Get Summary Holdings-System Design & Implementation

Understanding requirements.

- Clients are library staff who are auditing the books currently held at the library.
- The primary purpose of the audit is to understand the missing books and then request them from external sources.
- Equipment available for use is limited to computers and tablets.
- The screen should show as much information as possible without cluttering.

Plausible user roles:

User Role	Interactions
Library Staff	Select missing and available editions. Copy data in a standardized format.

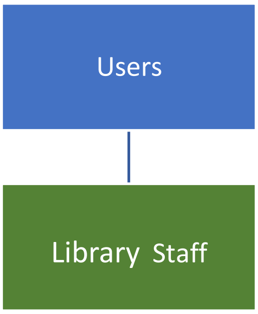

Figure 1:User Roles

Users - Key points:

1. Library Staff could have different levels of knowledge, it could range from newly appointed student workers to veterans.
2. Presently this work is being done with pen-and-paper, conveying this method to others is involving.
3. The learning curve of the software must be nearly flat.
4. Users could be doing many records/entries at a time, so software must have scripts that reduce physical interactions with the interface.

Software Architecture

- Java Spring Boot project
- Open API
- Microservices-like architecture. Only the required server needs to be scaled.
- Supports CI/CD pipeline (Continuous Integration/ Continuous Deployment).

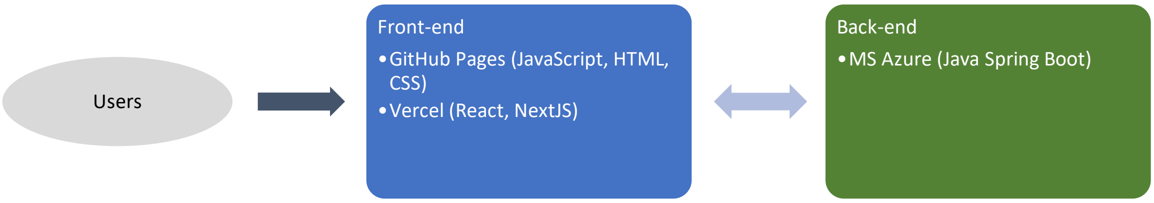
Users
Front-end
•GitHub Pages (JavaScript, HTML, CSS)
•Vercel (React, NextJS)
Back-end
•MS Azure (Java Spring Boot)

Figure 2: Software Architecture

Advantages:

- Less coupling. If the client decides to overhaul the application's UI in the future, the development work would be easier for the UI/UX team.
- Team members based on their expertise can split the work to produce high quality software.
- Supports different hosting services. Based on the budget and requirements, two different hosting services can be used.

Disadvantages:

- Deployment could be complex.

Generate Summary Holdings

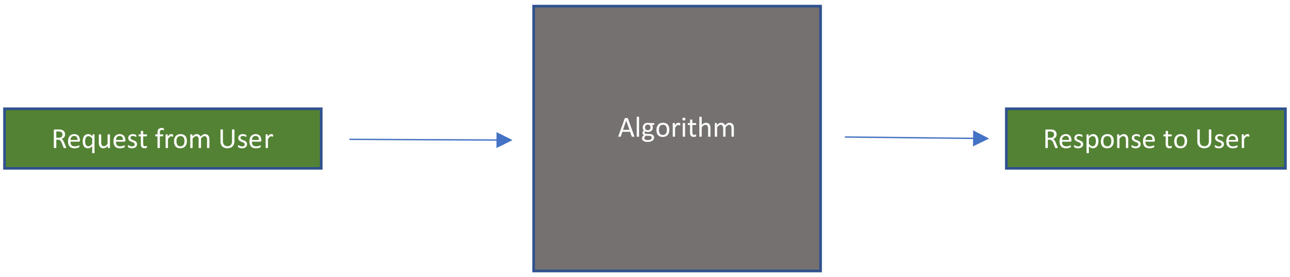

Figure 3:Generate Summary Holdings Process

Backend API Structure

- Algorithm process the request and delivers response.
- JSON Request to server
    - All fields are arrays except 'Editions in a year'.
    - 'Editions in a year' is single integer.
    - JSON Request Format
- JSON Response from server
    - All fields are strings.
    - JSON Response Format

GitHub Architecture Change.

- Currently source code is deployed on a GitHub personal account.
- GitHub personal accounts only allow 1 admin user.
- GitHub organization is a good alternative.

GitHub Organization

- Very similar to a GitHub personal account.
- Multiple users/collaborators can be assigned as admins.
    - This can help individual users configure GitHub pages and GitHub Actions to configure CI/CD pipelines.
- Supports multiple repositories (repo).
    - Repo for front-end and back-end can be stored under same organization.
    - GitHub Actions can be configured to run when an event happens on individual repo - segregation of both frontend and backend teams.

https://github.com/getSummaryholdings
I have created GitHub organization in the following organization to facilitate the transition.

Deployment

- Library computing devices generally do not give administrative privileges to users. In that case, deploying a desktop application is a complex process.
- Alternative is to deploy on an online cloud service and make it accessible to all.
- Front-end and back-end must be split.
- This could aid in better UX. Some cloud services have a long cold-start time in their cheaper plans. By deploying front-end in a high-performance machine and back-end in lower performance, during cold-start users can be notified about this.

Deployment Architecture

- The software has 2 user-interfaces (Lite and Updated).
- The backend is deployed on MS Azure webapp for containers as a Docker image.
- Frontend is deployed on Vercel.

URLs:

Lite Page	https://getsummaryholdings.com/lite
Backend	https://editiontracker.azurewebsites.net/
NextJS UI	https://getsummaryholdings.com/

JSON Request Format

{

"arrayEditionDescription": Edition description for each volume (e.g. vol, v, n.s. v).

"arrayEditionNumber": Edition number of each volume.

"arrayYear": Year of each volume.

"arrayAvailabilityStatusYear": Flag denoting availability status of the entire volume (0: All missing, 1: All found, 2: some missing).

"arrayAvailabilityStatusIssuesOfEachYear": Array-of-Arrays, here each individual array represents a single volume, the sub-array represents availability of issues in that volume. Availability represented by 1: found, 0: missing.

"arrayIssuesInTheYear": Specifies number of issues in the volume.

"editionsPerYear": The default number of editions per year.

}

JSON Response Format

{

"textAreaAvailableEditionsWithYear": Description of issues found with year.

"textAreaAvailableEditionsWithoutYear": Description of issues found without year.

"textAreaUnavailableEditionsWithoutYear": Description of issues missing with year.

"textAreaUnavailableEditionsWithYear": Description of issues found without year.

"textAreaAvailableSummaryHolding": Summary of the available issues in the format “edition_type(year)-edition_type(year)”.

}

Note:

1. The Azure server cold-start time could take up to 7 minutes. Please reload the page if it is not loaded.

Trivia:

- ‘Edition Tracker’ was the name that I had given to this application at first, but then I renamed it to ‘Get Summary Holdings’ because I felt the word ‘tracker’ could convey that the application could track books.
- I have been working in the university library as a Student Worker for 1 year, I got inspiration to make a solution for this complicated process after working on this task manually for a month.
- Initially I was planning to deploy using JavaScript, but I felt using data-structures in Java would be easier to do.
- I learned Java Spring Boot for this project, and it was well worth it.

Disclaimer

- I have tried my best to ensure that the calculations done by the software are accurate, however, there could be anomalies that I missed.
- I would always suggest double-checking the calculations to be on the safe side.