Household Inventory Management System

Isaac Tucker

CST-452 Capstone Project Requirements Document

Grand Canyon University

Instructor: Michael Landreth

Date: 05/15/2022

**ABSTRACT**

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.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 04/03/2022 | Isaac Tucker | Initial draft for review/discussion |
| 05/13/2022 | Isaac Tucker | Revision for final submission. |
|  |  |  |

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

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

**Integrated Instructor Feedback into Project Documentation**

Yes  No, no feedback was provided.

**Project Approval**

Professor Amr Elchouemi

**Table of Contents**

[Functional Requirements 5](#_Toc103357959)

[Epic: 5](#_Toc103357960)

[User Stories: 5](#_Toc103357961)

[Non-Functional Requirements 7](#_Toc103357962)

[User Stories 7](#_Toc103357963)

[Technical Requirements 8](#_Toc103357964)

[Requirements 8](#_Toc103357965)

[Tools and Purpose 8](#_Toc103357966)

[Logical System Design 9](#_Toc103357967)

[REST Service Design: 9](#_Toc103357968)

[User: 9](#_Toc103357969)

[POST: Create User 9](#_Toc103357970)

[GET: Get User By ID 9](#_Toc103357971)

[GET: Get Users 9](#_Toc103357972)

[GET: Get User By Email 9](#_Toc103357973)

[PUT: Update user 9](#_Toc103357974)

[DELETE: Delete user by ID 10](#_Toc103357975)

[POST: Authenticate 10](#_Toc103357976)

[Household Users: 10](#_Toc103357977)

[GET: Get HouseholdUsers 10](#_Toc103357978)

[GET: Get Users by Household ID 10](#_Toc103357979)

[GET: Get Households by User ID 10](#_Toc103357980)

[POST: Create Household User 10](#_Toc103357981)

[DELETE: Delete household users by ID 10](#_Toc103357982)

[Households: 10](#_Toc103357983)

[POST: Create Household 10](#_Toc103357984)

[GET: Get All Households 11](#_Toc103357985)

[GET: Get Household By ID 11](#_Toc103357986)

[GET: Get Household By UserID 11](#_Toc103357987)

[DEL: Delete Household by ID 11](#_Toc103357988)

[PUT: Update Household 11](#_Toc103357989)

[Items: 11](#_Toc103357990)

[POST: Create Item 11](#_Toc103357991)

[GET: Get Items 11](#_Toc103357992)

[GET: Get Item By ID 11](#_Toc103357993)

[GET: Get Items By Household ID 11](#_Toc103357994)

[PUT: Update Item 12](#_Toc103357995)

[DEL: Delete Item 12](#_Toc103357996)

[Reports: 12](#_Toc103357997)

[GET: Items for Donation Summary 12](#_Toc103357998)

[User Interface Design 13](#_Toc103357999)

[Sitemap: 13](#_Toc103358000)

[Wireframes: 14](#_Toc103358001)

[Login: 14](#_Toc103358002)

[Register: 15](#_Toc103358003)

[Landing Page: 16](#_Toc103358004)

[Households: 17](#_Toc103358005)

[New Household: 18](#_Toc103358006)

[Selected Household, Items not toggled: 19](#_Toc103358007)

[Selected Household, Items toggled: 20](#_Toc103358008)

[Delete Household/Item Prompt: 20](#_Toc103358009)

[Edit Household: 21](#_Toc103358010)

[New Item: 22](#_Toc103358011)

[Edit Item: 23](#_Toc103358012)

[About: 23](#_Toc103358013)

[Reports Design 24](#_Toc103358014)

[Household Summary: 24](#_Toc103358015)

[Items for Donation Summary: 24](#_Toc103358016)

# 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

# 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

# User Interface Design

## Sitemap:

Diagram

Description automatically generated

## Wireframes:

### Login:

Graphical user interface, application

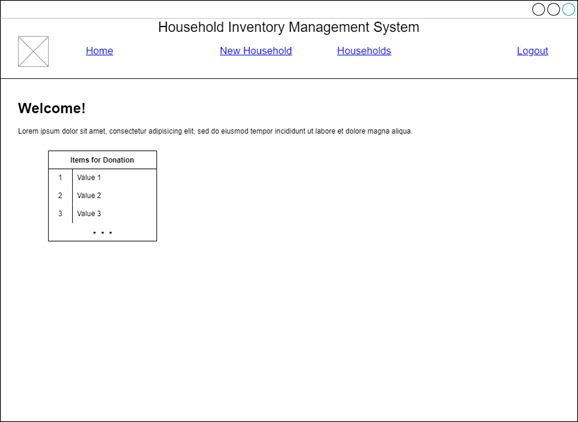
Description automatically generated

### Register:

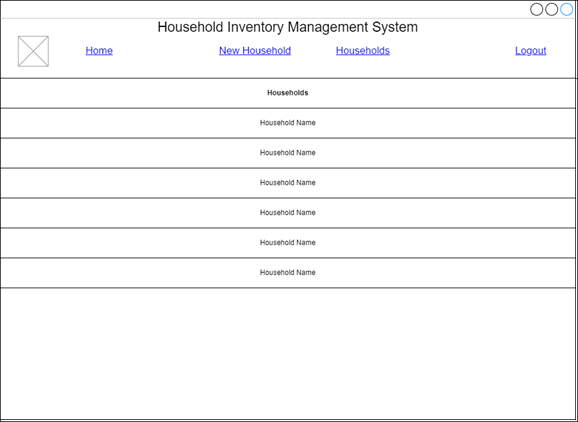
Graphical user interface

Description automatically generated

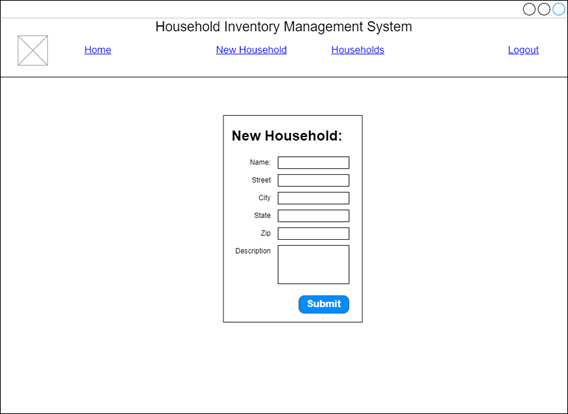
### Landing Page:



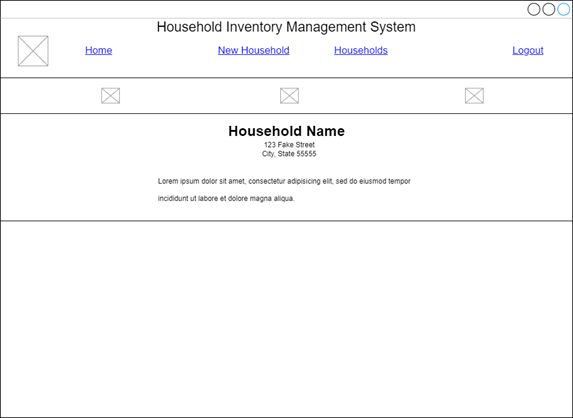
### Households:



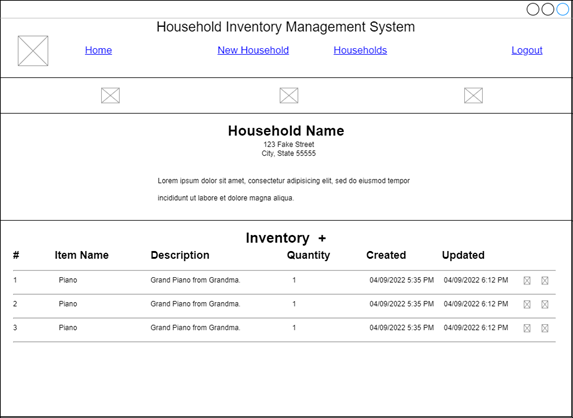
### New Household:



### Selected Household, Items not toggled:



### Selected Household, Items toggled:

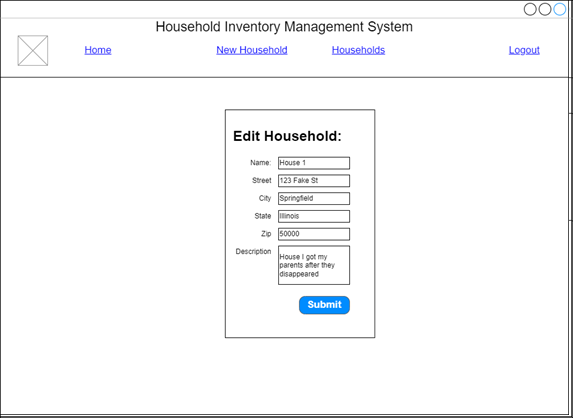


### Delete Household/Item Prompt:

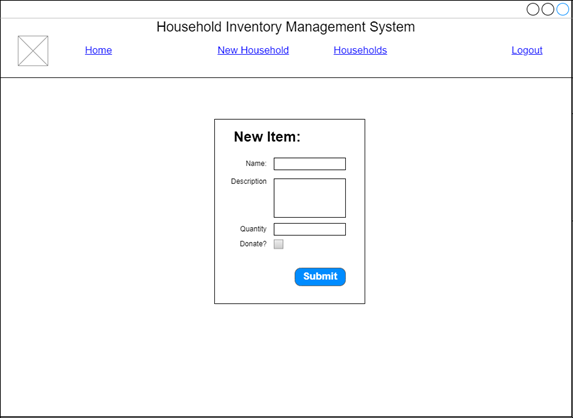
Table

Description automatically generated with medium confidence

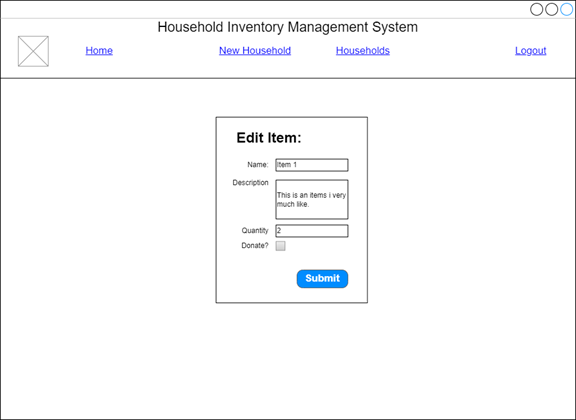
### Edit Household:



### New Item:



### Edit Item:



### 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 ID | Item.id | “1” |
| Item Name | items.name | “Item 1” |
| Item Description | items.description | “It’s an old chair” |
| Item Quantity | items.quantity | “3” |
| Item Updated At | Item.updated\_at | “2022-05-27” |
| Household State | households.state | “AZ” |
| User Email | users.email | “jon@smith.com” |