

Host Your House

Contents

1	Overview	5
	Individual Contributions Summary	5
2	Project Plan	6
	Project Overview	6
	Project Purpose & Objectives	6
	<i>Project Scope</i>	6
	<i>Out of Scope</i>	7
	<i>Assumptions</i>	7
	<i>Constraints</i>	7
	<i>Project Deliverables</i>	7
	<i>Schedule Summary</i>	7
	Project Organization	8
	<i>Roles and Responsibilities</i>	8
	Managerial Process Plans	8
	<i>Start-up Plan</i>	8
	Estimation Plan	8
	Staffing Plan	8
	<i>Work Plan</i>	8
	Work Activities	8
	Schedule Allocation	8
	<i>Control Plan</i>	9
	Requirements Control Plan	9
	Schedule Control Plan	9
	Quality Control Plan	9
	Risk Management Plan	9
	Technical Process Plans	9
	<i>Process Model</i>	9
	<i>Methods, Tools and Techniques</i>	10
	<i>Infrastructure Plan</i>	10
	Supporting Process Plans	10

	<i>Documentation Plan</i>	10
	<i>Quality Assurance Plan</i>	10
	<i>Problem Resolution Plan</i>	10
3	Requirements Specification	10
4	System Specification	11
5	User's Guide	11
	Configuring the Application - For Local Users	11
	<i>System Requirements</i>	11
	<i>Running the Application</i>	11
	Landing Page	12
	Login and Registering	13
	Reserving Properties	14
	User Management	15
6	Test Plan and Results	15
	Test Plan Identifier	15
	Revision History	15
	Introduction	16
	Features to Be Tested	16
	Main features test	17
	Approach	18
	Item Pass/Fail Criteria	18
	Properties Testing	18
	Rental Properties (1)	18
	User Management Testing	22
	User Management (2)	22
	Save and Reserve Testing	25
	Save and reserve (e)	25
	Test Deliverables	27
7	Design and Alternate designs	27
	<i>Purpose</i>	27
	<i>Scope</i>	27
	<i>Overview</i>	27
	<i>Reference Material</i>	28

Design Considerations	28
Style	28
Technologies to Use	28
Design Strengths	28
Limitations	28
Assumptions and Dependencies	28
General Constraints	29
Goals and Guidelines	29
<i>Goals</i>	29
<i>Guidelines</i>	29
Development Methods	29
System Overview	29
System Architecture	30
Architectural Design	30
Design Rationale	30
Data Design	31
Data Description	31
Data Dictionary	31
Human Interface Design	32
Overview of User Interface	32
Screen Images:	33
Requirements Matrix	37
8 Development History	38
<i>Project Initiation:</i>	38
<i>Project Plan:</i>	38
<i>User Guide and Test Plan:</i>	38
<i>Product Design:</i>	38
<i>Phase One:</i>	38
<i>Phase Two:</i>	38
<i>Phase Three:</i>	39
<i>Phase Four:</i>	39
9 Conclusions	39
Suggestions for Further Improvements	39

Key Takeaways	39
<i>Caleb Elkins</i>	39
<i>Lindley Thomas</i>	40
<i>Paul Ambrester</i>	41
<i>Troy Raines</i>	41

1 Overview

Consumers are frequently plagued by hotel prices that are exorbitant and frequently have fees that are designed to keep the customer from canceling or changing the time and date of the rental. On top of that, hotels are not always located in rural areas that don't get a lot of traffic or that aren't in major cities. To alleviate that, Airbnb was created to allow users to rent out their houses if they met certain criteria. This would afford frequent travelers the ability to stay in places that felt like home and weren't directly in the city where most hotels were located. The problem is that there aren't many Airbnb competitors, which means stagnation in the industry and a monopoly on the housing market. To combat that, we developed Host a House, a web-app that would allow users to host their own house to rent, giving another option in the house rental space.

Host Your House is a Full-stack Web application that uses React for the frontend, Node.js for the backend, and MongoDB for the database. frameworks/technologies like Express, Mongoose, Bootstrap, to facilitate development and design.

Individual Contributions Summary

Khalil Savoy:

- ❖ Acted as Product Owner and managed timelines, milestones, and helped develop technical documents
- ❖ Developed basic mockups and system architectural diagrams
- ❖ Developed the user authentication & authorization functionality
- ❖ Assisted the primary development teams overcome roadblocks and filled in as a developer where needed
- ❖ helped with initial set-up of application

Caleb Elkins:

- ❖ Created React UI components
- ❖ Iteratively styled pages and components of the site
- ❖ Unified site pages with a consistent theme

Geraldo Moura:

- ❖ Initial Setup - Created Repository on GitHub, created starting files for backend and frontend.
- ❖ Frontend - Create initial React app, added functionality to reserve houses saving on database for each user. Small tweaks on CSS to improve UX and UI.
- ❖ Backend - Create seed files for houses database, populated initial houses on database. Created endpoints to retrieve all houses from the database.

Troy Raines:

- ❖ Contributed to documentation
- ❖ Supported testing throughout development
- ❖ Provided feedback on visual layout etc.

Thomas Lindley:

- ❖ Main Page - Search and sort, initial design, load featured properties from database, and load all properties dynamically from database
- ❖ Reserved Page - linked to backend to retrieve data from database
- ❖ Saved Page - linked to backend to retrieve data from database
- ❖ Backend Endpoints - setup house endpoints for search and sort, setup user endpoints for retrieving saved and reserved properties, setup endpoints to update database with reservation data, and setup endpoint to retrieve user account information
- ❖ Contributed to documentation
- ❖ Reservation Management - logic to ensure users cannot make a reservation if dates conflict with other user reservations.
- ❖ Login/Registration - added more alerts to inform user of invalid inputs

Paul Ambrester:

- ❖ Created and configured Mongo database
- ❖ Contributed to documentation
- ❖ Testing throughout development
- ❖ Contributed to user information between frontend and backend

2 Project Plan

Project Overview

Project Purpose & Objectives

As stated in earlier sections, Host-A-House was designed to give property owners and people looking to rent another option in the housing market. We planned to develop a full-fledged web-application that would provide users the tools to make informed decisions on what properties they would want to stay in and what is the best deal for their money. By the end of the project our web application would be able to show users a range of properties, each with their own ratings, prices, and locations. We also intend to complete a featured listings section where users can see the most highly rated properties over a variety of categories and prices. The properties in both the featured categories and the general listings would be able to be reserved or saved for later review by users. Our goal is to create an application that can compete with Airbnb and give people another option.

Project Scope

This effort is determined to develop a high-quality web-app with a modern and responsive feel. The scope of this project includes a dynamic front-end and a robust back-end API that supports filtering and querying of properties and users. The application should allow users to create accounts and rent a property for a specified time frame. The application will also

allow users to cancel or change the date of the request and will have an account page in which current reservations can be seen.

Out of Scope

Out of scope for this effort is the ability to send confirmation notices via email or SMS messages and support for a fully functioning payment system is also out of scope. The ability to send the owner of the property a message within the app was also determined to be out of scope at this time. Finally, we have determined that due to time constraints and technological limitations that the ability to post a property for renting will also only be able to be completed if time allows.

Assumptions

Currently, there are no assumptions as there are no resources outside of personnel being provided to the team in the form of money, hardware, or software. The team at this moment in time also does not have any assumptions due to a lack of prior knowledge and frameworks.

Constraints

Current constraints on the project are a shortened time frame of only a month and a half and a limitation on programmatic tools to open-source software and free programmatic tools.

Project Deliverables

The project deliverables for this project have been determined to be a fully functional web-app. Project files will be hosted in an accessible GitHub repository for easy downloading, and the MongoDB database will be based on their ATLAS platform which can be referenced until one week after the end of the course. On top of a running program, the team plans to deliver a final product document that consists of our development plans, designs, and testing criteria.

Schedule Summary

As of the time this report was written, there are 6 weeks to develop the application and polish and deploy deliverables for the project. From March 30 to April 5, the development team will focus on user creation and the login and authorization module. This will include back-end work such as creating user models, setting up the database, and defining user retrieval methods. It will also include front-end work in the form of developing a log-in screen and confirmation of successful login. From April 6 to April 19, the team will be focused on developing the rental listing module. This module will allow users to search for, filter, and look at rental properties. Most of the work for this module will be in developing the front-end pages for displaying the list of properties and showing more detailed information but work must also be done in the back end to retrieve property details and images. From April 20 to April 26, we will be developing the ability to create and review reservations and save potential properties which will require equal amounts of front-end and back-end work. From April 27 to May 3, we will be working on the rating system and coming up with a weekly featured page that shows the hottest properties of the week. This will also be the time when we accomplish any stretch goals for the project.

Finally, from May 4 to May 10, we will work on polishing the application and quality assurance testing in order to hand over deliverables at the end of the timeline.

Project Organization

Roles and Responsibilities

The team will consist of 3 different roles: Project Manager, Project Owner, and Software developers. The project manager role has been filled by Troy Raines and is responsible for write-ups of documentation and ensuring the team is adhering to the timeline that has been presented in earlier sections. The product owner is Khalil Savoy and has the responsibilities of managing the task list and assisting in any task that has fallen behind or where assistance is required. Caleb Elkins, Geraldo Moura De Oliveira, Thomas Lindley, Paul Ambrester makes up our list of software developers and their responsibilities lie solely in developing the application.

Managerial Process Plans

Start-up Plan

Estimation Plan

As only open-source software and resources are utilized the cost will be nil, the projected start date is March 16th, 2022, with a target end date of May 10, 2022.

Staffing Plan

This project shall have six members that contribute to it. The current roles assigned, and the number of people assigned to each role are as follows: One project manager, One Project owner, and 4 software developers with all members of the team doing testing and quality assurance.

Each Member of the group is also expected to review documents when available to ensure the documents have correct grammar, spelling, and the documents design and flow is acceptable.

Work Plan

Work Activities

Work activities will consist of daily software development in either the back end or the front-end of the application. Features of the project have been broken down into smaller actionable items that can be worked on in parallel by each member of the team. Tasks relating to infrastructure or system architecture will be relegated to the product owner and if needed, a software developer.

Schedule Allocation

There are currently no constraints when it comes to scheduling but it can be assumed that each member of the team is able to contribute 1-2 hours of development time a day on average. Tasks will be broken into week-long sprints, twice a week, once on Wednesday and

then on Friday the team can discuss what went wrong, what went well, and how we can improve in order to deliver a high-quality product.

Control Plan

Requirements Control Plan

Product requirements will be tracked in Trello for visibility across the entire team and in order to avoid overlap in tasks assigned. We have also created a discord server in order to communicate changes in deadlines, expected delivery dates, and roadblocks that may appear.

Schedule Control Plan

Utilization of the Trello board will be key in keeping members of the team focused and accountable for missed deadlines and milestones. The Trello board provides statistics for the current amount of development time spent on a ticket and can be configured to warn a user if the ticket has been sitting for too long. The team has also constructed and committed to a plan that involves 2 weekly check-ins in order to discuss progress, roadblocks, and potentially missed deadlines. In the case that a task must be dropped, or a developer needs assistance, it is the responsibility of the product owner to step in where applicable.

Quality Control Plan

In order to assure quality deliverables, the team will be utilizing GitHub for branch management and testing. Users must have their branch reviewed, tested, and approved by another member of the team before the code can be added to the master branch. This ensures that no breaking changes are added to the application which could delay timelines.

Risk Management Plan

Risks such as member outages, missed deadlines, and roadblocks happen but our goal is to mitigate them as much as possible. Our method of implementation to mitigate these risks is by investing in great communication. Having two weekly check-ins enables the team to communicate when things are not going as planned, work is falling behind, or when any number of risk factors pop up.

Technical Process Plans

Process Model

Our process model is one based on the agile framework that has been popularized in recent times. Each ticket contributes to a larger feature that makes up one part of the application but is essentially independent of each other in that they can be worked on by different developers without confrontation happening. Information regarding the project will be communicated in the discord channel that has been set up previously and timelines and major milestones have been determined and communicated prior to this document being written with our Trello ticketing system also relaying that information to developers.

Methods, Tools and Techniques

Project files will be hosted in an accessible GitHub repository for easy downloading, and the MongoDB database will be based on their ATLAS platform which can be referenced until one week after the end of the course. This project will use the MERN stack, which is MongoDB for database, ExpressJS as framework for backend server, ReactJS as the frontend framework, NodeJS backend server.

Infrastructure Plan

Hardware required for development is a computer with at least 8GB of ram, at least 4 cores CPU, and internet access. The machine will need to have NodeJS version 16 or later, NPM version 6, and an IDE. Recommended IDE is Visual Studio Code with the following extensions: ESLint, GitHub Pull Request and Issues, Path Intellisense, Prettier - Code formatter.

Supporting Process Plans

Documentation Plan

Deliverables will be compiled previously and will be delivered in the form of a compressed file containing all project source files and a README which will instruct the user on how to utilize the system. An alternative server that is already up and running will also be delivered so that the user does not have to set up the project themselves. Finally, documentation on how to use the system will be included along with the source files and on a separate web page linked to our running server.

Quality Assurance Plan

There are two ways that the team will utilize in order to assure quality control and assurance. The first part of that plan is code reviews before a ticket or feature is added to the official project. This ensures that no breaking changes are ever introduced into the code and will also inspire communication between team members. The second method is our twice weekly check-ins. This allows each team member to relay their status and blockers before it becomes too late to fix them. These two preventative measures will ensure that problems and bugs are handled swiftly and don't interrupt our current timeline.

Problem Resolution Plan

In order to reduce and track problems that arise we will be utilizing the review and comment features that GitHub has. By noting problems in the code on Merge requests, we can keep track of problems that arise, and we can refer to them if need be. We will also be using the comment feature on the Trello tickets in order to track changes to criteria or feature development for any reason. This also helps us resolve and refer to issues that crop up in the future.

3 Requirements Specification

Users of this application shall be able to:

- Account Management

- Register
- Login and logout
- Change user account information (i.e., name, location)
- Viewing of Properties
 - View Featured properties on home page
 - View all properties on home page
 - Search all properties by city, price, rating and reservation date available
 - Sort all properties in either descending or ascending order by city, price, and rating
- Save and Reserve Properties
 - Save properties to their account
 - Reserve properties to their account

4 System Specification

This application is a Full-Stack web application meant for desktop web browsers. The application can be displayed on mobile devices but has not been thoroughly tested and designed for mobile devices. Server hardware requirements are completely dependent on the amount of traffic expected to the website.

The server to run this application software requirements are:

- Node version 16.14.2
- NPM version 8.6.0
- MongoDB atlas cloud database deployed on an AWS M0 Cluster
- React version 18.0.0

5 User's Guide

Configuring the Application - For Local Users

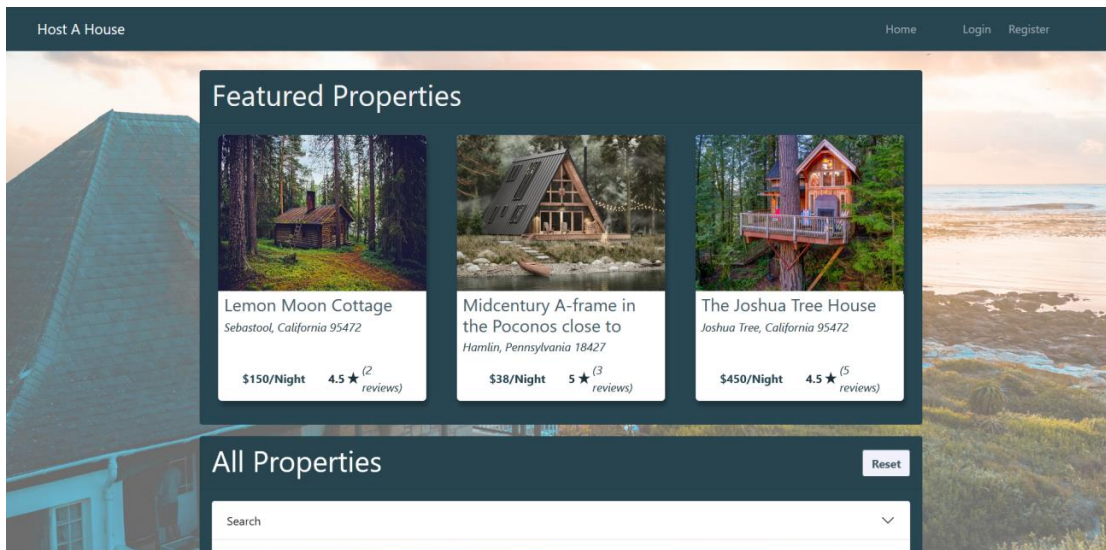
System Requirements

- NPM version 8.6.0
- MongoDB atlas cloud database deployed on an AWS M0 Cluster
- React Node version 16.14.2
- version 18.0.0

Running the Application

- Open a terminal
- Clone the repo from the following URL: Host A House Repo
- In the backend directory of the project, add the desired port and the connection URL of the MongoDB database to the config.env file
- In the root directory of the project run npm run dev in order to start both the backend server and the frontend client

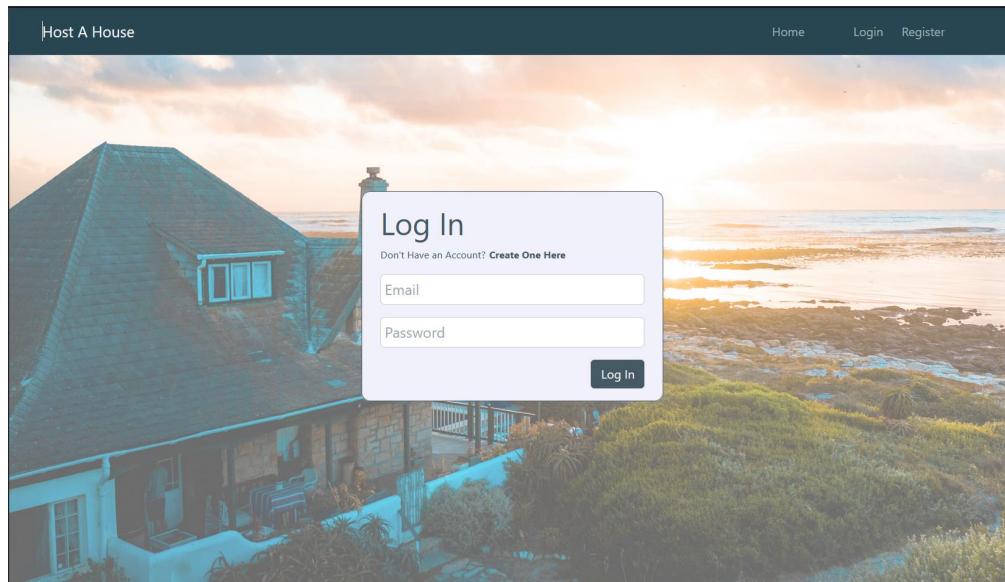
Landing Page



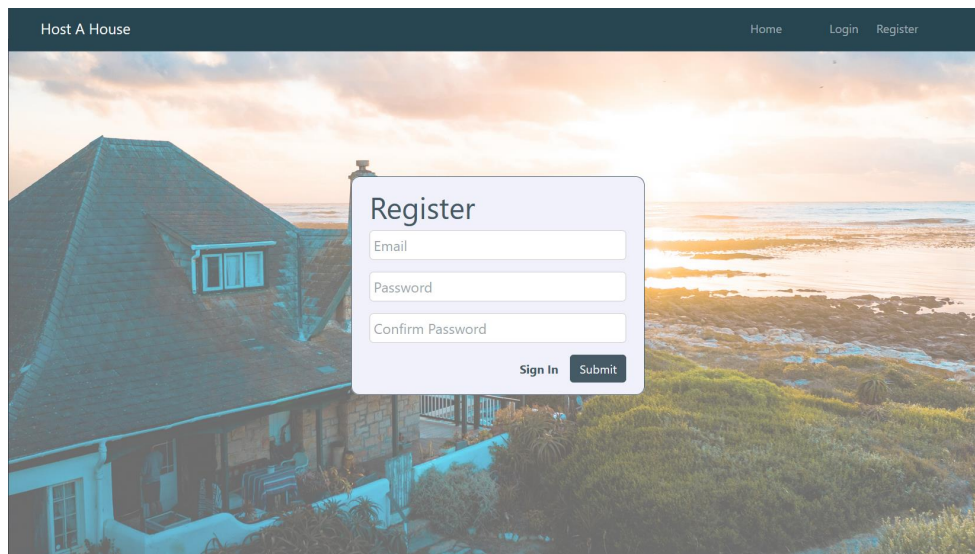
The Host A House landing page is what all users will be redirected to if they are not signed in or do not put in a specific path in the URL bar. When no information is entered, the landing page displays the featured list of houses which is a list of the top-rated properties available for booking at that current moment. However, users are also able to sort the list by date, location, rating, & price. If a timeframe has been entered by a user, the application will not show properties that have already been booked for the timeframe entered. When users click on any of the properties, more information on the property is shown, including but not limited to more pictures of the property. In order to reserve a property users must sign in using the sign-in button found on the right-hand side of the navigation bar.

Login and Registering

If a user is not signed in, hitting the sign-in button to the right of the navigation bar will take them to the login page. The login page will then ask for the user's email and password but will also have a button that will take users to the Sign-Up page if they have not created an account.



In the Sign-Up page the user will be able to create an account by inputting a valid email, password, and confirming the previously entered password. By clicking the register button, a new user will be created if the information is verified to be correct, and the email is not a duplicate. If the user wants to cancel signing up or wants to return to the login page, buttons for those actions will also be available.



Reserving Properties

Properties can be reserved for a specified timeframe by logging into the application and clicking on a property on the landing page. Once more information about that property has been shown, a reserve button can be clicked for the user to reserve the house. If a user has not entered a timeframe on the landing page, the reserve button will prompt a user for the dates they want to reserve, at which point the system will let them know whether the property is available for booking or not. If the user has entered a timeframe on the landing page, the timeframe will be applied to the booking automatically and the property will be booked.

The screenshot shows a web application interface for 'Host A House'. A modal titled 'Reserve Now' is open, allowing users to book a property. The modal includes a 'Back to Listings' link, a property image, and a reservation form. The form has fields for 'Check In' and 'Check Out', each with 'Month', 'Day', and 'Year' dropdowns. Below the form are 'Cancel' and 'Confirm Reservation' buttons. The background of the modal displays property details: 'Parking included', 'Breakfast included', 'Pet-friendly', 'Free Airport Shuttle', 'Cleaning and Safety Practices' (including disinfectant use, hand sanitizer, PPE, and social distancing), and a list of nearby locations with drive times: Baltimore, MD (11 min), Washington Intl. (11 min), Marshall (12 min), Baltimore Cruise Terminal (12 min), and Fort McHenry (13 min).

Host A House

Home Login Register

← Back to Listings

Reserve Now

Check In Month 1 Day 01 Year 2022

Check Out Month 1 Day 01 Year 2022

Cancel Confirm Reservation

Parking included

Breakfast included

Pet-friendly

Free Airport Shuttle

Cleaning and Safety Practices

Cleaned with disinfectant

Hand sanitizer provided

Personal protective equipment

Social distancing

Baltimore, MD (BMW Baltimore)

Washington Intl. Thurgood Marshall

Baltimore Cruise Terminal

Fort McHenry

11 min drive

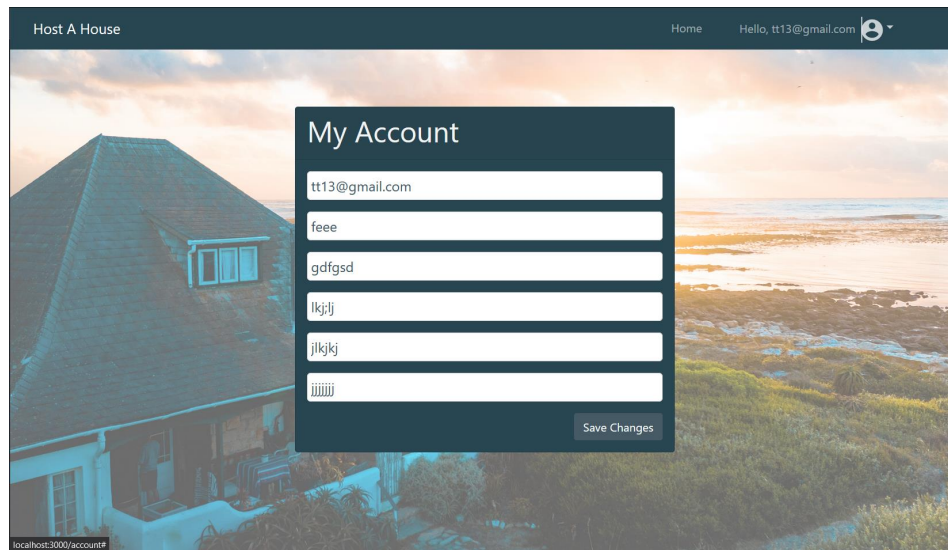
11 min drive

12 min drive

13 min drive

User Management

In the top right corner of our navigation bar, you will find an account button that allows users to go to the user management page or sign out. If a user is not signed into their account, it will take them to the signup page where the user can apply for an account. If the user management button is clicked and a user is already signed in, users are taken to the user management page where they can change the details of their account and reference reservations. Account details that can be changed include their address, their name, and their password. In the future we also have plans to add a payment section where users can store payment information for quick checkout.



6 Test Plan and Results

Test Plan Identifier

- Host A House Initial Test Plan
- App version: 1.00
- Version Date: 02APR2022
- Test Plan Version: 1.04
- Version Date: 04APR2022

Revision History

- v1.00 - (01APR2022) initial document
- v1.01 - (02APR2022) added test cases for featured rentals, rentals list, and overall program test
- v1.02 - (03APR2022)
 - added more test cases to featured rentals and rental list
 - cleaned up duplicates for easier use, will need to test some test items more than once to ensure proper functionality (like logged in user save more than one property,

different properties, etc.) and can be added with an indented letter on table to show it is a sub test

- added tests for all main features
- filled out expect outputs for all tests
- filled out approach, deleted test items, not sure if needed
- unsure of other sections if needed or not
- v1.03 - (04APR2022) added more to introduction, filled out deliverables, and deleted sections not used
- v1.04 - (05APR2022) Added user guide section from discussion post, Paul corrected spelling issues, Geraldo wrote user guide.

Introduction

- Purpose: Create an application users can use to rent houses in various locations.
- Description: Application uses MERN stack to create a fully functioning website.
- Goals: The goal of this project is to create an online, easy to use, and modern looking rental application that uses quality software and quality UI design to deliver an enjoyable experience to customers

Features to Be Tested

- Rental properties
 - list of rental properties
 - search rentals
 - by location
 - by date
 - by price
 - sort rentals
 - by location
 - by date
 - by price
 - featured rentals
- Saved/Reservation management
 - logged in user make reservation
 - logged in user save properties
 - logged in user view save/reserved properties
- User management
 - create user
 - log in/logout
 - account information update

Main features test

Overall Test Case	Requirement	Expected Output	Actual Output	Pass/Fail
	<u>Rental Properties (1)</u>	Shown on the main page, featured at top, all at bottom.	Shown on the main page, featured at top, all at bottom.	Pass
	Featured rentals (1a)	Shows featured properties at top of page	Shows featured properties at top of page	Pass
	List of rental properties (1b)	List of all properties below featured, with search and sort	List of all properties below featured, with search and sort	Pass
	Search rentals (1c)	Allow searching with price, dates and location	Allow searching with price, dates and location	Pass
	Sort rentals (1d)	Allow sorting by date, location, and price	Allow sorting by date, location, and price	Pass
	<u>User Management (2)</u>	Top right of page, allows user account management		
	Register (2a)	Top right of page, allow users to register		
	Login (2b)	Top right of page, allow users to login		
	Logout (2c)	Top right of page in dropdown, allow users to logout		
	Account Information (2d)	Top right of page in dropdown, allow users to view or change account info		
	<u>Saved Property and Reservation</u>	Pages for logged in users, shows saved properties and reservations		

	<u>Management (3)</u>			
	Save property (3a)	Clicking save on a rental property adds property to saved properties. Show saved properties in the user's saved page.		
	Make reservation (3b)	Clicking reserve on a rental property takes the user to a page with more details and dates. Show reserved properties in the user's reserved page.		

Approach

Testing frontend features for correct operations will bring to light bugs in the front end and the backend. The above main features to be tested are an overview of where the project is at and should not be marked pass until detailed testing below is done for the feature

Item Pass/Fail Criteria

Properties Testing

Rental Properties (1)

- Rental properties are shown on the main page of the website. This includes featured properties at the top of the page and all properties underneath featured properties.

Test Case	Requirement/ Input	Expected Output	Actual Output	Pass/Fail
1a	Featured rentals visible	3 featured properties are shown on main page	3 featured properties are shown on main page	Pass
1a.1	Featured rentals visible after login	3 featured properties are shown on main page after login	3 featured properties are shown on main page after login	Pass

1a.2	Featured rentals visible after logout	3 featured properties are shown on main page after logout	3 featured properties are shown on main page after logout	Pass
1a.3	Featured rentals visible after page refresh	3 featured properties are shown on main page after page refresh	3 featured properties are shown on main page after page refresh	Pass
1b	List of rental properties visible	List of all properties below featured, with search and sort	List of all properties below featured, with search and sort	Pass
1b.1	List of rental properties visible after login, logout, and page refresh	List of all properties visible	List of all properties visible	Pass
1c	Search rental properties dropdown present	When search clicked dropdown opens for search	When search clicked dropdown opens for search	Pass
1c.1.1	Search rental properties by city (Sebastool)	Searching for city displays properties in matching city	Searching for city displays properties in matching city	Pass
1c.1.2	Search rental properties by city (Key West)	Searching for city displays properties in matching city	Searching for city displays properties in matching city	Pass
1c.2.1	Search rental properties by price (0 - 100)	Searching for price displays properties that are in price range	Searching for price displays properties that are in price range	Pass
1c.2.2	Search rental properties by price (300 - 400)	Searching for price displays properties that are in price range	Searching for price displays properties that are in price range	Pass
1c.3.1	Search rental properties by stars (3 to 4)	Searching for stars displays properties that having rating in range	Searching for stars displays properties that having rating in range	Pass

1c.3.2	Search rental properties by stars (4 to 5)	Searching for stars displays properties that having rating in range	Searching for stars displays properties that having rating in range	Pass
1c.4.1	Search rental properties by date (05 to 14 Jan, 2022)	Searching for dates displays if property is or is not available in the date range.	Searching for dates displays if property is or is not available in the date range.	Pass
1c.4.2	Search rental properties by date (08 to 09, July 2022)	Searching for dates displays if property is or is not available in the date range.	Searching for dates displays if property is or is not available in the date range.	Pass
1c.4.3	Search rental properties by date (08 to 09, July 2022)	Searching for invalid dates (start day after end day) entered tells user dates are invalid.	Searching for invalid dates (start day after end day) entered tells user dates are invalid.	Pass
1c.5.1	Search rental properties by city and price	Searching for city and price displays results that match	Searching for city and price displays results that match	Pass
1c.5.2	Search rental properties by city and price	Searching for city and price displays results that match	Searching for city and price displays results that match	Pass
1c.6.1	Search rental properties by city and stars	Searching for city and stars displays results that match	Searching for city and stars displays results that match	Pass
1c.6.2	Search rental properties by city and stars	Searching for city and stars displays results that match	Searching for city and stars displays results that match	Pass
1c.7.1	Search rental properties by city and date	Searching for city and date displays results that match	Searching for city and date displays results that match	Pass
1c.7.2	Search rental properties by city and date	Searching for city and date displays results that match	Searching for city and date displays results that match	Pass
1c.8.1	Search rental properties by price and stars	Searching for stars and price displays results that match	Searching for stars and price displays results that match	Pass

1c.8.2	Search rental properties by price and stars	Searching for stars and price displays results that match	Searching for stars and price displays results that match	Pass
1c.9.1	Search rental properties by price and date	Searching for price and date displays results that match	Searching for price and date displays results that match	Pass
1c.9.2	Search rental properties by price and date	Searching for price and date displays results that match	Searching for price and date displays results that match	Pass
1c.10.1	Search rental properties by stars and date	Searching for stars and date displays results that match	Searching for stars and date displays results that match	Pass
1c.10.2	Search rental properties by stars and date	Searching for stars and date displays results that match	Searching for stars and date displays results that match	Pass
1c.11.2	Search rental properties by city, price, and date	Searching for stars, city, price and date displays results that match	Searching for stars, city, price and date displays results that match	Pass
1c.11.2	Search rental properties by city, price, and date	Searching for stars, city, price and date displays results that match	Searching for stars, city, price and date displays results that match	Pass
1d	Sort rental properties dropdown present	When Sort clicked dropdown opens for sort	When sort clicked dropdown opens for sort	Pass
1d.1	Sort by city ascending	When ascending city selected and sort clicked, sorts all properties by city ascending	When ascending city selected and sort clicked, sorts all properties by city ascending	Pass
1d.2	Sort by price ascending	When ascending price selected and sort clicked, sorts all properties by price ascending	When ascending price selected and sort clicked, sorts all properties by price ascending	Pass

1d.3	Sort by rating ascending	When ascending rating selected and sort clicked, sorts all properties by rating ascending	When ascending rating selected and sort clicked, sorts all properties by rating ascending	Pass
1d.4	Sort by city descending	When descending city selected and sort clicked, sorts all properties by city descending	When descending city selected and sort clicked, sorts all properties by city descending	Pass
1d.5	Sort by price descending	When descending price selected and sort clicked, sorts all properties by price descending	When descending price selected and sort clicked, sorts all properties by price descending	Pass
1d.6	Sort by rating descending	When descending rating selected and sort clicked, sorts all properties by rating descending	When descending rating selected and sort clicked, sorts all properties by rating descending	Pass

- Issues noted:
 - -background not fully functional when in dark mode on Firefox (black bar for half page, then the actual background)
 - reset button - shows all properties again after search, but does not reset values in fields, and some values are persistent after reset and added to new search results. Refreshing page fixes issues

User Management Testing

User Management (2)

- At top right of page,
 - if not logged in user sees login and register
 - if logged in, user sees their email/name and a dropdown menu

Test Case	Requirement	Expected Output	Actual Output	Pass/Fail
2	User Management (not logged in)	Present at top right of page, allows user to login/logout	Present at top right of page, allows user to login/logout	Pass

2.1	User Management (logged in)	Present at top right of page, displays user email/name, dropdown shows my account, saved, reserved, and logout links	Present at top right of page, displays user email/name, dropdown shows my account, saved, reserved, and logout links	Pass
2a.1	New User Register (link to page)	Clicking register takes user to registration page	Clicking register takes user to registration page	Pass
2a.2	Register with invalid email (fff)	Registering with invalid email alerts "Email is not valid!"	Registering with invalid email alerts "Email is not valid!"	Pass
2a.3	Registering with invalid password (too short)	Registering with short password alerts "Please use at least 8..."	Registering with short password alerts "Please use at least 8..."	Pass
2a.4	Registering with non-matching password	Registering with non-matching password alerts. "Passwords do not match!"	Registering with non-matching password alerts. "Passwords do not match!"	Pass
2a.5	Registering with no password	Registering with no password should alert the user.	Registering without a password doesn't alert the user and still creates an account.	Pass (fixed)
2a.6	Registering with an email address that already exists	Registering with an email address that already exists alerts the user that the account already exists	Registering with an email address that already exists alerts the user that the account already exists	Pass
2a.7	Registering with an email address that already exists does not change existing users' password	Existing user's password remains the same.	Existing user's password remains the same. (Ran after above with same user and logged in with correct password)	Pass

2a.8	Registering with no issues	When correct information is entered, the user is registered and sent to the login page.	When correct information is entered, the user is registered and sent to the login page.	Pass
2b.1	Login (free@gmail.com)	Top right of page, allow users to login, when clicked and correct info entered user is logged in.	Top right of page, allow users to login, when clicked and correct info entered user is logged in.	Pass
2b.2	Login (tt13@gmail.com)	Top right of page, allow users to login, when clicked and correct info entered user is logged in.	Top right of page, allow users to login, when clicked and correct info entered user is logged in.	Pass
2c.1	Logout (free@gmail.com)	Top right of page in drop down menu, logs user out.	Top right of page in drop down menu, logs user out.	Pass
2c.2	Logout (tt13@gmail.com)	Top right of page in drop down menu, logs user out.	Top right of page in drop down menu, logs user out.	Pass
2d	Account Information (user has account info saved)	When logged in, clicking my account in the drop-down menu brings the user to a page displaying account information saved.	When logged in, clicking my account in the drop-down menu brings the user to a page displaying account information saved.	Pass
2d.1	Account Information (user has no account info saved)	When logged in, clicking my account in the drop-down menu brings the user to a page displaying account information with blank fields.	When logged in, clicking my account in the drop-down menu brings the user to a page displaying account information with blank fields.	Pass
2d.2	Update account Information (user has no	When a user adds data to the fields on the my account page and clicks save, the	When a user adds data to the fields on the my account page and clicks save, the information will be stored.	Pass

	account info saved)	information will be stored.		
2d.3	Update account Information (user has account info saved)	When a user adds data to the fields on the my account page and clicks save, the information will be updated.	When a user adds data to the fields on the my account page and clicks save, the information will be updated.	Pass

Save and Reserve Testing

Save and reserve (e)

- buttons located on properties in the main page
- logged in users can save and reserve
- logged in users can view saved and reserved properties by clicking dropdown in top right of page and clicking saved or reserved properties.

Test Case	Requirement	Expected Output	Actual Output	Pass/Fail
3	Saved Property	When logged in, clicking saved properties from the dropdown takes the user to a page showing saved properties.	When logged in, clicking saved properties from the dropdown takes the user to a page showing saved properties.	Pass
3.1	Reserved Property	When logged in, clicking reserved properties from the dropdown takes the user to a page showing reserved properties with the reservation dates.	When logged in, clicking reserved properties from the dropdown takes the user to a page showing reserved properties with the reservation dates.	Pass
3a	Save property (not logged in)	Clicking save on a rental property while not logged in alerts the user they must be logged in to save.	Clicking save on a rental property while not logged in alerts the user they must be logged in to save.	Pass

3a.1	Save property (logged in)	Clicking save on a rental property while logged in alerts the user that the property was saved to the account and shows in their saved page.	Clicking save on a rental property while logged in alerts the user that the property was saved to the account and shows in their saved page.	Pass
3a.2	Save property (logged in, multiple saved)	Clicking save on a rental property while logged in alerts the user that the property was saved to the account and shows in their saved page.	Clicking save on a rental property while logged in alerts the user that the property was saved to the account and shows in their saved page.	Pass (see issues noted below)
3b	Make reservation	Clicking reserve on a rental property takes the user to a page with more details and dates. Show reserved properties in the user's reserved page.	Clicking reserve on a rental property takes the user to a page with more details and dates. Show reserved properties in the user's reserved page.	Pass
3b.1	Make reservation (returning to home)	Clicking back to listings on reserve property returns the user to the homepage.	Clicking back to listings on reserve property returns the user to the homepage.	Pass
3b.2	Make reservation (house is available on dates)	When the user clicks reserve now, and enters dates that are not already taken, the house is reserved, and it shows on the reserved page.	When the user clicks reserve now, and enters dates that are not already taken, the house is reserved, and it shows on the reserved page.	Pass
3b.3	Make reservation (house is not available on dates)	When the user clicks reserve now, and enters dates that are already taken, the house is not reserved, the user is alerted, and does not show on the reserved page.	When the user clicks reserve now, and enters dates that are already taken, the house is not reserved, the user is alerted, and does not show on the reserved page.	Pass

3b.4	Make reservation (house is not available on dates, different user trying to reserve dates taken by another user)	When the user clicks reserve now, and enters dates that are already taken, the house is not reserved, the user is alerted, and does not show on the reserved page.	When the user clicks reserve now, and enters dates that are already taken, the house is not reserved, the user is alerted, and does not show on the reserved page.	Pass
------	--	--	--	------

- Issues noted:
 - can save property more than once, and duplicates show on saved page

Test Deliverables

Test deliverables should include this document with actual outputs and pass/fail information. Screenshots of tests passing should be provided in folders named after features tested, and files should follow test case naming scheme i.e. for changing password test directly above 10.A 10.B etc.

Deliverables for failed tests should be inserted below the above test tables. These should state which test it is referring to, include an image of failure, if possible, detailed description of why it failed and possible solutions.

7 Design and Alternate designs

Purpose

This section's intended purpose is to give more context surrounding the design and implementation of the "Host-A-House" project. The intended audience for this section is our software developers and the stakeholders of the product, which in this case would be the entire team and our professor respectively.

Scope

The scope for the Host-A-House platform includes several core user functionalities regarding property management. Those core functionalities are property searching, property filtering, property reservation, and user management and each core functionality can be broken down into smaller, more manageable tasks. Each core functionality will have a frontend component and a backend component to support visualizations and personalized user feedback. At the end of the development cycle, we are hoping to present a fully fleshed-out property management system that allows users to search and filter existing properties against a wide array of criteria and then reserve these properties for vacations or other ventures.

Overview

This document is meant to provide technical information on the project and is broken down into several sections to communicate this information effectively. The System Overview section discusses the project at a high level and communicates everything in the following sections succinctly and without context. The System Architecture section follows that and

explains how we've structured the project, whether that be file structure, or the architecture patterns used. After, we have the Data Design section discussing how we plan to store and represent the information we need and the Human Interaction Design section in which we discuss the planned user experience and user interface in the form of visual mockups. Finally, we end with a Requirements Matrix and an appendix for any links or citations that we might have.

Reference Material

- <https://www.mongodb.com/languages/mern-stack-tutorial>
- <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>

Design Considerations

Style

Styling can be difficult, but should not go beyond developer knowledge base in this case, which helps keep down the amount of information to look up to complete tasks and reach a final product that looks close to the initial vision

Technologies to Use

There are a seemingly infinite number of frameworks and ways to complete a web application. The technologies or frameworks to use need to be carefully considered, or the team could end up in a mess of different things to remember to use for certain tasks.

Design Strengths

One of the design strengths for the project is how clean, modern and elegant the UI looks. With a few tweaks here and there, it could look like a full production website. As for what we cannot see on the website by looking at it, we used technologies like Mongo DB that could be upscaled and deployed to a full-scale website. Some error checking and security measures would need to be put in place for the web app to be production quality, but the team did manage to cover many areas in this short time frame.

Limitations

One of the limiting factors of the project was that a few members of the team did not have much experience in the technologies and languages used for the project. This meant that a few members had to watch tutorials for the first few weeks of the project and then learn as they went as the project moved forward.

Another limiting factor was the time constraint. With more time the project could have been more robust and redesigned to implement features in a better way.

Assumptions and Dependencies

It is assumed that the user running the app (locally) is using modern computer hardware, has node installed, has internet connectivity, installs dependencies included with the

project with npm install in the front and back end, and has basic knowledge of how to use node with their IDE/Terminal.

It is assumed for this project that the final product will not be hosted publicly on a server for anyone to access. There are security measures that may need to be implemented to make the application secure. There are also missing parts of the project that would not allow it to be a full production application and one of those is payment processing.

General Constraints

- 8 Week timeframe
- Inability to implement certain features like payment, user and house verification, and others due to cost constraint.

Goals and Guidelines

Goals

Deliver a mostly functional full stack web application to compete with Airbnb, that could be improved to a fully functional website that allows users to pay etc.

Guidelines

Do not enter passwords or actual personal information into the website since it is not fully complete and there could be some security risks in doing so.

Development Methods

The development method for this application was a bit unorthodox. The project started with a base that got the front end and back end up and running. From the base developers for the project worked on features one at a time, to build up to a fully functioning application. In some cases, this required setting up react components, and then adding in backend routes and endpoints in order to make the react components dynamic and load from the back end.

System Overview

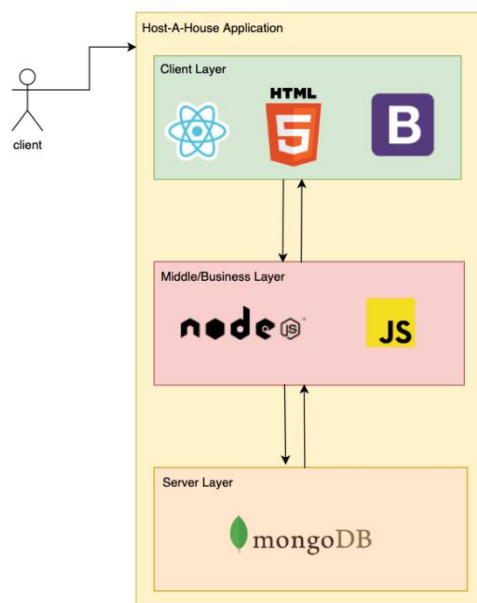
The Host-A-House application uses a layered architectural pattern in order to decompose the application into smaller parts that can be worked on in parallel. The first layer is the UI layer which we built in React and allows us to serve the users dynamic information based on information that they input into the system. We have several views that can appear based on core functionality and features that were presented in prior requirements documentation. Most of the business logic is also baked into the UI layer, such is the nature of the React framework. How users interact and use the system is truly all dictated by the visual components that are rendered on screen. Our second layer is there to provide middleware for connecting the backend and the front-end. This layer contains logic for how the client part of the application is structured, the different web pages that exist, user authentication, and the endpoints responsible for retrieving certain information. By having an additional layer, we can change certain logic such as the way data is

manipulated or global variables with minimal changes to both the UI layer and the data persistence layer. For our middle layer, we are using a combination of express and pure JavaScript to manage both our UI and our data persistence layer. Finally, our data persistence layer is responsible for managing the database and making sure we can connect to it. We have config files that are used in order to properly communicate to the database and a cloud MongoDB server for data to be persisted across the application.

System Architecture

Architectural Design

Our program structure uses a multi-layered architecture in order to make developing the system more efficient and to also abstract parts of the application away from each other in



such a way that we can work on multiple parts of the application at the same time. The client layer consists of the react layer, pure HTML5 for templating, and bootstrap to keep a consistent design across the application. Our middle or business logic layer consists of the node and express frameworks. Express allows us to set up a server that holds our connections to our backend layer and node allows us to dynamically build and load in libraries needed to build the application. Our last layer is our backend layer which consists of a cloud version of MongoDB for persistence across the properties and our users and some pure JavaScript for manipulating any data before it is loaded into the application.

Design Rationale

We went with a layered architectural system since this would allow us to adhere to agile methodologies much easier than if we had used an architecture that incorporated more waterfall techniques. By using a different system architecture and tightly grouping functionality together, our developers would be more reliant on each other's work before they could move on to the next task. This creates unnecessary roadblocks and means that we wouldn't be able to work as efficiently as we would have desired otherwise. By using a layered architecture system, we can have one developer working to build the backend and the models that we'll need in

order to support our visualizations, another developer working on the User Interface that will display this data in the frontend, and a final developer working on the endpoint that would connect these two systems if one was needed. This architecture is what we have found to work for us the best after doing research and proper requirements development.

Data Design

Data Description

The Host a House application uses MongoDB to store all data required for the application users and all the properties data. Mongo DB, unlike SQL databases, uses JSON documents instead of tables to store the data. The data is designed with normalized data models. The normalized data model uses references, `user_id` as an example, to create relationships between documents. The application will contain two documents, a user's document, and a properties document. These documents will have the models our data will use for the database. The user's model will contain a name with a type of string, email with a type of string, password with a type of string, and `isAdmin` with a type of Boolean. The properties model will contain the user (house owner) with a type of string, a name, an image (which is the path of the image in the website assets), a description, a state, and a city with a type of string. Also, a zip Code, rating, number of reviews, price with a type of number, and reservations will be an array of strings. There is also a review schema that will hold the information users will provide as reviews. The review contains a name, and comment with a string type, and a rating with a number type.

Data Dictionary

The properties model document will have three schemas:

- Review Schema:
 - `name: { type: String, required: true }`
 - `rating: { type: Number, required: true }`
 - `comment: { type: String, required: true }`
- House Schema:
 - `user: { required: true, ref: 'User' }`
 - `name: { type: String, required: true }`
 - `image { type: String, required: true }`
 - `description: { type: String, required: true }`
 - `state: { type: String, required: true }`
 - `city: { type: String, required: true, }`
 - `zipCode: { type: Number, required: true, }`
 - `reviews: [reviewSchema],`
 - `rating: { type: Number, required: true, default: 0 }`
 - `numReviews: { type: Number, required: true, default: 0 }`
 - `price: { type: Number, required: true, default: 0 },`
 - `reservations: [{ type: String, required: true, default: 0 }]`
- User Schema:
 - `name: { type: String, required: true }`

- email: { type: String, required: true, unique: true }
- password: { type: String, required: true }
- isAdmin: { type: Boolean, required: true, default: false }

Human Interface Design

Overview of User Interface

From a user's perspective, when they initially hit our application's webpage they are greeted by our main page. The page is populated with a search section, a featured list, and a navigation bar at the top to allow users to sign in and then subsequently access user management features. By clicking the logo in the top bar on any screen you are brought back to this page. The search bar allows users to input a destination along with check-in and out dates, and how many travelers there will be. If any of this information is invalid, the field is highlighted with a tooltip displaying the error. The featured section simply lists our top 5 properties and when clicked will take you to the details page where users can get more information on the property and reserve the property.

Logging in and registering are done from the account buttons found on the right side of the navbar. When either of the buttons is clicked, a modal will pop up for the appropriate action. If any of the information is invalid the field will be highlighted red and a tooltip will show giving the user error more information. If there are any errors for any reason, a window alert will populate the screen displaying the error message.

The listings page is what users are brought to when they complete a search successfully. The listings page keeps the navbar and the search bar at the top of the page but adds more options in order to filter the results that are found underneath the search bar. If no results can be found, the listings found underneath the search bar will be substituted with an error message that says no results found. Each result, if there are any, has its own card that consists of a cover image, the title and location of the property, a brief description of the image, the rating of the property, and the price per night. By clicking on any of the results, the user is brought to a details page that gives more information.

The details page is what users are brought to when they click on any property on any screen. It consists of a section for images that the user can browse through, the information that is found on the listings page, and more information on whatever amenities the property offers. In some cases, additional information is given for attractions that are close to the property.

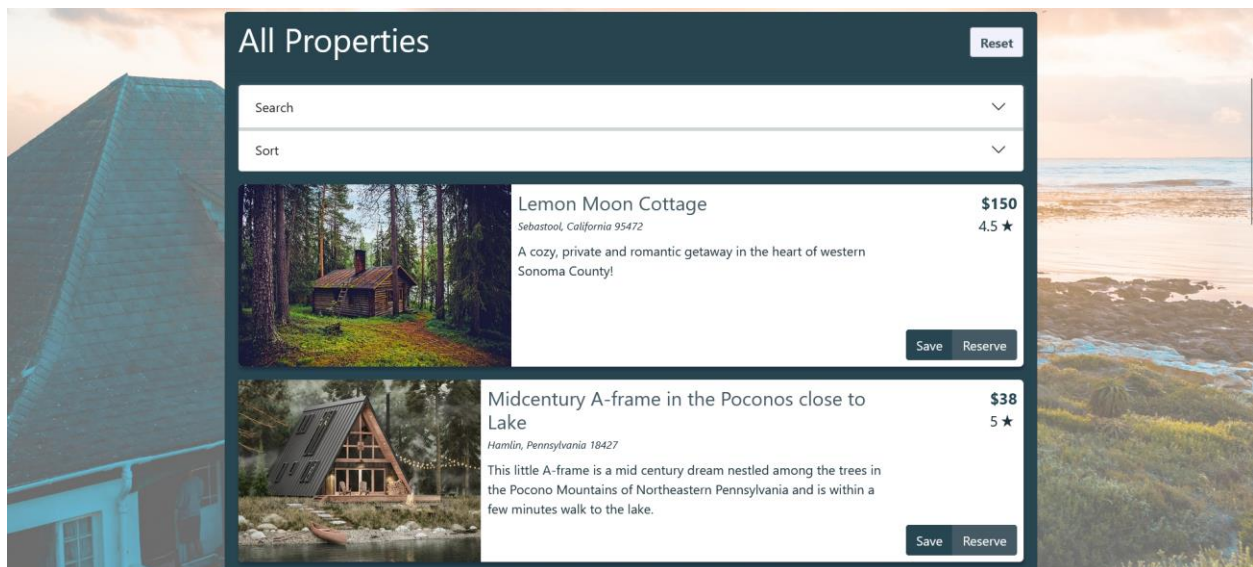
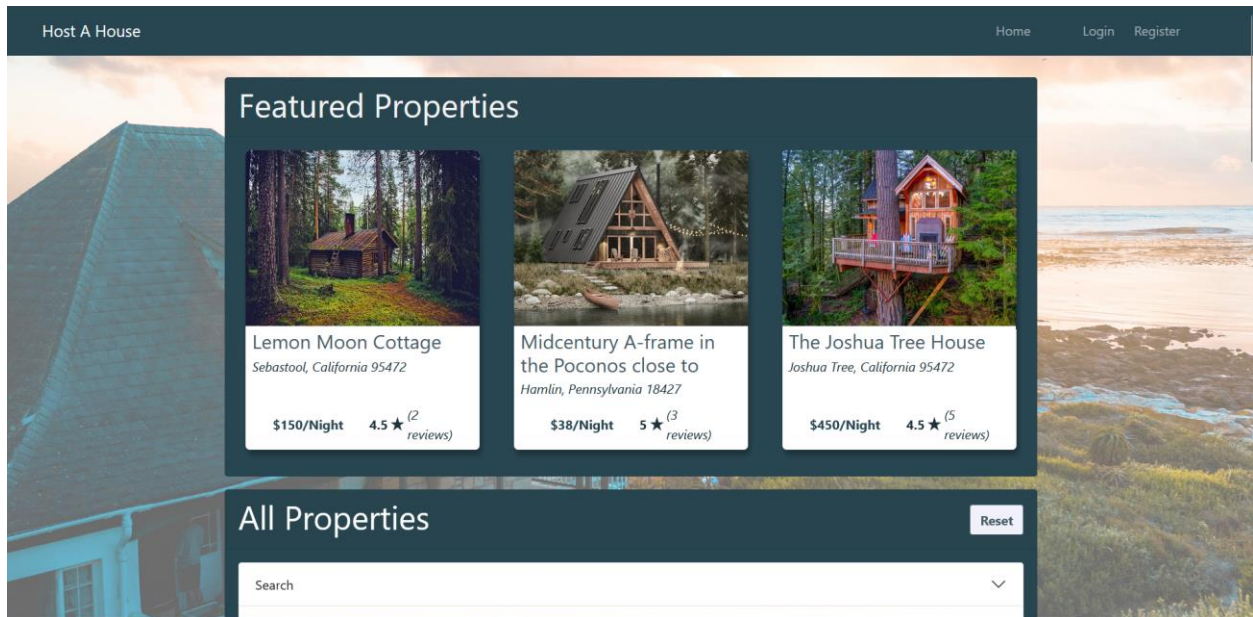
The reservations page is structured similarly to the listings page but has a singular card with two buckets. One bucket contains all the saved properties that a user has selected and presents all the information that is present on the listing itself. The reserved bucket presents the same information but also has a small section on each property that lists when the reservation is set to take place. If there are no listings in either bucket, the cards will give an error message stating that there are either no reservations or no saved properties.

Finally, we have the user management page. This is where users can fill out additional information in order to make reservations quicker. If a user has not filled out this information, then they must fill it out whenever they try to reserve a property, so it behooves them to fill it

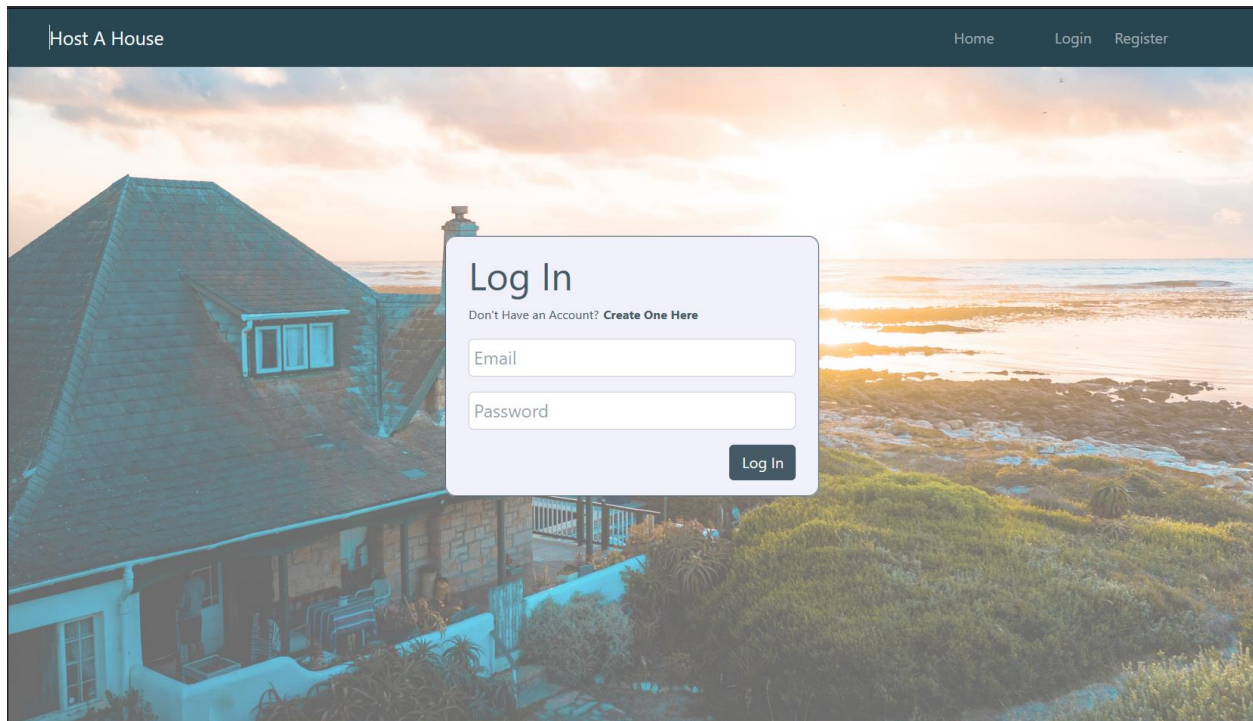
out here. The user management has fields for entering a new password and email and has an address section. There is currently no verification that the address is in fact a valid address, so it is crucial that users enter in the email and check it over.

Screen Images:

- Main Page



- Login Page



The screenshot shows the login page of a website titled "Host A House". The page features a dark teal header with the site name on the left and navigation links "Home", "Login", and "Register" on the right. The background is a scenic image of a house with a stone chimney and a balcony, set against a sunset sky over the ocean. A light purple login form is centered on the page. It has a title "Log In", a link "Don't Have an Account? Create One Here", and two input fields for "Email" and "Password". A dark teal "Log In" button is at the bottom right of the form.

Host A House

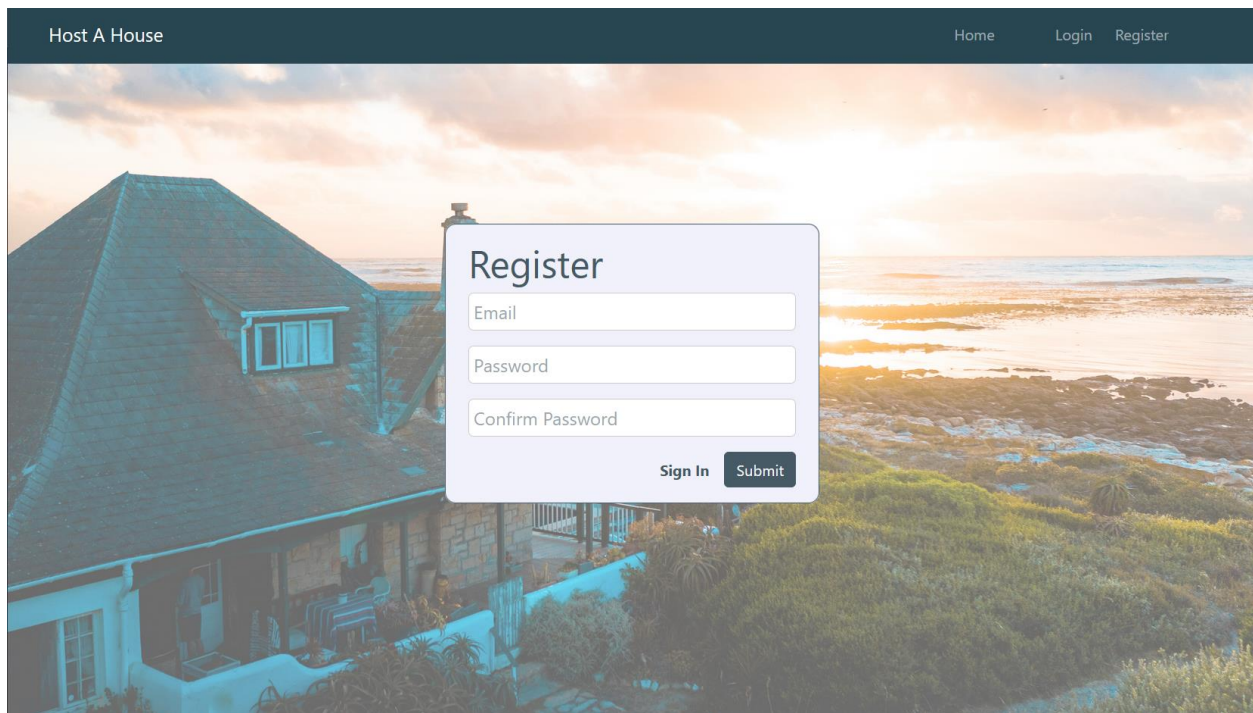
Home Login Register

Log In

Don't Have an Account? [Create One Here](#)

Log In

- Register Page



The screenshot shows the register page of the same website. The header and background image are identical to the login page. A light purple register form is centered on the page. It has a title "Register", three input fields for "Email", "Password", and "Confirm Password", and two buttons: "Sign In" and "Submit".

Host A House

Home Login Register

Register





Sign In Submit

- More details

Host A House

HomeLoginRegister

← Back to Listings


Lemon Moon Cottage


Sebastool, California 95472


Rating: 4.5


Reserve Now


Popular Amenities


 Pool

 Parking included


 Breakfast included


 Free WiFi


 Pet Friendly


 Free Airport Shuttle

Cleaning and Safety Practices


 Cleaned with disinfectant

 Hand sanitizer provided


 Personal protective equipment

 Social distancing


Explore the Area

 William P. Didusch Utolgical Museum


7 min drive

 Baltimore, MD (BWI-Baltimore Washington Intl. Thurgood Marshall

11 min drive

 Baltimore Cruise Terminal

12 min drive

 Fort McHenry

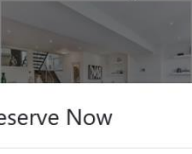
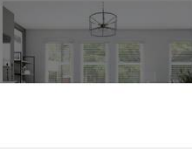
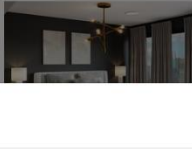
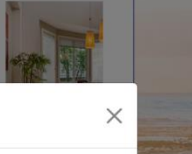
13 min drive

- Reservations Page

Host A House

HomeLoginRegister

← Back to Listings


Lemon Moon Cottage


Sebastool, California 95472


Rating: 4.5


Reserve Now


Popular Amenities


 Pool

 Parking included


 Breakfast included


 Free WiFi


 Pet Friendly


 Free Airport Shuttle

Cleaning and Safety Practices


 Cleaned with disinfectant

 Hand sanitizer provided


 Personal protective equipment

 Social distancing


Explore the Area

 William P. Didusch Utolgical Museum


7 min drive

 Baltimore, MD (BWI-Baltimore Washington Intl. Thurgood Marshall

11 min drive

 Baltimore Cruise Terminal

12 min drive

 Fort McHenry

13 min drive

Reserve Now

Check In

Month

1

▼

Day

01

▼

Year

2022

▼

Check Out

Month

1

▼

Day

01

▼

Year

2022

▼

Cancel

Confirm Reservation

- User Management

Host A House Home Hello, tt13@gmail.com

My Account

tt13@gmail.com

feee

gdfgsd

lkj;lj

jlkjkj

jijijij

Save Changes

localhost:3000/account#


- Saved Properties

Host A House Home Hello, tt13@gmail.com

Lemon Moon Cottage

Sebastool, California

Rating



Popular Amenities

- Pool
- Parking included
- Breakfast included
- Free WiFi
- Pet Friendly
- Free Airport Shuttle

Cleaning and Safety Practices

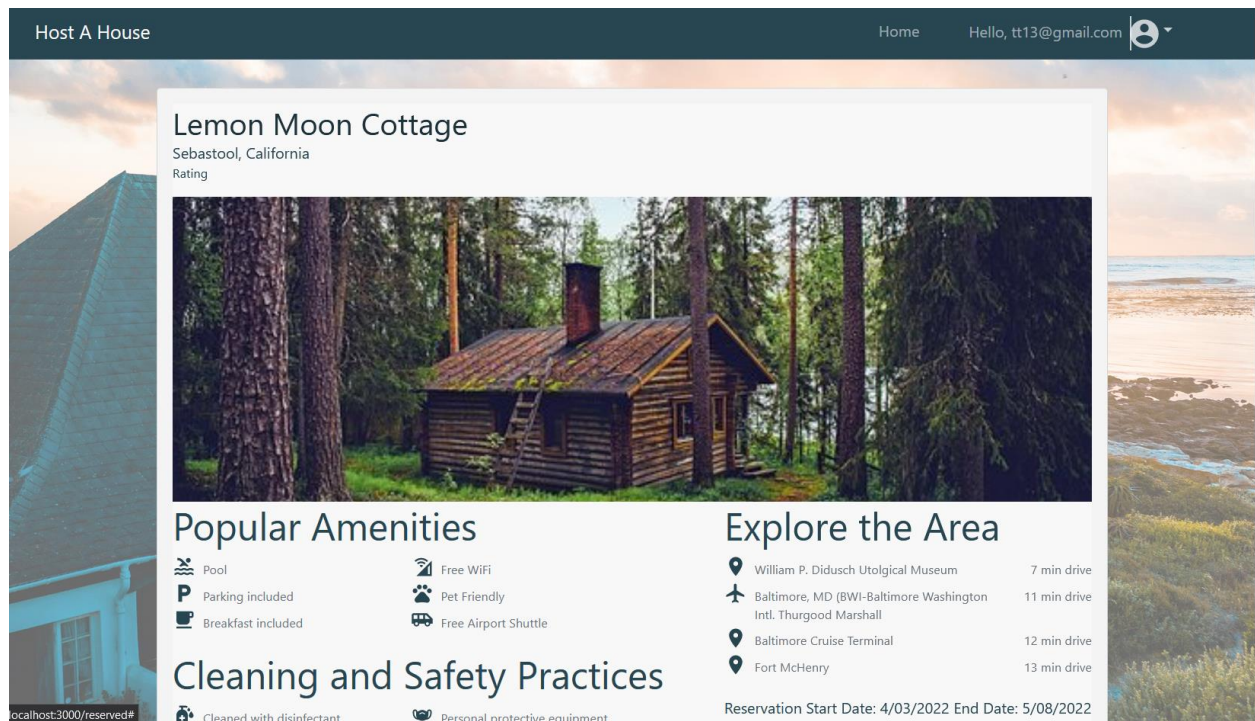
- Cleaned with disinfectant
- Personal protective equipment

Explore the Area

- William P. Didusch Utoigical Museum 7 min drive
- Baltimore, MD (BWI-Baltimore Washington Intl. Thurgood Marshall) 11 min drive
- Baltimore Cruise Terminal 12 min drive
- Fort McHenry 13 min drive

localhost:3000/saved#

- Reserved Properties



Requirements Matrix

- Searching rentals by location, date available, and price would be completed via a search bar for input which would tie into the backend to calculate and return results
- Sort rentals by location, date, price, and featured rentals will be accomplished by a drop-down menu.
- Saved/Reservation management will be done by a logged-in user to make reservations, to save properties, and to save/reserve properties this will be accomplished by logging in with a username that would compose the user's email.
- User management such as creating users and logging in/out will be accomplished by a registration form, the login and logout will be accomplished via a login, and entering a username and password and logout will be accomplished by a logout button

*See above images for visualizations

Requirement	Expected Output
Rental Properties (1)	Listings Page, Main Page
list of rental properties (1a)	Listings Page
Search rentals (1b)	Main Page, Listings Page, Search Bar
Sort rentals (1c)	Listings Page, Search Bar
Featured rentals (1d)	Main Page

User Management (2)	Account -> Account Management
Create user (2a)	Account -> Register
Login/Logout (2b)	Account -> Login
Saved Property and Reservation Management (3)	Account -> Saved/Reservation
Make reservation (3a)	Main Page, Listing Page -> reserve
Save property (3b)	Main Page, Listing Page -> save

8 Development History

Project Initiation:

- Group formed.
- General discussions of what we planned to do, and how we were going to do it

Project Plan:

- Decided what technologies to use to complete the project.
- Declared what the minimum specifications of hardware are to run the project.
- Overview of what our application is meant to do.

User Guide and Test Plan:

- Planned Features defined
- How to use the program
- General idea of how users will interface with the program

Product Design:

- Mockups of how the UI will work
- Schemas defined

Phase One:

- Login visualization done
- Register visualization done
- Backend system set up to store users
- Users are properly retrieved and logged-in (some status indicator)

Phase Two:

- Create Model for Users
- Create Model for Properties
- Create Endpoints for Retrieving Property information
- Get Login Functionality Fully working
- Populate database with pre-populated information
- Generic cleanup of files and file structure

Phase Three:

- The listings component is now sorts and searches properties dynamically
- Styling changes were made to be more in line with mockups and consistent throughout the application
- More appropriate error messages for both the login and register pages
- MongoDB is configured to more accurately reflect records in the collection

Phase Four:

- Database updates with reservations in the user and property collection
- Error checking for registration and login
- Search fixed to search dates from properties reservation array
- Reserve button functionality implemented
- Save button functionality implemented
- User account information implemented
- New design fully implemented

9 Conclusions

Suggestions for Further Improvements

In the future we'd like to optimize the way that we do user authentication and authorization. The team did a great job at making the front-end responsive, but there are still some pages that need to be redirected if a user is not signed in. A streamlining of the user authorization would also allow us to integrate the setting of user tokens more so into the backend and make automatic timeouts possible.

Another area of improvement we could make was the integration of a payment system. Users can currently save and reserve properties but there's no way to pay for a reservation. Integration of a payment system would allow us to have a truly functioning workflow from beginning to end. It would also allow us a greater level of immersion of the experience.

Key Takeaways

Caleb Elkins

This is not really a topic we explicitly covered, but the team-work aspect of the course has by far been the most beneficial and educational. Having to work together on a group project has been a very rewarding experience. This is the first time that I have worked together on a software-development project, and I feel as though it has given me some much needed, real-world experience. For instance, I had been using Git a little bit for version control on my personal projects, but now I have a much better understanding of how it works when multiple developers are contributing and how to coordinate those efforts.

As I near graduation and begin looking for work in the software industry, I think the skills I gained from working with my team will be invaluable. I was very worried before-hand since I have no

practical experience, but now I think I have a better understanding of what the actual development process can look like and how to be a functional member of a team.

Lindley Thomas

I found the group work to be the most interesting and applicable to my future. The topics in the course covered in our individual discussions were also interesting, but I found the group project the most memorable. I think that one great thing with the weekly discussions is that I added something to my list of things to investigate and read more about. I wish that the topics of the weekly discussions were something we covered in a full-length course with UMGC, because I feel like the topics covered are something that would benefit our futures, and not directly taught in many universities, or required.

Some of the main takeaways from working in a group that I found, and other things I have learned doing the project:

- GitHub
 - I have never really used GitHub before, except for random fixes and tools for my keyboards and other projects. Using it for the first time to manage a project I was working on was a great learning experience. ,I first of all, learned how to use GitHub and found GitHub Desktop super helpful. I also learned that in the future I should create as many branches as I need, at least locally. This was a painful lesson, because I undid all my changes and started over on the same branch due to bugs I was running into, just to realize that one change on my original code would have fixed everything.
- Code Styling
 - I have learned time and time again how important it is to follow styling conventions, but still tend to have jacked up tabbing in my code at the end of the day. I was doing pretty well before this class, but when I get really stressed from working all day on something, just making shots in the dark to see if it works, I tend to stop caring about style and more about it working.
- Variable naming / naming in general
 - I relearned the importance of naming in code. Again, I used to do pretty decently, at least I think, but after a long day I start to give up. I really regret this now, because I have multiple files to go over and fix naming.
- Working in a group
 - I haven't worked in a group since I left the Navy 2 years ago. It was definitely interesting to work with other people again. I was glad to find that my team members were willing to help when needed. I learned here that I need to communicate more when I am struggling with something. In a group people have

strengths and weaknesses, and if you just reach out, people are usually willing to help.

- Stepping outside my comfort zone
 - I really wanted to end up on a team that was working on a project that explicitly used languages and technologies that I learned with the CS major at UMGC. However, I am really glad I ended up working on a project involving many things that I have never learned. I am sure I lost some hair from the project, but it was extremely satisfying to jump into the dark, figure things out (maybe not in the best way) and see results with more and more things I worked on in the project actually working. I also found from this that I learn best from actually doing things and trial/error. I tried to watch a ton of tutorials at the beginning of the project but realized that none of it was sticking. I found that just researching things and trying to implement them made concepts stick in my mind, and even if what I tried did not work for what I wanted, I would remember it later for where it would actually work.

Paul Armbruster

For me, the most interesting part of this class was the group project aspect. I was not too thrilled when I learned the last class would be a group project, especially when the initial project ideas were topics or languages that I didn't have much experience in. My worries quickly went away once I started communicating with my team. Each one of my team members were quick to jump in and assist each other when it was needed. They even took the time to set up calls and work through issues so everyone could understand. This is the biggest takeaway that I envision to use in the future. Building chemistry amongst a team is a huge key to success in my opinion. Time management and deadlines are additional items that are important. When my work dictates whether or not others can proceed with theirs, it really made me reprioritize what I needed to do and plan accordingly.

Technical documentation wasn't always my strongest. From the work that my team did and the extensive feedback from the professor, it really taught me what proper documentation was and what all needed to be included. In addition, when developing code with a team, I quickly realized how important comments, naming conventions, and styling are. That's not to say I didn't use them correctly in my previous projects, but when others must review, add or use your work, they need to be able to easily understand what is happening within the code.

Troy Raines

For me the most valuable lesson for me has been teamwork, figuring out the group dynamics, what are each other's strengths? Their least skilled ability? How do you work together and figure out who fits where? The biggest element to this is communication, to not be afraid to admit that is not my comfort area, but I am good at this. Also, to coordinate who is doing what part, what the status of that particular part and what needs done as a whole.

Documentation, without documenting the steps used to accomplish one's code, or development overall, even testing then it makes it extremely hard to go back and compile all that later.

I feel that this team communicated and documented very effectively, what I enjoyed most about working with this team was the willingness to help others. Either offering to step in when someone had to work or to do family things, to helping teach a particular topic or methodology in the programming process. This to me is the most important element that occurred during this course.

I work as a security engineer, my team working with various other teams on the project and communication is the weakest area involving other individuals and teams that I have seen in both my current position and my previous ones as well. I have learned a great deal about programming, documenting the software development process and working with others in a project that deals with software development.

Khalil Savoy

Throughout this class, I gained the opportunity to work with an awesome team and a wide variety of tools to deliver a project that we could all be proud of. While I already work in the technology sector and thus was able to apply a lot of knowledge that I already had, I still learned a lot from working with my team. For example, it has been years since I've worked with the REACT framework and there are various things that have changed since I last worked with the framework that I can now take into my job. I also never got the experience of being able to work with a full MERN stack as all the projects I've worked on typically follow a more traditional JAVA backend and pure html5/JavaScript system architecture.

The two technologies listed above I will definitely be taking and applying to my professional life as a whole. Knowing REACT gives me a much wider variety of options when starting up a new project than just Angular or pure HTML/JavaScript and even when not using that specific framework, the techniques I've learned can be applied across any project I work on from here on out. The same thing applies to the use of the MERN stack.

Finally, the greatest thing I've learned and will be applying is the collaboration skills I've gained. While I use all the tools my team used in this project on a daily basis in my job, I've been working with the same team for years now. Having to build and collaborate with a team that I've known less than a week presented its own struggles, but I know that I am a lot better for it.