Storage Rents

by

Austin Alvarez
Nathan Binsky
Ryan Rich
Robert Bathalter

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology/Cybersecurity

Austin Alvarez	04/24/2022
	Date
Nathan Binsky	04/24/2022
	Date
Ryan Rich	04/24/2022
	Date
Robert Bathalter	04/24/2022
	Date
yahya M Gilany	04/26/2022
Advisor	Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services
April 2022

Table of Contents

Tables and Figures

Figures

Tables

Abstract

Storage Rents is a web application that provides users with an alternative for local storage services. Local storage is not always cheap and can be excessive; the program is a solution to this issue. Storage Rents aims to provide users with the ability to search for storage spaces and connect with the owners. The application also allows users to list their own storage spaces and connect them with interested renters. Storage Rents is user driven, with a focus on keeping things local for clients. Along with that, Storage Rents aims to give users an intuitive renting experience that fits their needs. If someone ever finds themselves in need of storage, or they have excess storage; Storage Rents will be there.

Introduction

Project Summary:

Storage Rents is an alternative solution to the current storage market. Users will be able to put their spaces up for rent and rent other users' spaces for storage purposes. This is meant to be a cheaper, community driven alternative to typical storage solutions.

Problem Statement:

External storage is expensive and can have a lot of unfilled space that makes using it cost ineffective. With research, it was found that the national average monthly price on storage is $88.85 (Neighbor, 2020). This price also varies depending on a unit's square footage, with the average price per square foot being $0.96 (Neighbor, 2020). It was found that the average annual budget for utilities and things like repairs and other miscellaneous costs in an average American household is $7,068 (Price, 2021). With these statistics, one can infer that on average, a storage unit will cost a household $1,066.20 a year, 15% of the average American utilities budget. This large price point leaves many households choosing to forgo storage units in favor of their own garages, with only 9.4% of U.S households utilizing units (Neighbor, 2020). Along with that, these units still manage an average occupancy of 92%, leaving some people with no options even if they can afford it (Neighbor, 2020). Utilizing a local and collaborative platform, more people will have more options, regardless of household wealth. Clients with excessive storage will be able to list and price their space for users to rent, giving them the opportunity to earn money just for having excess space.

Solution:

Storage Rents is an Airbnb-like storage service for those looking for cheap storage options, as well as those with extra space. It is a cheap alternative to storage units and offers less space for those who don't need the entirety of a storage unit because users don't pay for a whole storage unit, just what they need. This solution allows users to browse the list of clients with storage available for rent. There, they can view pricing and information about the client as well as their offered space. If the user is interested in renting the space, they can then contact the client via phone or email. Price negotiation is dependent on the client listing the storage space.

Project Source:

The idea of Storage Rents was conceived in collaboration by the team as way to provide a service that for a common problem that didn't have many solution options. To provide a service for people to have access to external storage at a cheaper cost than storage units. At full deployment, it is hopeful that users will interact with each other, using Storage Rents as a middleman connection establishment.

The team was created by a group of men who met in college and wanted to create something that could be used by people across Ohio, and possibly the United States.

Discussion

Project Objectives/Goals:

The project's goal is to offer a storage service that will allow users to store items at a cheaper rate and allow them to offer storage options with their free space. This service will provide Storage Rents and users offering storage profit. Functionality details are as follows:

- Users can find storage options through Storage Rents. Users will list details and specifications of the listed space.
- Users can make and update a profile for their account.
- Users can view and contact locations near them, users will provide their own contact information.
- Users can post and edit locations they have access to.
- Users have support from the development team and can have their problems addressed if they arise.
- Users can view the distance from their location on a map, propagating from user submitted addresses.

Project Scope:

The team will develop a web application that enables users to store and offer storage to other users. This will be done in an Airbnb-like format that is user friendly and easy to navigate. The system will not be responsible for the physical space and location used, along with any objects being stored in said locations. The team aims to deliver a new option for average people with storage needs, providing a platform for household storage through crowdsourcing. The application will have two sides: space owners listing their storage space for rent, and renters renting space to store their possessions. Space owners will be able to list all relevant details pertaining to their space, including size, condition, etc. Renters will be able to search locally through various options, finding the best fit for their needs. Some restrictions to the projects development are: the team's schedules, as each member will be working as part of a CO-OP during the development of the application. In person team meeting are also less of an option for the team due to the ongoing Covid pandemic.

Quick Project Timeline:

The following timeline will show the current expectations regarding deliverables and their fulfillment.

Task #	Task Name	Duration	Start Date	End Date
1	Research/Scope Definition	2 weeks	9/13	9/27
2	Creating Framework/ project skeleton Database framework will be done	2 weeks	9/27	10/11
3	Both User-logins will be functional	1 week	10/11	10/18

4	Complete basic frame of user features	2 weeks	10/18	11/1
5	Complete basic framework for business users	2 weeks~	11/1	11/15
6	Buffer weeks for project timeline	2 weeks	11/15	11/29
7	Finish front end	2 weeks~	11/29	12/13
8	Refine framework of users	2 weeks	12/13	12/27
9	Refine framework of business users	2 weeks	12/27	1/10
10	Final Build for testing	2 weeks	1/10	1/24
11	Buffer weeks for project timeline	2 weeks	1/24	2/7
12	QA-Testing and troubleshooting (Can also take from this week for other sections)	4 weeks	2/7	3/7
13	Presentation Rehearsal	1 week	3/7	3/14
14	IT EXPO Final Prep	4 weeks	3/14	4/12

Table 1 Project Timeline

Technologies Used:

The technologies used in the creation of Storage Rents included different software in development and users will utilize the frontend directly and the backend indirectly.

- Backend: Node.js, Express, PostgreSQL (database), JSON Web Token
- Frontend: ReactJS
- Misc.: GitHub, Amazon Web Services, Postman, VSCode, Security Onion

These technologies were selected for their cost, familiarity, and their capabilities. Every technology in the PERN Stack is open source and free to use, allowing us to build most of a full-stack web application with no cost. Cost becomes a concern when it comes to hosting our database, utilizing Amazon Web Services to host our infrastructure keeps the costs low, and includes various useful services like security benchmarks. Security Onion was used as well because it is open source and it can monitor the network and how the web application interacts with the online world.

Technical Architecture Diagram:

The following diagram details Storage Rents' technical makeup, the applied color coding refers to separated sections that make up the overall architecture: blue is the client section, purple is the server section, and green is the administrative section. From the client layer, users will access the application through their mobile browsers, web browsers, or a specific client application if applicable. The client interaction with the application is stood up using ReactJS. From there, users will interact with the server through the web service API. Through this, users will access the application's action service and pull requested data from the database. Express along with a little Node.js make up the middle layer of technology servicing these requests. Database requests with then be handled with PostgreSQL to grab the requested data. The server layer encompassing the API, application, and database will be managed using applicable admin tools. This will be done to ensure proper use of the application and conduct proper administration.

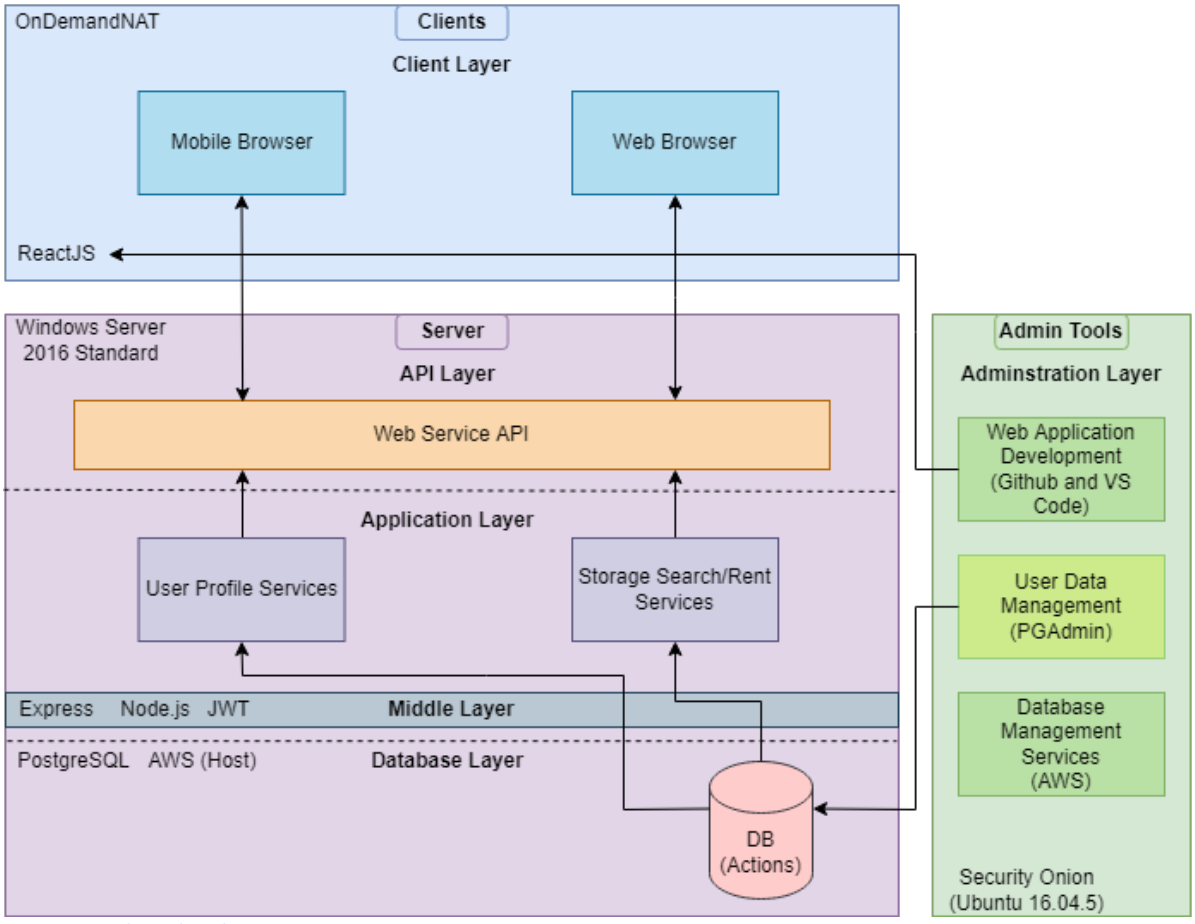

Figure 1 Technical Architecture Diagram

User Personas:

User Persona: 1	
	Title Stay-at-home Mom
	Name Jane Smith
	Age 35
	Gender Female
Behavior	Domestic
Pain	Cluttered household
Needs & Goals	Wants to declutter and needs more room for her things

Table 2 User Persona 1

User Persona: 2	
	Title Accountant
	Name John Doe
	Age 32
	Gender Male
Behavior	Workaholic
Pain	Stress and finances
Needs & Goals	Financial Gain, utilize empty basement, bring in additional income

Table 3 User Persona 2

Use Cases:

Use Case ID	01
Use Case Name	User creates an account
End Objective	User can login to account
User/Actor	John Doe
Trigger	Status check
Frequency of Use	Frequent, every time user goes to application
Preconditions	Can access Storage Rents application
Basic Flow	Access Storage Rents application, register account, login to account
Alternate Flow	Access storage rents application, have an account already, login to account
Postconditions	Have access to all information utilized by the web application

Table 4 Use Case 1

Use Case ID	02
Use Case Name	Upload new Storage Rents Locations
End Objective	Create a new entry on the site for users to view
User/Actor	John Doe
Trigger	Wants to find a use for excess storage space
Frequency of Use	1+/infrequently
Preconditions	Owner must have storage space and an account already setup on the site
Basic Flow	Owner logs into site Owner navigates to the storage page to input storage for rent Owner adds relevant info of storage for rent Owner submits
Alternate Flow	Owner creates new account Owner navigates to upload storage for rent Owner adds relevant info of storage for rent Owner submits
Postconditions	Owner must have a valid account and valid storage rent submission

Table 5 Use Case 2

Use Case ID	03
Use Case Name	View locations available to rent
End Objective	View locations available to store
User/Actor	Jane Smith
Trigger	View locations available to rent within the application
Frequency of Use	Weekly
Preconditions	Have internet connection
Basic Flow	Access webpage, hit location tab, view locations
Alternate Flow	Access webpage, hit view locations button on home page, view locations
Postconditions	Make/have an account

Table 6 Use Case 3

Use Case ID	04
Use Case Name	Search for locations nearby
End Objective	View locations within a nearby area based on search criteria
User/Actor	Jane Smith
Trigger	Search for nearby locations on dashboard
Frequency of Use	Weekly
Preconditions	Have internet connection
Basic Flow	Access webpage, Navigate to homepage/dashboard, Click on search nearby.
Alternate Flow	Access webpage, Navigate to homepage/dashboard, Enter search criteria.
Postconditions	Have access to information returned by the website.

Table 7 Use Case 4

Use Case Diagram:

The following diagram describes Storage Rents' various use cases, with the aim of giving a preconceived view of how a user would use Storage Rents. The diagram starts with the general user, which is then broken into three separate groupings, new users, users looking for space to rent, and users with space to rent out. From there, each group has lines connecting them to the various uses of Storage Rents that they would utilize. For instance, the average renter using Storage Rents will be able to log in and search for storage space.

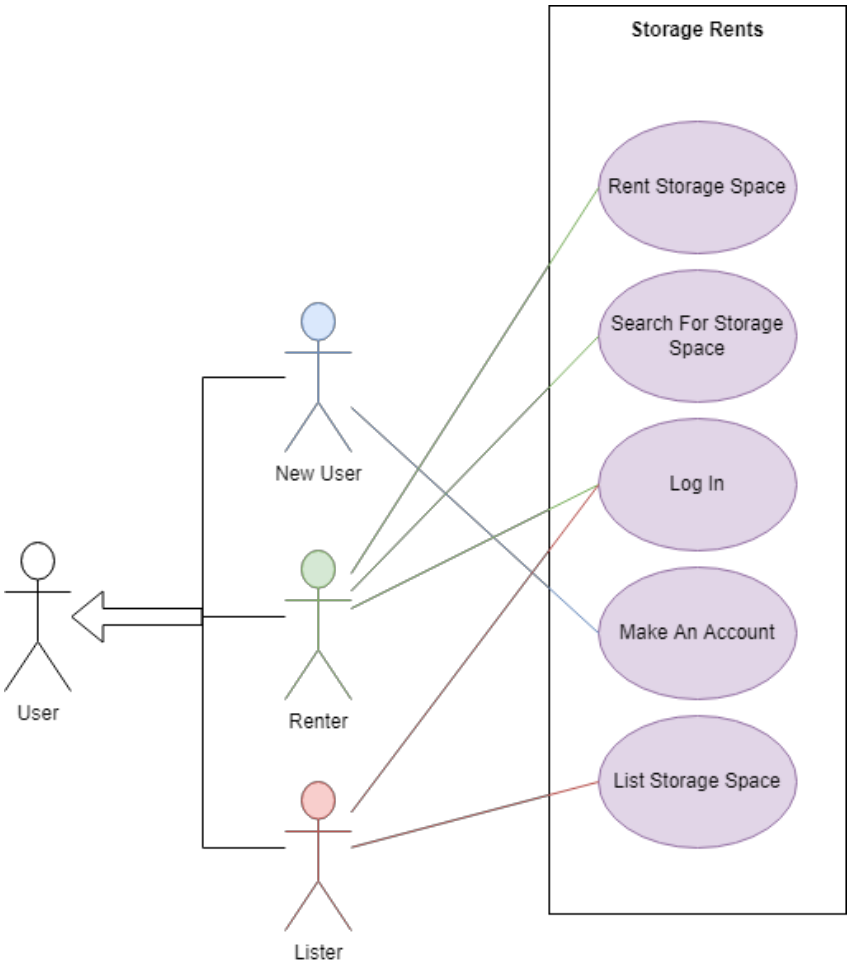

Figure 2 Use Case Diagram

Testing Plan:

Overview

This testing section is a general rundown of what and how frequently the team will be testing on code pushes as the project is developed as well as after the project is functional. The focus will be on functionality and what the user should be able to do and what the user can actually do. The team will go through a set of guidelines to test the functionality of each part of the application to be thorough and ensure that there aren't any bugs or glitches with typical use.

Methodology

The team's testing approach is to test important functionality on the website that is required by the main use cases. The focus of this is to make sure each proposed function works as intended. Primary testing will be performed on user accounts to ensure the team knows how a user can affect the application. This user focused testing approach is to make sure the application is preforming correctly on all user accounts. Additionally, as the project moves forward, the use cases and testing will be refined as the scope is narrowed to the current functionality of the site. The testing will be done manually after every major push/update on the system. If while working on the system after a push there is a bug, the system will need to be retested. After the completion of main user acceptance testing, the team can move on to test different aspects of the application that a user wouldn't in the search for errors. Each test will be documented and then reviewed by the team to see where things went right or wrong.

Scope

Storage Rents aims to provide users with several different useful features when it comes to renting storage space. The goal is to make renting local space easier by giving users a platform to find and share storage space. With that in mind, Storage Rents needs to fulfil certain requirements and criteria for a user to utilize the web application without run into any problems. Following the testing plan will assist in discovering these issues.

Use case/features:

Testing will ensure that users can access and use the application and its tools to make an account, make and view locations, and see them in a visual form as well.

- Be able to create an account
- Be able to add a new storage rents entry for an existing account
- Be able to modify a user's existing storage rent entry
- Be able to navigate functionality without site/widgets breaking.

Objectives

The overall goal of Storage Rents is to provide an alternative solution to local storage needs, whilst also ensuring ease of operation and user security. To that end, Storage Rents must meet certain objectives.

- All major features and use cases need to be accounted for (Ability to search for storage, list storage, contact owners/renters, etc.)
- All use cases must account for all the user roles
- All bugs need to be resolved before IT Expo
- Features need to be working consistently across user roles.
- End goal is that a user who has not been to the site, will be able to navigate the functionality of the site and be able to create a new account. After having an account, they will be able to add/apply for storage locations.

Test Logs and Procedures

Item#	1
Test Case#	1
User	new user/ John Doe
Role	User (Space Owner)
Expected Output	user creation success
Actual Output	user creation success
Pass/Fail	pass
Reason for Failure/Success	User was able to create a new account
Date	11/5/2021

Table 8 Test Log 1

Item#	2
Test Case#	2
User	existing user/John Doe
Role	User (Owner)
Expected Output	Creation of listing
Actual Output	Listing is successfully posted
Pass/Fail	pass
Reason for Failure/Success	User was able to create a new storage entry
Date	11/9/2021

Table 9 Test Log 2

Item#	3
Test Case#	3
User	new user/ Jane Smith
Role	User (Renter)
Expected Output	Found locations in her area
Actual Output	Found locations in her area
Pass/Fail	pass
Reason for Failure/Success	User was able to find locations in her area

Date	12/15/2021

Table 10 Test Log 3

Item#	4
Test Case#	4
User	new user/ Jane Smith
Role	User (Renter)
Expected Output	View a specific location's information
Actual Output	Viewed a specific location's information
Pass/Fail	pass
Reason for Failure/Success	User was able to see a specific location's information
Date	12/29/2021

Table 111 Test Log 4

Item#	5
Test Case#	5
User	new user/ John Doe
Role	User (Space Owner)
Expected Output	View other locations
Actual Output	Viewed other locations
Pass/Fail	pass
Reason for Failure/Success	User was able to see other locations
Date	1/19/2022

Table 12 Test Log 5

Item#	6
Test Case#	6
User	new user/ John Doe
Role	User (Space Owner)
Expected Output	View locations on a map
Actual Output	Viewed locations on a map
Pass/Fail	pass

Reason for Failure/Success	User was able to see locations on a map
Date	3/20/2022

Table 12 Test Log 6

Item#	7
Test Case#	7
User	new user/ Jane Smith
Role	User (Renter)
Expected Output	View locations on a map
Actual Output	Viewed locations on a map
Pass/Fail	pass
Reason for Failure/Success	User was able to see locations on a map
Date	3/20/2022

Table 14 Test Log 7

Testing Review

Testing concluded with a realization of certain aspects of custom input fields not functioning as intended. This has caused the project to undergo minor changes, like routing changes and changing the code to how we want it to work, to allow for these fixes. Additionally, there were parts of the project in a nonfunctioning state that will need to be returned to for further testing, like map routing to pages and the look of the dashboard.

Change Management Plan:

Any change that will affect the project will need to be discussed amongst team members. To keep things impartial, all team members will be able to request changes, as well as review them. Major changes, like changing functionality, to the project's code will be reviewed before they are pushed to GitHub, major changes should include proper write-ups of expected results. This process will ensure all voices are heard and the project is going in a way that can be agreed upon.

The triaging process will see these changes sorted into separate classes depending on their severity. Class 1 changes are large and have a heavy impact on the product's development, these changes will require write-ups and approval by at least one other team member. Class 1 changes include removal of current components, incorporation of unplanned features, etc. Class 2 changes have a medium impact on development and will only require write-ups. Class 2 changes include the implementation of planned features, implementation of new components, etc. Class 3 changes are minor and require no review or writeup. Class 3 changes include changes to styling, minor bug fixes, etc. Class 1 changes take priority in development, followed by Class 2, and then Class 3. Stake holders will be notified and walked through any planned changes if they will affect the overall product of development.

Budget:

Once the project is ready for deployment, the most viable option is to go with a web hosting service. The price is heavily dependent on what service chosen and the required specs of servers. Given the size of the project, the cost can be kept low. Bluehost is a web hosting service, with $5.45/mo for hosting plus package extras making the total at $137.16/year. This will go into the software costs. Hardware will cost $4,000.00 for computers for the development process. The annual cost of $500.00 is for repairs and anything the team might need to replace for the server. Miscellaneous is for transportation, $80/month, and internet fees, $50/month per person which comes out to $200/month.

Table 135 Project Budget

				Ongoing Annual		
	Rate Per/Hr	**Work Effort (Hours)**	**1 X Costs**	**Rate Per/Hr**	**Work Effort (Hours)**	**1 X Support Cost**
Labor - IT	20	Total Hours – 1440 (4 People)	28000 $	20	0	$ -
Labor - External	10	Location Setup – 20(4x5)	200 $		0	$ -
Software - External			PERN Stack $0 Web Hosting (BlueHost) – $11.45/mo			Support/Repairs – $500.00
Hardware - External			Computers – $3,999.00			Transportation – $960.00 ($80/mo)
Misc.			Internet – $200.00 ($50 Per)			Internet- $2400 ($200/mo)
TOTAL			33,210.45 $			3,997.16 $

Problems Encountered and Analysis of Problems Solved:

The first major problem encountered was the team's required restart from HTML to React and the rest of the PERN stack. Originally, the front end for Storage Rents was put together entirely with HTML. This would not work, as the app would have no way to communicate with back-end infrastructure with just HTML. To that end the team had to scrap our first iteration and start from nothing using React as the front end. This led to a delay of development initally, as each team member was introduced to some sort of new technology within the PERN stack. The restart also put the team about a week behind, requiring additional work from team members to catch up. Each member was required to go through research and learning processes to understand and interact with the new tech, causing the overall development of the app to slow at times. The team also struggled with early communication issues. The team had trouble finding an optimal timeframe to meet, as all members had their own busy schedules. Along with that, the team week

to week planning was less than ideal, leaving actual tasks poorly planned out and causing some project inconsistencies. Expectations for the final product were also vague at best. A solid idea was in place, but reaching that ideal was difficult and left team members wondering who should do what and when. The team addressed these issues by implementing a triaging process and a new time management tool in GitHub.

Conclusion

While there were struggles, each member of the team either developed or enhanced their application development skills with the PERN stack. It has been found that solving issues and bugs takes dedication and hard work. The experience in dealing with these challenging tasks will carry over and motivate the group during future development.

For the sake of time and priority for functionality these project features were pushed back on the timeline: The system currently does not have a functional login. Because of the lack of login features, the application presently only has 1 user. This means that anyone that accesses the application is currently updating/adding the same user info.
The contact system pulls the email associated to the storage entry listed on the site for contact and directs you to that email. Ideally the system would be able to connect storage owners through the application with the users. Finally, the filtering and searching of data aims to be improved upon. At present the application pulls the available data entered on the system and displays it on tables and maps.

References

Neighbor (2020, December 15). *Self storage Industry Statistics*. The Neighbor Blog. https://www.neighbor.com/storage-blog/self-storage-industry-statistics

Price, S. (2021, April 26). *Average household budget*. ValuePenguin. https://www.valuepenguin.com/average-household-budget