Household Inventory Management System

Isaac Tucker

CST-452 Capstone Project Final Form Document

Grand Canyon University

Instructor: Michael Landreth

Date: 05/15/2022

# Final Plan

Household Inventory Management System

Prepared by Isaac Tucker

|  |
| --- |
| Design Planning Summary |

I am a homeowner and often has difficulty keeping track of the items within his household; there have been times when multiples of the same item are obtained/purchased, and money is wasted as the fact that those items are already in one’s possession is overlooked. For that purpose, I propose a household inventory management system where a user can list households and keep a running inventory of items for each household. The user might then share their household inventory listing with others who live at the household for collaborative contribution to the inventory list. Moreover, in an effort to help reduce clutter, the inventory listing will have features to indicate items needing to be donated and provide a list of items for donation for public review.

This project will be a cloud deployed application for household inventory management. Users should be able to register, log in, create/read/update/destroy household listing and add/read/edit/delete items within each respective household. Households can be shared among users for collaborative inventory management. Items can be flagged for donation.

|  |
| --- |
| Overview of Design Concepts |

On a high level to achieve the intended project, Household Inventory Management System, we will make use of a database management system for data persistence. Information to and from this database will be communicated through a back-end API REST service. Then a front-end will provide an interface for users to make use of the REST service and make use of the database. These three components will make up the application itself. The needed application might look something like the following diagram.Diagram

Description automatically generated

The intention of the REST service will be to decouple the logic for the database interactions from the view itself. This will allow for easier implementation of other GUI front-ends if the application grows onto other platforms as it will provide a centralized means of communicating with stored data.

Additional UI mockups and other design deliverables are included below. Please see the table of contents for more.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Deliverable Acceptance Log | | | | | |
| ID | Deliverable Description | Comments | Evaluator (internal or external as applicable) | Status | Date of Decision |
| 1 | NodeJS/Express Back-End REST API | Hosted on Heroku | Isaac Tucker | Approved | 04/14/22 |
| 2 | ReactJS Front End UI | Hosted on Heroku | Isaac Tucker | Approved | 04/14/22 |
| 3 | MySQL DB | Host on Heroku’s JawsDB Dyno | Isaac Tucker | Approved | 04/14/22 |
| 4 |  |  |  |  |  |

|  |
| --- |
| Detailed Solution Architecture |

The Majority of this content will be expanded upon with the additional information included below the Table of Contents.

In essence, the above design will address the business need by providing an online, easily accessible, logging system for household systems. This way users can access the log from their mobile devices or computers at their convenience. The point of disconnecting the DB connection logic into a REST service will allow for future expandability of UI front-ends should a native phone app be developed; the current iteration only intends for a Web UI at this time, however.

Intended user workflow flowcharts and other system specific diagrams are included below for further development detail.

For security purposes, the functions of the application will require authentication. Moreover, the API end point will also require authentication for the calls to be made to load the data into the application. In this way will the application seek to protect the data within the application.

|  |
| --- |
| Hardware and Software Technologies |
| 1 -MySQL Database Management System |
| 2 -ReactJS Front End UI Framework |
| 3 -NodeJS/Express back-end RESTful API |
| 4 -Bootstrap Front End reactive display CSS library |
| 5 -Axios HTTP request library |
| 6-Heroku Online Cloud Web Hosting Platform |
| 7-Heroku JawsDB Dyno, Online Database Hosting solution. |

|  |
| --- |
| Revision and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Editor** | **Revision Notes** |
| 04.14.22 | Isaac Tucker | Initial draft for review/discussion |
| 05.13.22 | Isaac Tucker | Draft revisions. |

|  |
| --- |
| **Overall Instructor Feedback/Comments**  No feedback received. |

**Table of Contents**

[Final Plan 2](#_Toc103359004)

[Functional Requirements 6](#_Toc103359005)

[Epic: 6](#_Toc103359006)

[User Stories: 6](#_Toc103359007)

[Non-Functional Requirements 8](#_Toc103359008)

[User Stories 8](#_Toc103359009)

[Technical Requirements 9](#_Toc103359010)

[Requirements 9](#_Toc103359011)

[Tools and Purpose 9](#_Toc103359012)

[Logical System Design 10](#_Toc103359013)

[Workflow Flowchart: 11](#_Toc103359014)

[REST Service Design: 12](#_Toc103359015)

[User: 12](#_Toc103359016)

[POST: Create User 12](#_Toc103359017)

[GET: Get User By ID 12](#_Toc103359018)

[GET: Get Users 12](#_Toc103359019)

[GET: Get User By Email 12](#_Toc103359020)

[PUT: Update user 12](#_Toc103359021)

[DELETE: Delete user by ID 12](#_Toc103359022)

[POST: Authenticate 12](#_Toc103359023)

[Household Users: 12](#_Toc103359024)

[GET: Get HouseholdUsers 12](#_Toc103359025)

[GET: Get Users by Household ID 12](#_Toc103359026)

[GET: Get Households by User ID 12](#_Toc103359027)

[POST: Create Household User 12](#_Toc103359028)

[DELETE: Delete household users by ID 12](#_Toc103359029)

[Households: 12](#_Toc103359030)

[POST: Create Household 13](#_Toc103359031)

[GET: Get All Households 13](#_Toc103359032)

[GET: Get Household By ID 13](#_Toc103359033)

[GET: Get Household By UserID 13](#_Toc103359034)

[DEL: Delete Household by ID 13](#_Toc103359035)

[PUT: Update Household 13](#_Toc103359036)

[Items: 13](#_Toc103359037)

[POST: Create Item 13](#_Toc103359038)

[GET: Get Items 14](#_Toc103359039)

[GET: Get Item By ID 14](#_Toc103359040)

[GET: Get Items By Household ID 14](#_Toc103359041)

[PUT: Update Item 14](#_Toc103359042)

[DEL: Delete Item 14](#_Toc103359043)

[Reports: 14](#_Toc103359044)

[GET: Items for Donation Summary 14](#_Toc103359045)

[Expected Data Objects (UML) 15](#_Toc103359046)

[Database ER Diagram: 15](#_Toc103359047)

[Some Intended Code Implementation(s): 16](#_Toc103359048)

[Data Transfer Object Expectation: 16](#_Toc103359049)

[REST Calls (for expected paths): 17](#_Toc103359050)

[Axios Implementation: 17](#_Toc103359051)

[Front-End GetItems Intended Code: 18](#_Toc103359052)

[Some Initial Intended Front-End Navigation: 19](#_Toc103359053)

[User Interface Design 20](#_Toc103359054)

[Sitemap: 20](#_Toc103359055)

[Wireframes: 21](#_Toc103359056)

[Login: 21](#_Toc103359057)

[Register: 22](#_Toc103359058)

[Landing Page: 23](#_Toc103359059)

[Households: 24](#_Toc103359060)

[New Household: 25](#_Toc103359061)

[Selected Household, Items not toggled: 26](#_Toc103359062)

[Selected Household, Items toggled: 27](#_Toc103359063)

[Delete Household/Item Prompt: 27](#_Toc103359064)

[Edit Household: 28](#_Toc103359065)

[New Item: 29](#_Toc103359066)

[Edit Item: 30](#_Toc103359067)

[About: 30](#_Toc103359068)

[Reports Design Items for Donation Summary: 31](#_Toc103359069)

# Functional Requirements

## Epic:

As a user I want a system where I can track my various items that I have within my household(s).

## User Stories:

* Register: As a user I want to be able to register with the application so I can use the application and have a unique account to associate my household entries with.
  + Store user information in a relational database to be used for authentication purposes and to associate affiliated household instances.
* Login: As a user I want to be able to log into my account to access my households.
  + Authenticate provided user credentials against those stored within the relational database to allow access to application features and associated household instances.
* Add household: As a user I want to be able to make different household listings so I can separate items based off different locations.
  + Store household instance data in a relational database along with associated user id for future reference.
* Delete household: As a user I want to be able to delete households in case I no longer need the inventory list.
  + Delete household instance data from relational database if associated with active user.
* Update household inventory: As a user I want to be able to update my household information in case I made a mistake or details change.
  + Update household instance data within relational database if associated with active user.
* View households owned/managed by user: As a user I want to be able to view the households that I am managing.
  + Retrieve all household instance data for any given user id associated with active user and display those households within the application UI.
* View household items for households owned/managed by user: As a user I want to be able to view the inventory items that are currently in a select household.
  + Retrieve all items associated with a given household id associated with a given active user and display those items within the application UI.
* Add Items to household: As a user I want to be able to add items to a household listing.
  + Store provided item instance data in the relational database and associate new data with the selected household id for the associated active user.
* Delete Items from household: As a user I want to be able to delete items from a household listing.
  + Delete selected item instance data from the relational database for the selected household associated with the active user.
* Update Items in household: As a user I want to be able to update item information within a household.
  + Update provided item instance data for a given item id in the relational database for the associated active user.
* Flag for donation: As a user I want to be able to flag items for donation.
  + Set the donation flag to true for the given item id in the relational database. This will then be indicated on the application UI.
* Invite Collaborators: As a user I want to be able to invite other users to also add to my households.
  + Provide a text field to add contributors to a selected household instance associated with a given active user. If the intended contributor is a user within the system then associate that selected household id with the intended contributor account. The contributor will now have that household displayed within their application UI for CRUD privileges.

# Non-Functional Requirements

## User Stories

* As a user I want to be able to run the application on popular web browsers (Chrome, Safari, Firefox)
* As a user I want to the application UI to be responsive to different resolutions (computer, phone).
* As a user I want the application interface to be simple and to be in English, so it is easy to understand.
* As a user I want the application UI menu bars to be minimal, so I do not feel overwhelmed.
* As a user I want the application UI to be high contrast, so it doesn’t hurt my eyes in extended use.
* As a user I do not want to have to pay to utilize this application so I can save my money for the new items to be recorded in the application.
* As a user I want the application page load times to be under 5 seconds so I am not wasting time waiting for items to load.
* As a user I want the application to be available at least 90% of the time, if not 99% of the time, so I can add items at any time of day.
* As a user I do not want the application to contain any vulgarities so my children can help with data entry without putting them at risk.
* As a user I want the application to auto sort my item listings alphabetically to make finding items easier.
* As a user I want the web application to work with screen readers in case I cannot see.

# Technical Requirements

## Requirements

* The Application should be deployed online.
* The application should make use of online data persistence
* The application should use an online back-end rest service
* The application should have responsive UI elements
* The application should be available in popular web browsers.

## Tools and Purpose

* Heroku Dyno JawsDB:
  + DB server to host MySQL Relational Database for data persistence.
* Heroku Cloud Deployment:
  + Server to host back-end rest NodeJS/ExpressJS service.
* NodeJS/ExpressJS REST Service:
  + Service to Get/Post/Put/Delete JSON data to MySQL Relational Database.
* Heroku Cloud Deployment:
  + Server to host the application UI front-end
* ReactJS:
  + Front-End framework to render the UI elements and make calls to the back-end REST service.
* Bootstrap V5:
  + Front-End library to assist with the UI responsive elements.
* Axios:
  + Library to be used within ReactJS to make calls to the back-end rest service.
* Chrome:
  + Web Browser intended for main use of application features

# Logical System Design

Diagram

Description automatically generated

# Workflow Flowchart:

**Diagram

Description automatically generated**

# REST Service Design:

## User:

### POST: Create User

{

“first\_name”:”Jon”,

“last\_name”:”Smith”,

“email”:”jon@me.com”,

“password”:”aysdgajsbaisuyfa76”

}

### GET: Get User By ID

### GET: Get Users

### GET: Get User By Email

### PUT: Update user

{

"id":"2",

"first\_name": "Tesasdasdt",

"last\_name": "Tester",

"email": "tester@metoo.com",

"password": "aaaaaaaaaa"

}

### DELETE: Delete user by ID

### POST: Authenticate

{

"email":"tester@me.com",

"password":"aaaaaaaaaa"

}

## Household Users:

### GET: Get HouseholdUsers

### GET: Get Users by Household ID

### GET: Get Households by User ID

### POST: Create Household User

{

“user\_id”: “3”,

“household\_id”: “3”

}

### DELETE: Delete household users by ID

### 

## Households:

### POST: Create Household

{

"name": "Milestone 2 Video House",

"street": "123 Fake Street”,

"city": "Springfield",

"state": "Illinois",

"zip": "85053",

"description": "Some Home"

}

### GET: Get All Households

### GET: Get Household By ID

### GET: Get Household By UserID

### DEL: Delete Household by ID

### PUT: Update Household

{

"id": "2",

"name": "Milestone 2 Video House",

"street": "123 Fake Street”,

"city": "Springfield",

"state": "Illinois",

"zip": "85053",

"description": "Some Home"

}

## Items:

### POST: Create Item

{

"name": "Cat",

"description": "Kitty The Cat",

"quantity": "1",

"households\_id": "2",

"donation\_flag": "N"

}

### GET: Get Items

### GET: Get Item By ID

{

“id”: “3”

}

### GET: Get Items By Household ID

{

“household\_id”: “3”

}

### PUT: Update Item

{

"id": "2"

"name": "Cat",

"description": "Kitty The Cat",

"quantity": "1",

"households\_id": "2",

"donation\_flag": "Y"

}

### DEL: Delete Item

{

“id”: “3”

}

## Reports:

### GET: Items for Donation Summary

# Expected Data Objects (UML)

Table

Description automatically generated

# Database ER Diagram:

DBMS: I expect that the DB will be deployed on Heroku using the JawsDB MySQL hosting. An initial ERD of the DB may look like…

Diagram

Description automatically generated

The main intention here with this database structure is that one household will have many items. Then there is an intermediary table to hold user and household associations. This way households can be shared with other users and stored within this table to allow collaboration with household item data entry. The expectation is that, upon household creation, a household\_users entry will also be created to affiliate the new household ID with the creating user ID. Then if additional users are invited to collaborate, additional user/household pairings will be made.

# Some Intended Code Implementation(s):

## Data Transfer Object Expectation:

Text

Description automatically generated

## REST Calls (for expected paths):

Graphical user interface, text, application

Description automatically generated

## Axios Implementation:

Text

Description automatically generated

## Front-End GetItems Intended Code:

Text

Description automatically generated

## Some Initial Intended Front-End Navigation:

Text

Description automatically generated with low confidence

# User Interface Design

## Sitemap:

Diagram

Description automatically generated

## Wireframes:

### Login:

Graphical user interface, application

Description automatically generated

### Register:

Graphical user interface

Description automatically generated

### Landing Page:

Graphical user interface, application, Teams

Description automatically generated

### Households:

Graphical user interface, application

Description automatically generated

### New Household:

Graphical user interface

Description automatically generated

### Selected Household, Items not toggled:

Graphical user interface, text, application, email

Description automatically generated

### Selected Household, Items toggled:

Table

Description automatically generated

### Delete Household/Item Prompt:

Table

Description automatically generated with medium confidence

### Edit Household:

Graphical user interface, application

Description automatically generated

### New Item:

Graphical user interface

Description automatically generated

### Edit Item:

Graphical user interface, application

Description automatically generated

### About:

Table

Description automatically generated with medium confidence

# Reports Design Items for Donation Summary:

The report would run to populate the table on the landing page. This way users can see what items are listed for donation and can reach out to the email address listed if they are in need. This report can also be shared with local donation centers for the same purpose.

This report would run at the start of the application (login) and be stored in the local cache for 5-10 minutes. After expiration, it will run again to update the landing page table.

|  |  |  |
| --- | --- | --- |
| Items for Donation Summary | | |
| Display Value | Variable or Field | Format Notes |
| Item Name | items.name | “Item 1” |
| Item Quantity | items.quantity | “3” |
| Item Description | items.description | “It’s an old chair” |
| Household Zip | households.zip | “85053” |
| User Email | users.email | “jon@smith.com” |
| User First Name | users.first\_name | “Jon” |