Smart Tagged Inventory Management System

STIMS

Jack Margeson

Project Abstract

STIMS, the Smart Tagged Inventory Management System is a database driven storage catalog capable of adding and tracking any combination of real world items. This multi-modal storage paradigm is achieved through the use of many tagging systems, in comparison to other inventory databases using simply one for a specific type of physical media.

The goal of STIMS is to provide database administrators with the tools to easily catalog new items and even create new object types for tracking, while providing a streamlined search and recall dashboard for end-users to facilitate inventory check-in and check-out operations while maintaining database security throughout the whole process.

Purpose and Goals

Usability

- Users don’t interact with the database system directly
- Database catalogers can add new items from the GUI app
- User onboarding process is quick and self-guided

Simplicity

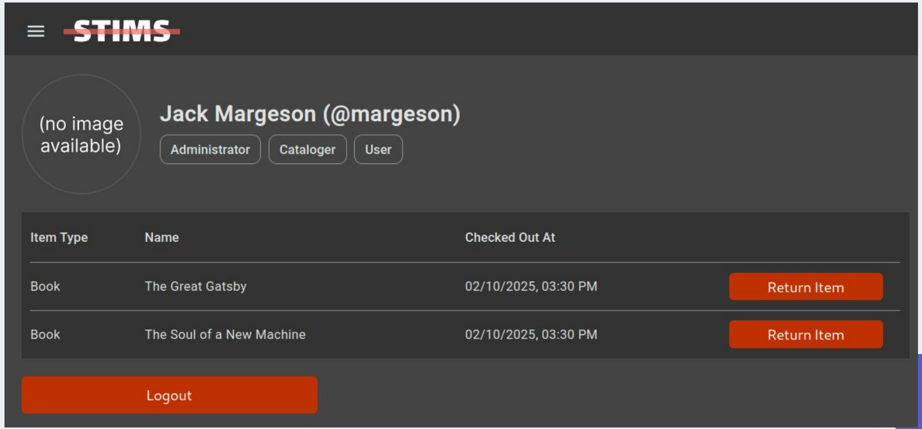
- Easy to understand GUI web application
- Utilizes Docker to facilitate easy set-up by database admins
- Works on any device capable of displaying web content

Customizability

- Account system allows for options and saved queries
- New item types (classes) can be added by administrators
- Item filters and sorting allow for user dashboard modification

Intellectual Merits

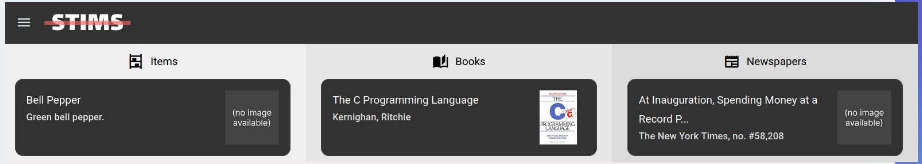
- Automated self-defined class system
  - Need to keep track of newspapers? Simply define what details a newspaper should include and STIMS takes care of the rest.
- Integrated user role system
  - System administrators can give user accounts permissions to interact with certain features in the program, such as intake
- Clean, user-friendly UI and design
  - Maximizes usability over niche functionality
  - Designed as a web application to be portable and functional on any device



Jack Margeson (@margeson)

Administrator · Cataloger · User

Item Type	Name	Checked Out At	
Book	The Great Gatsby	02/10/2025, 03:30 PM	Return Item
Book	The Soul of a New Machine	02/10/2025, 03:30 PM	Return Item



Bell Pepper
Green bell pepper.

The C Programming Language
Kernighan, Ritchie

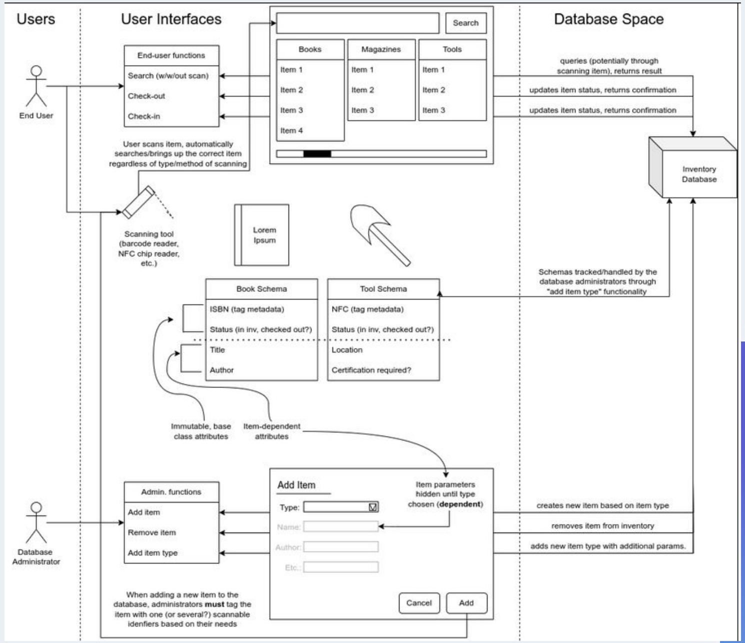
At Inauguration, Spending Money at a Record P...
The New York Times, no. #58,208

Broader Impacts

- Improved alternative to current open-source ILS (integrated library software) systems
    - Newer code and technologies equals less set-up hassle and more performance
- Type generality expands the system's use-cases beyond libraries
    - Makerspaces (tool rentals)
    - Archival offices (research management)
    - Legal firms (document lookup)
    - Healthcare institutions (educational material, equipment)
    - Storerooms (inventory)

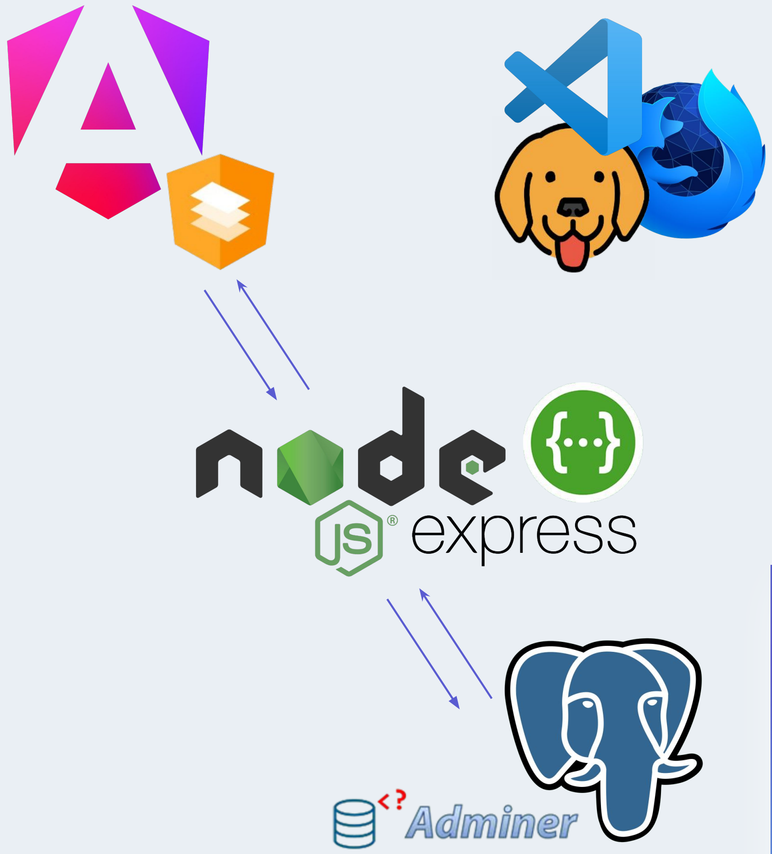
Design Specifications

- STIMS consists of three main programs:
  - Web application
    - Main program interface, accessible by administrators and users
    - Easy searching, check out, item/category creation
    - Interacts with the database via API calls
  - API middleware
    - Handles data requests to and from the database
    - Called by the web application to display information
  - Database
    - Securely stores user login information
    - Contains information regarding the main catalog, item types, check-out information, and more


Users
User Interfaces
Database Space
End User
End-user functions
Search (w/w/out scan)
Check-out
Check-in
Search
Books
Magazines
Tools
Item 1
Item 2
Item 3
Item 4
Item 1
Item 2
Item 3
Item 1
Item 2
Item 3
queries (potentially through scanning item), returns result
updates item status, returns confirmation
updates item status, returns confirmation
Inventory Database
User scans item, automatically searches/brings up the correct item regardless of type/method of scanning
Scanning tool (barcode reader, NFC chip reader, etc.)
Lorem Ipsum
Schemas tracked/handled by the database administrators through "add item type" functionality
Book Schema
ISBN (tag metadata)
Status (in inv, checked out?)
Title
Author
Tool Schema
NFC (tag metadata)
Status (in inv, checked out?)
Location
Certification required?
Immutable, base class attributes
Item-dependent attributes
Database Administrator
Admin. functions
Add item
Remove item
Add item type
Add Item
Type:
Name:
Author:
Etc.:
Item parameters hidden until type chosen (**dependent**)
Cancel
Add
creates new item based on item type
removes item from inventory
adds new item type with additional params.
When adding a new item to the database, administrators **must** tag the item with one (or several?) scannable idenfiers based on their needs

Technologies

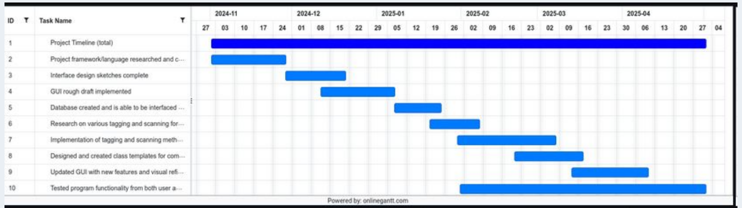
- Web application
  - Written in Angular (TypeScript)
    - Design implemented with Angular Material UI
  - Utilizes HttpClient request to talk to the API
  - Hosted locally on post 4200
- API middleware
  - Written in Node.JS with Express
  - Displays public API endpoints via Swagger web documentation (locally hosted under /docs)
  - Server runs on port 3000
- Database
  - Uses PostgreSQL for database software
  - Spun up in a Docker environment, all data saved to a Docker volume for easy backup and restoration
  - Adminer (database web-viewer tool) hosted on port 8080
  - PostgreSQL database hosted on port 5432 (default)
- Other dev tools: Visual Studio Code, Bruno, Firefox Dev Tools

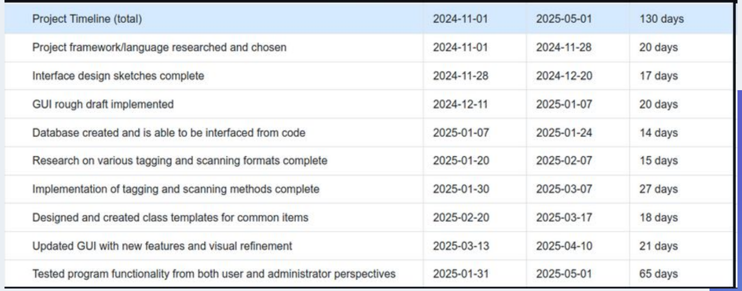


Milestones

- Nov 28th - Initial project research complete
- Dec 20th - Interface design sketches complete
- Jan 7th - Tagging system research complete
- Jan 26th - Major UI layout finalized and implemented
- Jan 29th - Core API functionality created
- Jan 31st - Database view & user dashboard functionality complete
- Feb 5th - Item checkout system complete
- Feb 7th - Search/filter functionality complete
- Feb 9th - Intake (add items) functionality complete \
- Feb 11th - Custom item type implementation functional
- Feb 15th (est.) - UI for custom type additions complete
- Feb 17th (est.) - All major features complete
- Feb 28th (est.) - STIMS project feature complete, begin testing phase

Original project timeline



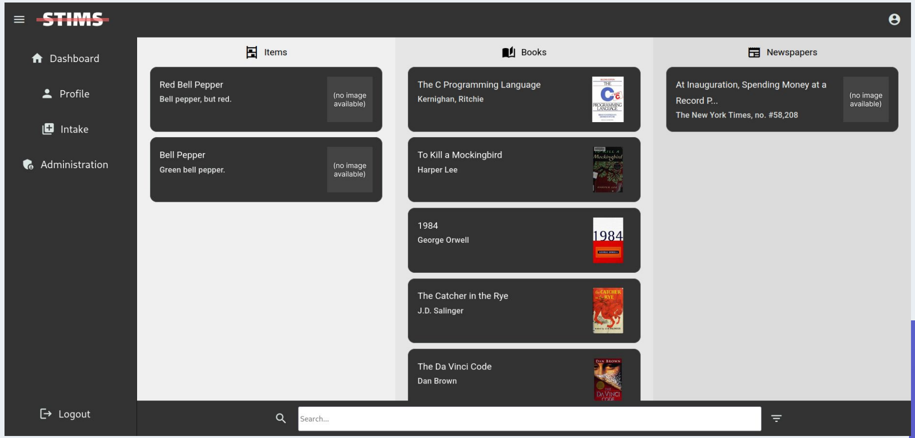


Project Timeline (total)	2024-11-01	2025-05-01	130 days
Project framework/language researched and chosen	2024-11-01	2024-11-28	20 days
Interface design sketches complete	2024-11-28	2024-12-20	17 days
GUI rough draft implemented	2024-12-11	2025-01-07	20 days
Database created and is able to be interfaced from code	2025-01-07	2025-01-24	14 days
Research on various tagging and scanning formats complete	2025-01-20	2025-02-07	15 days
Implementation of tagging and scanning methods complete	2025-01-30	2025-03-07	27 days
Designed and created class templates for common items	2025-02-20	2025-03-17	18 days
Updated GUI with new features and visual refinement	2025-03-13	2025-04-10	21 days
Tested program functionality from both user and administrator perspectives	2025-01-31	2025-05-01	65 days

Results

- As of February 11th, 2025, STIMS is estimated to be 90% feature complete
- Main three goals (usability, simplicity, and customizability) have been met through feature completion
- Plans to implement STIMS in a real-world, small scale library at a local non-profit makerspace in Cincinnati

- Remaining tasks
  - All milestones with an estimate time (UI for custom type implementations, final major features completion)
  - System testing and security self-audits
  - Demonstration set-up and practice for the CEAS Expo



Items

Red Bell Pepper
Bell pepper, but red.
(no image available)

Bell Pepper
Green bell pepper.
(no image available)

Books

The C Programming Language
Kernighan, Ritchie

To Kill a Mockingbird
Harper Lee

1984
George Orwell

The Catcher in the Rye
J.D. Salinger

The Da Vinci Code
Dan Brown

Newspapers

At Inauguration, Spending Money at a Record P...
The New York Times, no. #58,208
(no image available)

Challenges

- Longest software project that I’ve personally worked on
    - Solo project meant I was responsible for the entire workload of the capstone
    - Multi-month endeavor including planning, design, and testing
- Full tech stack
    - The STIMS project was the first time that I’ve set up a full tech stack (web front end, API middleware, and database backend) for a project
    - While getting everything to link together correctly was difficult, my experience in working on each part individually for similar projects/through co-op experiences helped me connect services properly