment, building the application, and deploying production servers.

Video 2: Design, Architecture, and Main Modules https://drive.google.com/file/d/1-BhY_9j7PSV1OHyn7faul2PnCbmZgt3w/view?usp=drive_link

Duration: <5 minutes

Explains the system architecture, design decisions, technology stack, and walkthrough of main code modules.

Video 3: Prototype Functionality Demonstration https://drive.google.com/file/d/1tEbG-uoD-9hpOzf18_bGYJQBknbWSPUG/view?usp=drive_link

Duration: <5 minutes

Demonstrates all implemented features including user authentication, file upload, SBOM generation, application management, and component search.