The Homeless Helper

Team Thunder

Group 10

Zachary Schirm

Thor Johansson

Benjamin Weiss

Kathleen Tiley

GitHub: https://github.com/The-Zen-Cat/CEN3031_Group10_Project

Table of Contents

SECTION 1 – Project Description, Solution, Features and Functionality

Challenge Statement and Solution

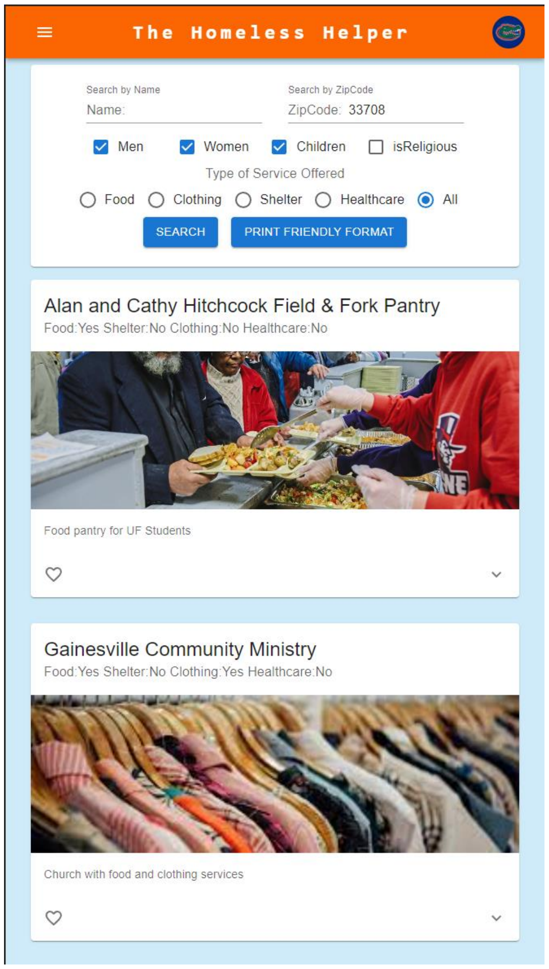
Many homeless individuals are transient and unfamiliar with local resources available to them. They cannot easily discover information about these resources in order to take advantage of them. While hospital emergency departments receive a large number of homeless patients every day, staff are often not able to provide information about these resources without the help of social workers, who are usually not available around the clock, and sometimes are not available at all. However, many homeless individuals do have phones with which they can access the Internet using Wi-Fi, although some do not.

To address this information gap, our team created a mobile compatible web application prototype that consolidates and displays information about local resources available to the homeless. The application allows homeless individuals to search and filter resources based on key factors of interest. It also provides a print functionality, allowing homeless individuals to print lists of resources from public computers. In addition, staff at hospitals and other locations regularly visited by the homeless can keep printed copies of local resources on hand for quick distribution.

Features and Functionality

Search and Filter

The search and filter functionality is available to all users regardless of account log in status. It displays information about resources stored in the database based on user-inputted parameters in an easy to read format. Users can choose to view resources that are age or gender specific, resources affiliated with religious organizations, or resources that provide a specific type of service. Users can also search for resources by zip code. Our application takes advantage of the Google Maps API to automatically detect the user’s location and provide resources within that zip code.


The Homeless Helper

Search by Name
Name:

Search by ZipCode
ZipCode: 33708

[✓] Men [✓] Women [✓] Children [ ] isReligious

Type of Service Offered

( ) Food ( ) Clothing ( ) Shelter ( ) Healthcare (●) All

SEARCH | PRINT FRIENDLY FORMAT

Alan and Cathy Hitchcock Field & Fork Pantry
Food:Yes Shelter:No Clothing:No Healthcare:No

Food pantry for UF Students

Gainesville Community Ministry
Food:Yes Shelter:No Clothing:Yes Healthcare:No

Church with food and clothing services

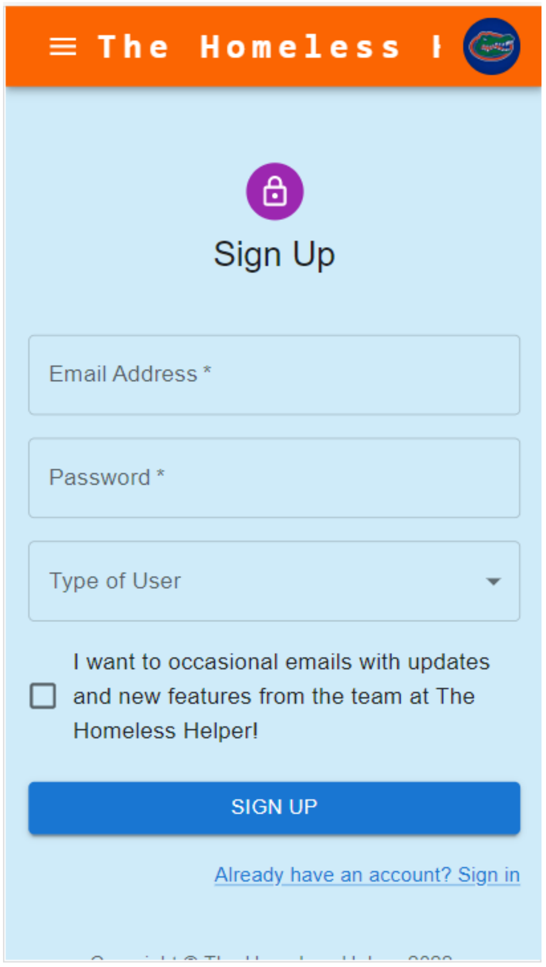
Print Function

The print functionality is also available to all users regardless of account log in status. Users can print the current filtered list of resources by pressing the easy-to-find print button in the interactive search box. After generating a list of resources, users can select certain resources as favorites and print only those resources instead of the full list. This functionality is provided regardless of account log in status, but favorites made by logged out users will not be saved for future sessions.

Map Display

The map is not implemented in the prototype version of our application. In the final version, we will use the Google API to embed a map on the home page displaying the pinned locations of nearby resources based on the user's location, which is automatically detected. The map will be filterable based on the same parameters currently provided, and will include clickable pins that bring up further details about each resource listed. The map display will allow users to quickly see the closest resources that meet their needs. When a resource is selected within the map, the application will provide an option allowing users to generate and print Google Maps directions to the location.

User Accounts and Information Management



I want to occasional emails with updates and new features from the team at The Homeless Helper!

The final version of our application will support accounts for three types of users. The first user group consists of homeless individuals. These individuals can create accounts to store lists of favorite resources and to write reviews of resources, a functionality described below. The Service Provider user group consists of staff members working at organizations providing services to the homeless. These users can create accounts to add and update information about their service. The Resource Coordinator user group consists of hospital-based social workers or other local representatives who volunteer to manage the information provided about services for the homeless in their area. These users will be able to collectively approve the accounts of new Resource Coordinators, independently approve accounts for service provider representatives in their area, and revoke account access as needed. The prototype version of our application implements account creation and functionality for Resource Coordinators only.

The final version of our application will also include a Dashboard for logged in users based on the user type. For homeless users, this will include lists of favorites and information about reviews the user has made about service providers. For Service Providers and Resource Coordinators, this will include information about the resource or resources they are responsible for, including account approval and information update workflows.

Reviews

In the final version of our application, homeless users will be able to leave reviews about their experience at a service provider location. The review can be made public or private. Only users who are logged in will be able to leave reviews, but all users will be able to view public reviews. Private reviews will only be visible to the user who left the review, the service provider representative, and the resource coordinator for that area. Service providers will be able to flag reviews that contain inappropriate or offensive content. Resource coordinators will review these flagged reviews and remove them if necessary. They will also be able to revoke the accounts of users who regularly leave problematic reviews. This review system will allow homeless individuals to get honest information about the quality of a service provider from other beneficiaries, empowering them to make more informed decisions about which services to use.

System Models

Our application employs a hybrid client-server and MVC (model-view-controller) model. As a web application, it includes a traditional server, database, and client view of the application via a web browser. Our application deviates from the traditional client-server architecture because we use React.JS to manage the user interface. React most closely aligns with the MVC architecture. Using an MVC model provides us with many benefits over a traditional client-server architecture. In particular, an MVC model makes the application easier to maintain and faster to develop. It also allows us to create multiple views for different groups of users, which our application requires.

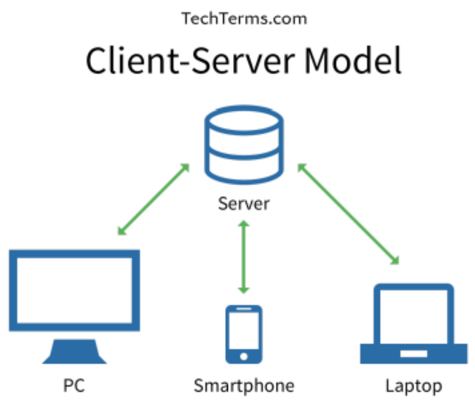


Figure 1: A Client-Server Architecture [1].

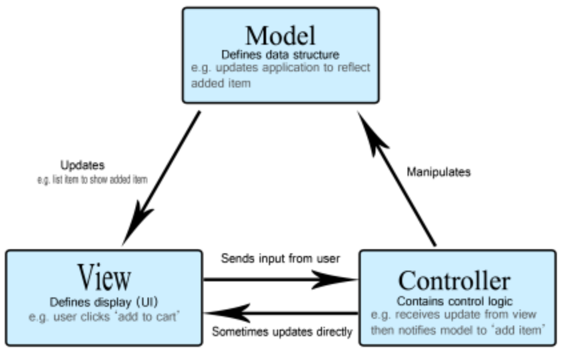
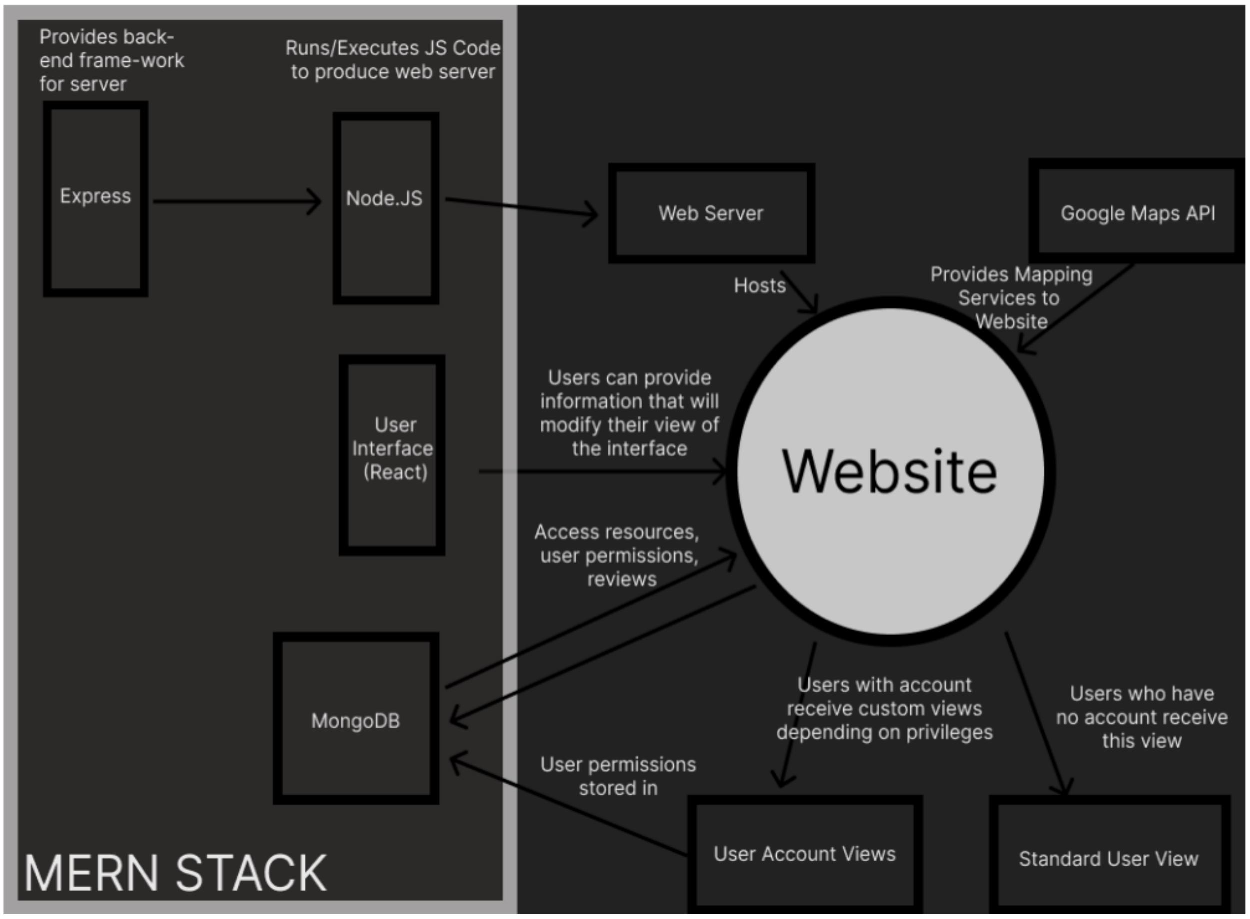

Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Figure 2: A Model-View-Controller Architecture [2].

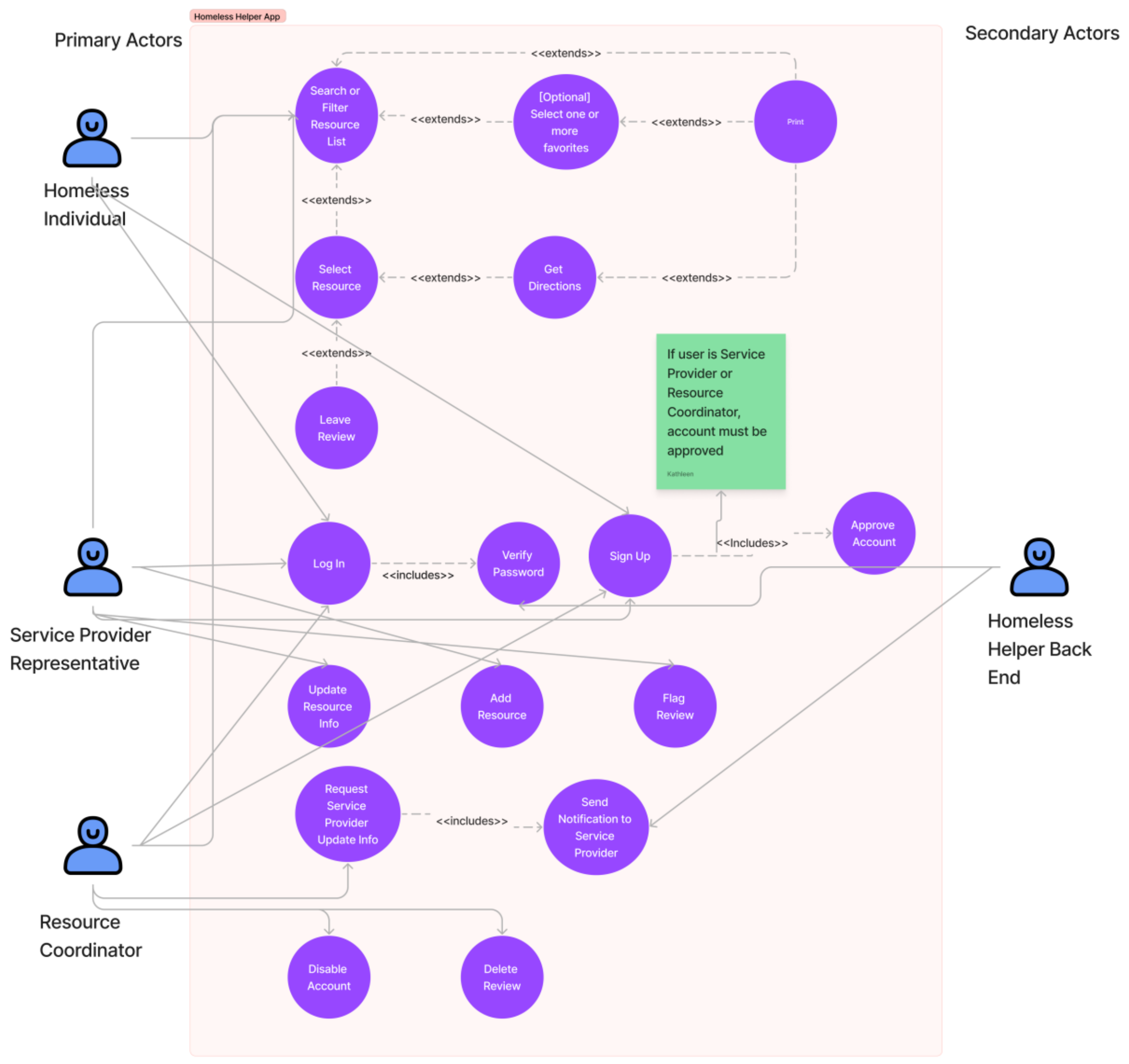
System Context Model

We built our application using the MERN stack, which consists of React.JS, Express,JS, Node,JS, and MongoDB. The server consists of Express.JS and Node.JS, and MongoDB is the database that will be deployed. The user interface is designed using React.JS. In our system context model, the web server hosts the Website. The website is responsible for serving both standard user views and user account views, which include views for social workers, individual resource providers, and homeless users. User account information is stored in MongoDB. The Website can access stored resources, user permissions, and reviews from MongoDB. Finally, the Website is reliant upon the Google Maps API to provide mapping services and directions. This API is central to the vision of the application as it allows users to view where resources are in relation to themselves and also view and print out turn by turn directions to get them to their needed resources.


Provides back-end frame-work for server
Runs/Executes JS Code to produce web server
Express
Node.JS
Web Server
Google Maps API
Hosts
Provides Mapping Services to Website
User Interface (React)
Users can provide information that will modify their view of the interface
Website
Access resources, user permissions, reviews
MongoDB
Users with account receive custom views depending on privileges
Users who have no account receive this view
User permissions stored in
User Account Views
Standard User View
MERN STACK

Use Case Model

Below is our Use Case Diagram outlining the different actions users can take within the system. Note that all types of users can search or filter the resource list. This allows service providers and other support staff to print out lists of nearby resources for Homeless Individuals who do not have Wi-Fi enabled mobile phones.



SECTION 2 – Code Management, Test Plan, Static Code Analysis and Report

Code Management

Code management has been performed using GitHub. As the most popular implementation of the Git version control tool with widespread support, and free public accounts for teams, this was an easy choice. A Team Repository was setup and configured with ESlint for static code analysis, codeQL to automate code security checks, and prettier for automated styling of code to keep team members' contributions uniform. All team members regularly pulled and pushed changes to individual dev branches, and when ready for review, pull requests were created for merging with the master branch. Github was configured such that review by at least one other team member was required prior to merging with the master branch. If code required changes, these could be requested by the reviewing teammate prior to reconsideration of the pull request.

Test Plan

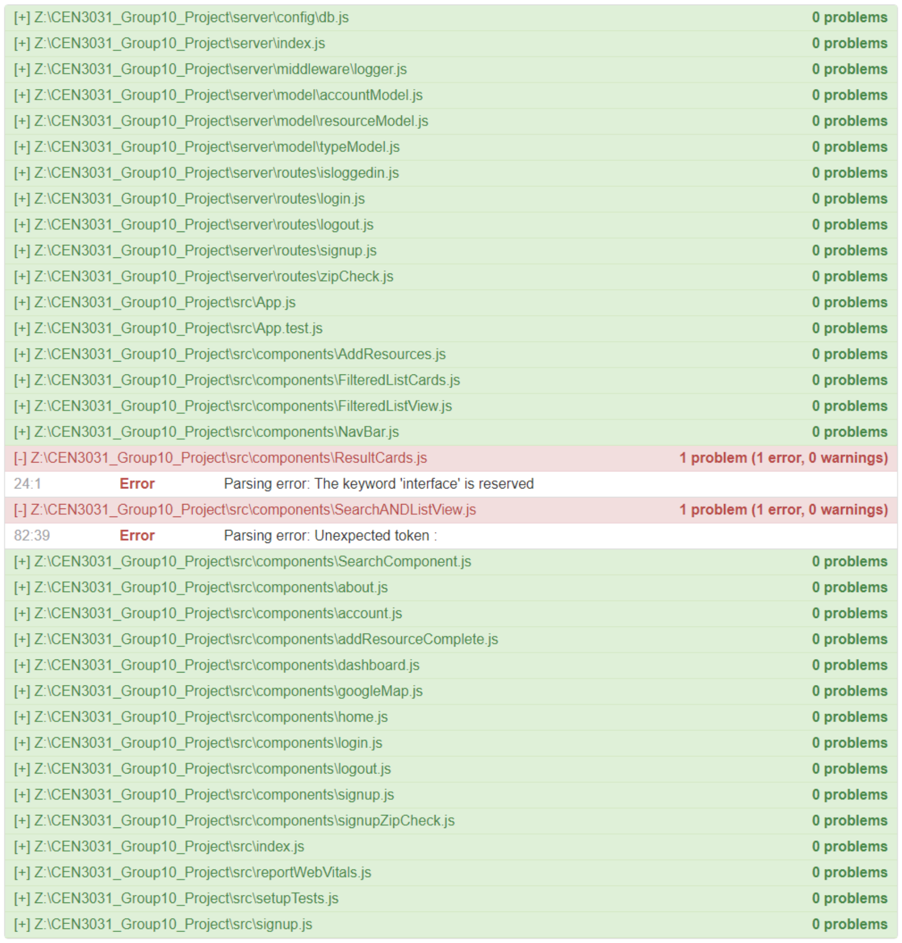
We employed test-driven development principles while designing and implementing our application. Before starting to code, we identified a set of test cases and expected output parameters. When each component of our application was implemented, we ran the tests to ensure the app was working as expected. Although we established a method of automated testing, we primarily used manual testing because our tests involved user input. Tests involving user interactions are difficult to implement, and we judged it would be less time-consuming to perform the tests manually, given the small scale of our web application. As our application expands and as new features are added, we will make greater use of the automated testing process we established.

TEST CASE ID	TEST CASE DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	STATUS	COMMENTS
tc_001_DBquery	Correct fetch from database	"this is the soupiest soup kitchen"	"this is the soupiest soup kitchen"	complete	
tc_002_AreaManagerLogin	Verify login for area manager	successful login	successful login	complete	
tc_003_IndSiteLogin	Verify login for independent provider of services	successful login	n/a	to be completed	feature not completely implemented for testing yet
tc_004_FilterResultsByType	Verify filtered results are only of type specified	Results are only of type "food"; all "food" category from database present in result	Results are only of type "food"; all "food" category from database present in result	complete	
tc_005_FilterResultsByLocation	Verify filtered results are only within a certain zip code, city, or neighborhood	List of service providers in Gainesville, FL	List of service providers in Gainesville, FL	complete	this feature was implemented for zip code only

tc_006_FilterResultsByRadius	Verify filtered results are within a given radius of a given location (an address or user current location)	List of service providers within 5 miles of Gainseville, FL center is generated	n/a	to be completed	feature not completely implemented for testing yet
tc_007_FilterResultsByGroupServed	Verify filtered results serve the specified group - male, female, children, young adult, families, etc	List of service providers who only serve female young adults	List of service providers who only serve female young adults	complete	
tc_008_FilterResultsByReligiousAffiliation	Verify filtered results are affiliated with the specified religion: No Affiliation, Catholic, Protestant, Muslim, Jewish	List of providers affiliated with the Catholic Church	n/a	in progress	feature is implemented and tested for general religious affiliation
tc_009_FilterResultsbyOpeningHours	Verify filtered providers are open at the specified times. Options: Choose open now or specify a day of the week and a time (can be all day)	List of service providers that are currently open	n/a	in progress	feature not completely implemented for testing yet
tc_010_UIButtonNavigation	Tests that UI buttons operate as expected upon user click	Log in page is loaded	Log in screen displays	complete	
tc_011_DBAuthenticationCheck	Tests to ensure authentication is working and required when attepting to add to the database	The server throws an error, and the resource is not added.	n/a	in progress	feature not completely implemented for testing yet
tc_012ResourceFormatCheck	Verify that resources are correctly formatted before adding it to the database	Error text displaying the incorrect field	n/a	in progress	feature not completely implemented for testing yet

tc_013SignUp	Verify that when users sign up user info is registered in the database	successful sign up	successful sign up	complete	
tc_014TrackingLogInStatus	Verify UI displays different options based on user log in status	Navbar options are 'Add Resource' and 'Log Out'	Navbar options are 'Add Resource' and 'Log Out'	complete	

Static Code Analysis and Report

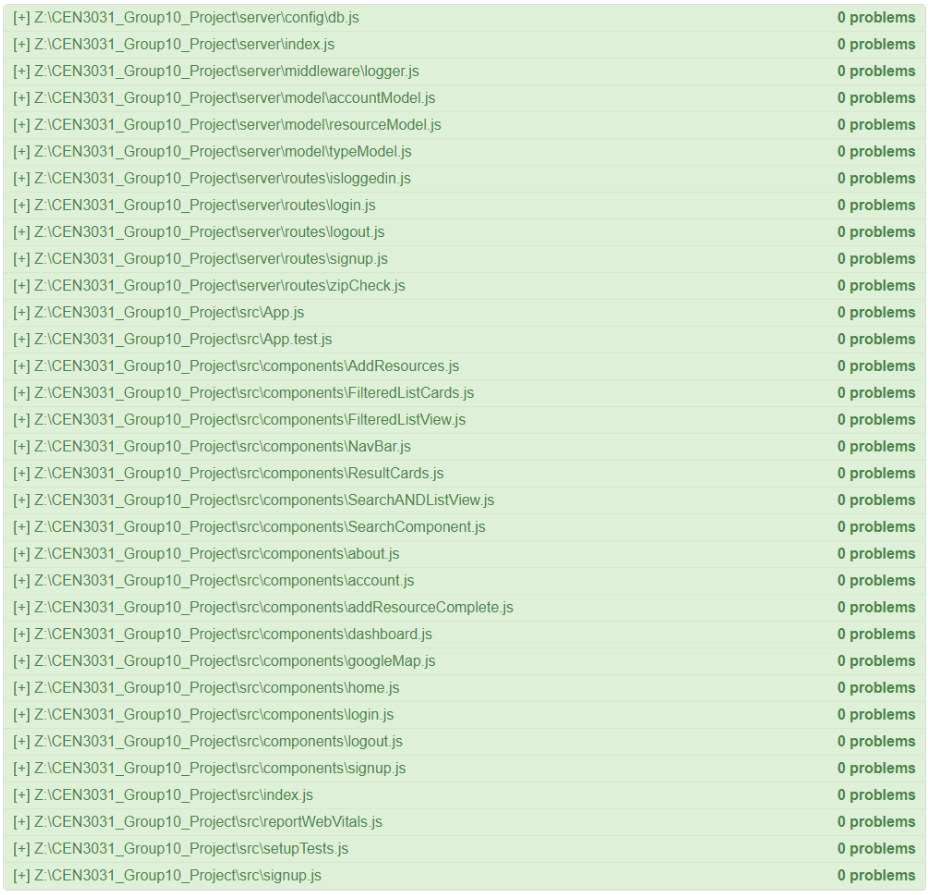

```
[+] Z:\CEN3031_Group10_Project\server\config\db.js                                  0 problems
[+] Z:\CEN3031_Group10_Project\server\index.js                                      0 problems
[+] Z:\CEN3031_Group10_Project\server\middleware\logger.js                          0 problems
[+] Z:\CEN3031_Group10_Project\server\model\accountModel.js                         0 problems
[+] Z:\CEN3031_Group10_Project\server\model\resourceModel.js                        0 problems
[+] Z:\CEN3031_Group10_Project\server\model\typeModel.js                            0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\isloggedin.js                          0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\login.js                               0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\logout.js                              0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\signup.js                              0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\zipCheck.js                            0 problems
[+] Z:\CEN3031_Group10_Project\src\App.js                                           0 problems
[+] Z:\CEN3031_Group10_Project\src\App.test.js                                      0 problems
[+] Z:\CEN3031_Group10_Project\src\components\AddResources.js                       0 problems
[+] Z:\CEN3031_Group10_Project\src\components\FilteredListCards.js                  0 problems
[+] Z:\CEN3031_Group10_Project\src\components\FilteredListView.js                   0 problems
[+] Z:\CEN3031_Group10_Project\src\components\NavBar.js                             0 problems
[-] Z:\CEN3031_Group10_Project\src\components\ResultCards.js       1 problem (1 error, 0 warnings)
24:1      Error      Parsing error: The keyword 'interface' is reserved
[-] Z:\CEN3031_Group10_Project\src\components\SearchANDListView.js 1 problem (1 error, 0 warnings)
82:39     Error      Parsing error: Unexpected token :
[+] Z:\CEN3031_Group10_Project\src\components\SearchComponent.js                    0 problems
[+] Z:\CEN3031_Group10_Project\src\components\about.js                              0 problems
[+] Z:\CEN3031_Group10_Project\src\components\account.js                            0 problems
[+] Z:\CEN3031_Group10_Project\src\components\addResourceComplete.js                0 problems
[+] Z:\CEN3031_Group10_Project\src\components\dashboard.js                          0 problems
[+] Z:\CEN3031_Group10_Project\src\components\googleMap.js                          0 problems
[+] Z:\CEN3031_Group10_Project\src\components\home.js                               0 problems
[+] Z:\CEN3031_Group10_Project\src\components\login.js                              0 problems
[+] Z:\CEN3031_Group10_Project\src\components\logout.js                             0 problems
[+] Z:\CEN3031_Group10_Project\src\components\signup.js                             0 problems
[+] Z:\CEN3031_Group10_Project\src\components\signupZipCheck.js                     0 problems
[+] Z:\CEN3031_Group10_Project\src\index.js                                         0 problems
[+] Z:\CEN3031_Group10_Project\src\reportWebVitals.js                               0 problems
[+] Z:\CEN3031_Group10_Project\src\setupTests.js                                    0 problems
[+] Z:\CEN3031_Group10_Project\src\signup.js                                        0 problems
```

ESlint was run against our code base throughout the entire project to ensure that any errors were quickly identified and corrected. Accordingly, the team had very few errors on our routine reports as these were generally corrected immediately at the time of initial coding of a page. Of note, we did receive two errors when incorporating some MUI (Material UI) components into our application.

The first error is “keyword ‘interface’ is reserved”. Upon researching this ‘error’ it was discovered that this is a “future reserved keyword” in JavaScript. However, the TypeScript compiler removes this definition such that it does not actually appear in the compiled JavaScript. Therefore, this is not truly an error, but rather a ‘false positive’ on the part of ESLint and can be fixed by removing ts type definitions with ESlint.

The second ESlint error, “Unexpected token” was relating to using typescript annotations, however it was again a false positive as the syntax was correct, but this was rather another false positive due to using ts type definitions with ESLint. This was removed with resolution of the ‘error’.

As can be seen, with some configuration modifications, ESLint is now producing correct output with 0 problems for our codebase.



```
[+] Z:\CEN3031_Group10_Project\server\config\db.js                          0 problems
[+] Z:\CEN3031_Group10_Project\server\index.js                              0 problems
[+] Z:\CEN3031_Group10_Project\server\middleware\logger.js                  0 problems
[+] Z:\CEN3031_Group10_Project\server\model\accountModel.js                 0 problems
[+] Z:\CEN3031_Group10_Project\server\model\resourceModel.js                0 problems
[+] Z:\CEN3031_Group10_Project\server\model\typeModel.js                    0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\isloggedin.js                  0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\login.js                       0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\logout.js                      0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\signup.js                      0 problems
[+] Z:\CEN3031_Group10_Project\server\routes\zipCheck.js                    0 problems
[+] Z:\CEN3031_Group10_Project\src\App.js                                   0 problems
[+] Z:\CEN3031_Group10_Project\src\App.test.js                              0 problems
[+] Z:\CEN3031_Group10_Project\src\components\AddResources.js               0 problems
[+] Z:\CEN3031_Group10_Project\src\components\FilteredListCards.js          0 problems
[+] Z:\CEN3031_Group10_Project\src\components\FilteredListView.js           0 problems
[+] Z:\CEN3031_Group10_Project\src\components\NavBar.js                     0 problems
[+] Z:\CEN3031_Group10_Project\src\components\ResultCards.js                0 problems
[+] Z:\CEN3031_Group10_Project\src\components\SearchANDListView.js          0 problems
[+] Z:\CEN3031_Group10_Project\src\components\SearchComponent.js            0 problems
[+] Z:\CEN3031_Group10_Project\src\components\about.js                      0 problems
[+] Z:\CEN3031_Group10_Project\src\components\account.js                    0 problems
[+] Z:\CEN3031_Group10_Project\src\components\addResourceComplete.js        0 problems
[+] Z:\CEN3031_Group10_Project\src\components\dashboard.js                  0 problems
[+] Z:\CEN3031_Group10_Project\src\components\googleMap.js                  0 problems
[+] Z:\CEN3031_Group10_Project\src\components\home.js                       0 problems
[+] Z:\CEN3031_Group10_Project\src\components\login.js                      0 problems
[+] Z:\CEN3031_Group10_Project\src\components\logout.js                     0 problems
[+] Z:\CEN3031_Group10_Project\src\components\signup.js                     0 problems
[+] Z:\CEN3031_Group10_Project\src\index.js                                 0 problems
[+] Z:\CEN3031_Group10_Project\src\reportWebVitals.js                       0 problems
[+] Z:\CEN3031_Group10_Project\src\setupTests.js                            0 problems
[+] Z:\CEN3031_Group10_Project\src\signup.js                                0 problems
```

SECTION 3 - Technical Information

Technical Details

The application is split into three technical layers, the user-facing front-end, and the server back-end, and the database. The languages and tools used on the front-end are the classic web- development trio: HTML for document markup, CSS for styling and design, and JavaScript for interactive components. Additionally, the React framework for JavaScript was used on the front end to implement web application components that could be more easily written and integrated into a modern web application.

The backend server was implemented using the Express.js JavaScript server-side framework. The Node.js framework was used for server-side code execution as well as for project and dependency management. The database used was MongoDB which is a no-sql non-relational database. The back-end server frameworks work in tandem to fetch content from the user and send it to the database. They also work in the reverse, responding to user queries and fetching information from the database to be served to the front-end to be displayed by the user-interface.

Installation

The following steps can followed to install and run the web application:

1. Clone the repository from our GitHub link provided above
2. Open the folder in VSCode
3. From the command line run "npm install" to install all necessary dependencies for the project via the npm package manager
4. From the command line run "npm run dev". This will activate a script that will start a react development server on http://localhost:3000 and a development Express.js server on http://localhost:3001. The development project is configured to use these port settings and they should not be changed as it will break application logic.
5. A browser window should automatically open in your preferred operating system's browser at http://localhost:3000 and the main page of the application should automatically load.

Login and API Keys

For security purposes, the database and google maps API have been locked to only allow IPiddresses of group members. The MongoDB username and password are as follows:

Username: cen3031group10

Password: dangerzone

The google maps geolocation API key used for user lookup of zip code based on latitude and longitude data from the browser is the following:

Key: key=AIzaSyBkW9_W0_3RA0eJ7zddGbVj667mZe—cPM

For the Application, login can be tested using the following credentials:

Username: aUser@homelesshelper.com

Password: securepassword

SECTION 4 - Risk Management Plan and Software Quality Attributes

Risk Management Plan

In order to adequately manage risk our team implemented a number of DevOps principles. Our team had joint responsibility over all software development and delivery in order to ensure code integrity and individual accountability. Accountability and integrity were maintained through use of a software version control system, namely git. We automated as many tests and deployment operations as possible in order to increase the rate of software deployment, increase team productivity, and reduce the overall risk inherent in the software development process. We also practiced continuous integration (CI) through the use of the CircleCI tool. Implementing these methods and principles alongside the principles of agile development and scrum helped us reduce the number and impact of risk factors we encountered during development.

During the development of our application, some of the risks we initially identified were realized. In particular, training on the required technologies did take longer than expected. However, since we chose well-documented technologies like the MERN stack and Material UI, with which some of our team members already had experience, we were able to overcome this problem and still deliver a functioning prototype. In addition, we faced project development delays due to team member emergencies and unforeseen time requirements. One of our team members was deeply impacted by Hurricane Ian and faced an emergency situation which prevented project development work for a period of time and delayed task completion. Three of our team members were also in two other classes together, which both became unexpectedly and extremely time consuming. These factors made it difficult for us to devote as much time as we had hoped to the development of our prototype. However, because we worked ahead to build it time for delays and assigned team members overlapping responsibilities, we were still able to achieve our goals for the application prototype.

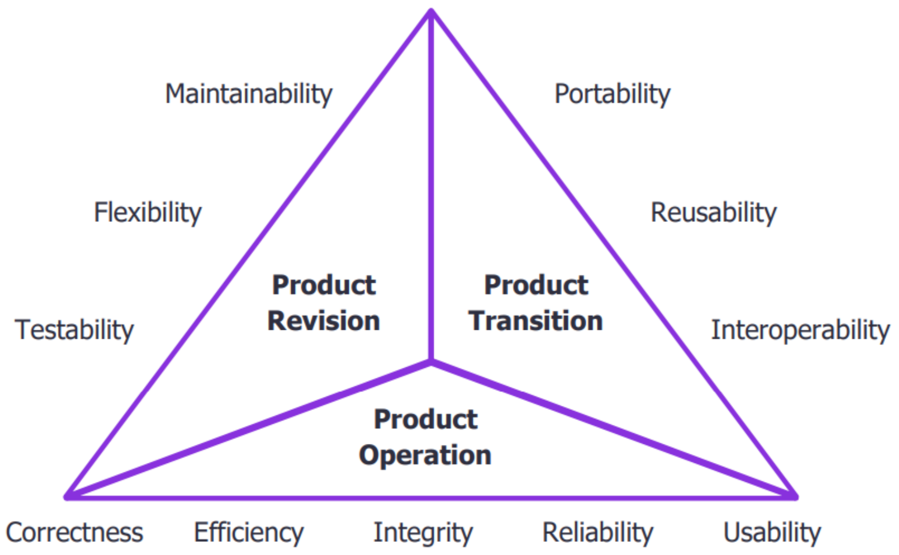
Risk	Probability	Effects of Risk	Strategy
Database chosen cannot support estimated number of users or volume of queries	Low	Serious	In the planning phases of the project, ensure that appropriate database technologies are chosen such that anticipated future expansion is possible

			and supported without having to shift to another product.
Team Member(s) Illness/Emergency	Moderate	Tolerable	Complete work ahead of schedule to allow for unexpected delays; overlapping responsibilities to ensure redundancy.
Tools risk	Very low	Very High	If tools are unstable or become unavailable (e.g. github going offline), risks would be very high. To mitigate this possibility, only use high quality, reputable tools. Avoid small/minimally supported dependencies and tools.
Time to develop project underestimated	High	Serious	Use prior experience to anticipate delays and realistic timelines. Build in time for roadblocks, which are inevitable.
Code Errors/Bugs	High	Tolerable	Test automation, code quality analysis, automated style guideline enforcement, and version control through Git, codeql-analysis, and CircleCI
Training the development team on required technologies takes too long.	Moderate	Serious	Ensure popular, well documented technologies are chosen for the project and try to overlap technologies chosen with team member experience.
Changes to project scope causes major necessary code changes	Low	Serious	Be diligent in the planning phase to ensure project scope does not change unexpectedly. Use

			agile methods so adaptation to changes is easier for the team.
Team member(s) drops the class	Low	Very High	Every team member must own every aspect of the product such that if a team member were lost, the team would be flexible and knowledgeable enough to cover the lost member's work.

Software Quality Attributes

During the software construction phase of our application development our team focused on writing clean, readable, and maintainable code that upheld McCall's 11 software quality factors, as depicted below.



McCall's Quality Factors

In order to achieve this goal, we wrote code that used the accepted design patterns of our chosen languages, technologies, and frameworks. Through use of the React framework we were able to maintain a relatively flat inheritance hierarchy which contributed greatly to the ease of maintaining and understanding our code base. In addition, React and Material UI components are designed to simplify and accelerate the web design process, so they are highly reusable, interoperable, and portable. The component and function names clearly describe code's purpose, minimizing the effort required to understand how the software works and so improving usability. When we created our own components and functions, we used accepted React terminology in assigning variable names to maintain usability.

We also took advantage of the modularity of React and Material UI components to design a code base in which each piece of functionality was contained within a function or components, allowing for easy testability.

We also used a code formatting known as prettier to ensure we maintained constant code formatting standards. This also helped improve the maintainability and usability of our code base, as well as its flexibility, or its ability to be easily improved. Clear and readable code is also easier to test and maintain. To ensure our code was correct, we used ESlint to catch minor programming errors early in development as well as codeQL to catch security errors. We also employed test-driven development principles by identifying tests for each software component before implementation and running the tests on the component before marking it complete. This helped us ensure our application functioned as expected according to our original design and expected user requirements.

In addition, we aimed to develop a code base with a minimal level of complexity that maintained good programming practices. We were able to limit the complexity present in our code and application by constantly analyzing the simplicity of the code base and the functionality that was being implemented by asking ourselves the question, “Is it simple?”. Some ways in which we would look for unnecessary complexity was by examining the code for some common ‘code smells’. We took advantage of the existing efficiency and reliability of React and Material UI components to keep the size and scope of our classes, methods and functions to the absolute minimum needed to implement their functionality. Lastly, we categorized users into different groups enjoying different sets of permissions and established a log-in system with password verification to ensure the integrity of our software.

References

[1] Mozilla Development Network (MND) Contributors, "MVC," 20 September 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/MVC.

[2] TechTerms.con, "Client-Server Model," TechTerms.com, 17 June 2016. [Online]. Available: https://techterms.com/definition/client-server_model.