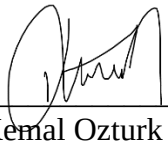
LinkTailor Final Report

by

Daniel Hickman, Kemal Ozturk, and Lauren Tillery

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

Daniel Hickman — 4/12/2021
Daniel Hickman — Date



4/12/2021
Kemal Ozturk — Date

Lauren Tillery — 4/12/2021
Lauren Tillery — Date

Bander Alyami — 4/12/2021
Bander Alyami, Faculty Advisor — Date

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

November 2020

Table of Contents

Table of Contents i
List of Figures ii
Project Introduction 2
Problem 2
Solution 2
Project Goal 3
Overview 3
Discussion 4
Project Concept 4
Design Objectives 5
Methodology 10
User Profile 14
Use Case Diagram 18
Technical Discussion 19
Testing 21
Budget 25
Timeline 27
Analysis of Problems Encountered 36
Recommendations for Improvement 37
Conclusion 38
Appendix A. Code 39
The Vuex Store 39
Windows Context Menu – Registry Integration 40
Saving Images 41
Windows Installer 41
Obtaining unique IDs 43
Fetching Website Icons 43
Appendix B. Branding 45
Logo 45
Website 46
Poster 48
ReadMe 49
The App Interface 50
References 53

List of Figures

1 Kanban Backlog and Project Buckets 8
2 Team Kanban Board (from November 2020) 12
3 Sprint Boards (which summarize each Sprint) 13
4 Use Case Diagram for LinkTailor 18
5 Diagram of Technologies used by LinkTailor 21
6 User Testing Form screenshot of the “Zoom out” task 23
7 Original Project Timeline 29

Abstract

LinkTailor is a virtual “wardrobe” where instead of arranging clothes, one can arrange links (to applications, files, and websites), treating them as garments that can be styled, tagged, and organized into outfits for any occasion. Current link-management solutions lack grid layouts, mass-opening of links, and user customization. So LinkTailor was developed to give users flexibility for their own workflow. LinkTailor can be installed on both Windows and Mac, being coded within Electron and thus with web languages and utilizes the Vue.js framework. LinkTailor even sports integration with the Windows context menu. The final product allows users to save links in grid-based layouts, re-arrange with drag-and-drop, and assign custom tags. Development is continuing as the LinkTailor team continues to add additional conveniences.

Project Introduction

Problem

Keeping track of browser bookmarks, desktop icons, file system shortcuts and other "quick links" can be a frustrating task for computer users who use many links a day[i] and frequently add new ones.[ii] Desktops get cluttered with files and documents, bookmark bars overflow with website links, and browser windows cram full of open tabs.[iii] This is an increasingly relevant concern as a growing number of computer users[iv] are accessing more websites and applications on a daily basis[v]. Current solutions such as Raindrop.io for bookmarks, Pocket for articles, and the Windows Start menu for applications are helpful, yet many users desire additional capabilities,[vi] feeling their current link-managing options are restrictive, tedious, frustrating, and sub-optimal for their desired workflow.

Solution

LinkTailor is a desktop application that allows users to save links to websites, documents, files, and applications for later access, adding them to layouts, tagging them, and organizing them spatially. LinkTailor can be used as a supplement to current link managing options such as the Windows Start Menu and Chrome Bookmarks. LinkTailor is a flexible solution wherein the grid-based layouts allow for users to arrange links as they prefer, opening the door to many use-cases: layouts for work projects, organizing one's personal file system, sharing resources with friends, creating a tutorial from web resources, organizing one's favorite music or podcasts, storing all the important documents and websites for an academic course, and more. Conveniences like having an "undo" (Ctrl+Z) functionality and being able to open all the links of a layout with a single click goes a long way in increasing the usability of LinkTailor.

Project Goal

The goal of LinkTailor is to improve the user experience of managing bookmarks, application links, and file system items. The application allows for complete customization to suit the user's needs. Resizable links, tagging, multiple layouts for various categories of links, and styling are just some of the features LinkTailor has to offer. Non-computer savvy users can use the simple features just to sort and resize their links, whereas users who are computer savvy are able to enhance their links with all of the great customizable features available to them. LinkTailor can be the easiest and most reliable way for users to manage their links.

Overview

This report summarizes the goals and uses of the LinkTailor application. First discussed is the project concept and the inspiration for LinkTailor. Then the team's design objectives with the guidelines for designing the UX as well as scope-management for app development. Following that the project's Agile methodology is discussed and the deliverables of user profiles and use-case diagram are showcased. After that, the technical aspects of LinkTailor are discussed followed by the user testing and budget planning. Finally, there is a look through the project timeline and an analysis of problems the team encountered with recommendations for future improvement. The Appendices hold technical code and screenshots for the branding and of LinkTailor and the UI of the final product.

Discussion

Project Concept

The LinkTailor application can be thought of as a wardrobe for storing various garments which the user can select for the proper occasion. Just like clothes are organized spatially within a wardrobe, links can be organized spatially within LinkTailor. Whole outfits can be stored together for quick opening of multiple links, garments can be tailored or styled for a better appearance, and the latest fashion trends can be shared with other users to inspire their own link-management style.

This initial idea for the project was conceived by Daniel Hickman, who found himself suffering from an over-abundance of bookmarks – more than 300 – sorted into various folders such as “Games,” “Technology,” “Reading,” “Podcasts,” “Music,” “Work,” “Recipes,” “D&D,” and “Return To.” Finding and organizing saved bookmarks was becoming an increasingly difficult and frustrating task. While investigating possible solutions such as Tagpacker, Pocket, Dewey bookmarks, Dragdis, Dropmark, and Station, all were found significantly lacking. None of these possible solutions allowed for easy adding of links to a spatially organized and customizable layout. Daniel did benefit from his research by using RocketDock and the Windows Start menu for better organization of file-system links and applications, but neither provided any improvement to the original concern of bookmark management.

Thus, the concept for an easy way to organize not only bookmarks, but also file system files and folders and application links was born! Daniel found two other Software Development majors who were passionate about the project idea, Kemal and Lauren, and together they brainstormed ways to create an incredible app using the exciting new technologies of Vue.js and Electron.

Design Objectives

Below is a list of the high-level values that have guided the team's design choices for LinkTailor. Then is discussed how specific app features were prioritized by the team and re-assessed in Sprints throughout the development of LinkTailor.

LinkTailor UX Considerations

Below is a list of values that have guided the team's design choices, with priority given to the higher-listed items:

Functionality: Links can be added and organized.

- Web-links, files, folders, and application links can all be added and arranged spatially.
- Next-level functionality is bulk saving and opening links, setting links to run in a particular program, as admin, or with other specifications.

Convenience: Quick to use.

- Measured by how many keypresses or clicks it takes to perform an action such as adding a link. Minimize the steps it takes!
- This may mean creating browser extensions or Windows-integrations.

Aesthetics: The app looks good.

- Measured by user testing. Users will look at their layouts a lot, so the app must be aesthetically pleasing.

Guidance: Helps build good workflow habits.

- Measured by how little users have to change or think about for the app to be useful.
- Users who will not invest much time in the app must be given the best link-managing options by default and helped with suggestions (or default templates) of how to tag or what kinds of layouts or widgets to create.

Freedom: Users are able to customize their experience.

- Measured by users being able to change the look and feel of their layouts easily. This might be done through allowing many variables to be changed by users very easily. This may also be done through providing more advanced users with the ability to craft and publish various designs.

Sharing: The ability to share design themes and link layouts with others.

- Measured by how easy and intuitive it is to export and import layouts.

Website: A space for users to share their layouts and download what others have created.

- Will promote community innovation around the app.
- If successful, new users will be able to browse and download beautiful themes and layouts.

Data Recovery: If a user's PC is irrecoverable, they can load their link layouts on their new device.

- The first level is allowing for the importing and exporting of layout files within the app.
- The second level is creating an auto-back-up feature.

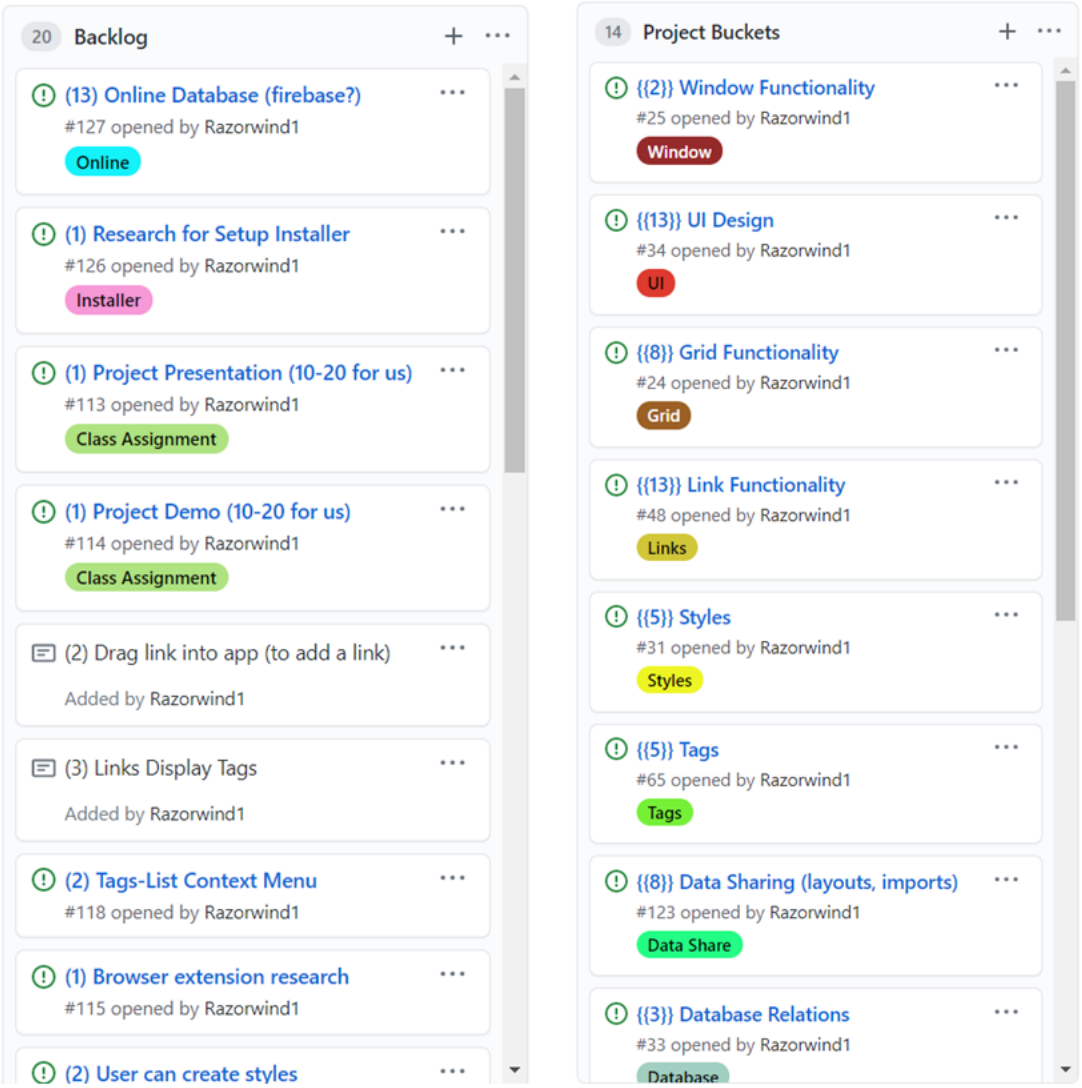
Device Syncing: A user's change to a layout on one device will update a synced layout on another device.

LinkTailor Scope Prioritizing

The team established bi-weekly Sprint Planning sessions specifically for assessing priorities. As time went on the team found it most effective to simply update the Kanban board with any new understanding of priorities whenever those understandings were articulated in any kind of meeting.

Daniel updated the Kanban board "Backlog" column, usually multiple times a week, keeping the most high-priority and relevant tasks at the top. He also created a "Project Buckets" column with categories of app features, keeping the top priority categories at the top and tagging all tasks with the relevant buckets to help track how much work was being put into which project buckets. This increased how much the team thought about prioritization and made it more clear which work was being done, helping the team assess how much they were sticking to the priorities.

Below are two columns from the LinkTailor Kanban board: Backlog and Product Buckets which were both used to assess and define priorities and scope.



1 Kanban Backlog and Project Buckets

The initial list of app features, in order of priority was:

- Layouts with spatial organization and drag-and-drop functionality
- Tagging functionality with stylistic differences based on tags
- Link Searching
- Widgets with custom locations and ability to summon based on mouse movements
- Extra options for opening links (admin, open with…)

- Themes & detailed customizations (color/shape/size)
- Custom hotkeys
- Importing and exporting link layouts
- Chrome extension (for easier web-link adding and accessing)
- Windows-integration (adding “New link…” to the right-click menu)
- Detailed searching filters
- Syncing between devices
- Mobile compatibility (beginning with Android)
- An easy way for others to contribute themes, customizations, etc. (& share them with others)
- Custom notifications, rotating messages, reminders, and triggers

It did not take long for the team to start making changes. One of the first features slated to develop – categories of links – turned out to require a lot of investment into the grid-system being used, and there was uncertainty as to whether the current grid library was sufficient for the long-term needs of the app. To avoid having to re-do work upon a likely grid library change, the team chose to halt any “non-core” work on the grid. Link searching and multiple layouts were also judged to not be “core functionality” and relatively easy things to implement, and thus they pushed to the following semester.

This turned out to be a highly successful approach. The team was able to focus on core functionality, and indeed was able to implement link searching and multiple layouts relatively quickly at the end of the second semester of development, just as hypothesized.

Methodology

The team chose to implement an Agile methodology for project management. Frequent communication was the number one priority. This was largely accomplished through 10-minute virtual "Standup" meetings every Monday, Wednesday, and Friday. During these meetings, code was shared, problems expressed, new meetings scheduled, and life updates conveyed. This boosted team cohesion and helped each team member to prioritize their work better based off what others were working on.

Communication

- ✓ Weekly 3-hour Monday night virtual meetings for class
- ✓ 10-minute virtual meetings Monday, Wednesday, and Friday
- ✓ GroupMe text communication, frequently used
- ✓ Code Review and Retrospective meeting every two weeks at the end of a Sprint
- ✓ Code Sharing and Troubleshooting meetings spontaneously scheduled as helpful

Kanban Board

The team used a Kanban board to…

A) Conceptually chunk work into **small tasks**, making it easier for team members to pick up new tasks and convey the work they accomplished. New tasks were added every week.

B) **Prioritize** work by re-ordering and re-defining tasks on the board whenever new insights were discovered or feature priorities adjusted. Tasks were adjusted by the project manager multiple times a week, often real-time during meetings when new features were discussed.

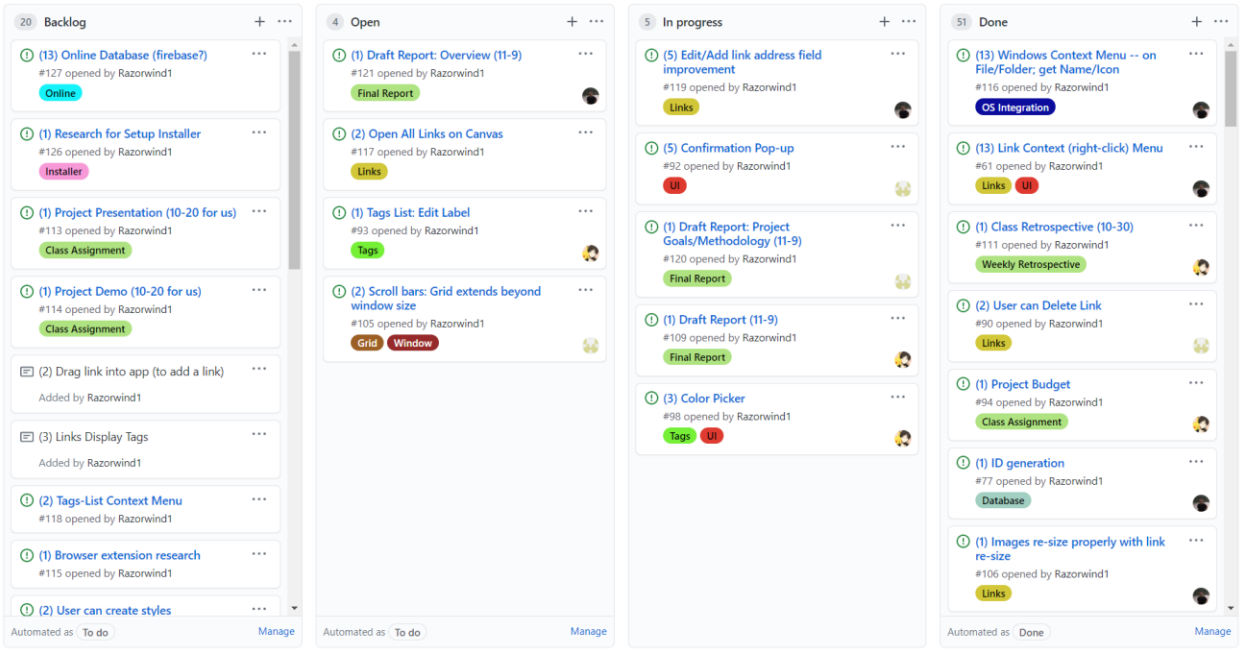
C) Track **velocity** (the speed at which work is being completed). Each task was assigned a velocity number (more often known as "story points") representing a mix of the amount of value it

provided to the application and the amount of work it took to complete. These numbers could be adjusted after the task was completed to better reflect the work effort and value provided. Then the velocity numbers for every task in a Sprint were added together so work efficiency and progress could be assessed over time.

D) Create work **transparency**. The team and the advisors and anyone really (since the Kanban board is public) can clearly what work was completed when and by whom. This is in line with the team's value of honesty and helps hold the team members accountable for doing work. If work is not being completed, it is a visible concern, encouraging the team to address that issue.

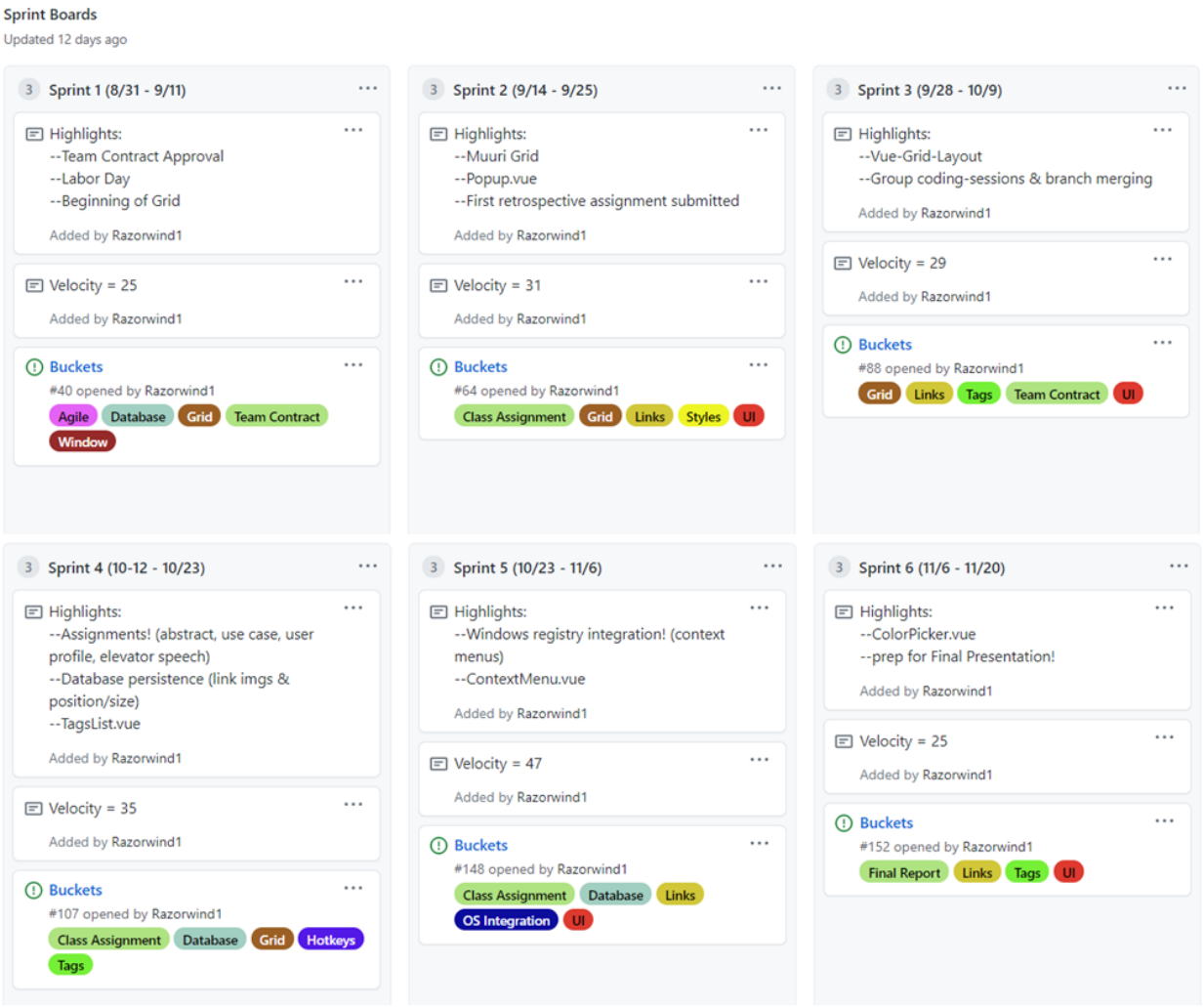
E) **Document** the work completed. The Kanban board is a record of the LinkTailor project which tells a story of the team's progression, thought processes, successes, and challenges. This assists in the team's reflection on the project, helping them learn from the project and articulate stories and lessons from the project. It is also an uplifting experience to look back on completed work with appreciation and pride.

Below is the Team's Kanban board from November of 2020, showing the Backlog, Open, In Progress, and Done columns.



2 Team Kanban Board (from November 2020)

The team worked in 2-week Sprints and summarized each one in a “Sprint Board.” This was an idea Daniel had for documenting Sprints, viewing velocity across the project, and for having intentional reflection on the over-arching story of creating LinkTailor. Several Sprint Boards are shown below.



3 Sprint Boards (which summarize each Sprint)

Values

The team articulated the following values in their team contract. These are helpful values to re-visit to renew the mutual desire for a positive group dynamic and successful final project.

Working Software
Working hard to create a satisfying product.

Communication
Frequent, effective, and honest communication.

Empathy
Assuming the best of one another, treating each other with respect and care.

Integrity
Doing what is right, avoiding deceit, and owning up to mistakes.

Celebrating the Try
Our efforts are worthy of recognition and celebration!

User Profile

The following user profile indicates three categories of users which will interact with LinkTailor. First is the **developers/system administrators** who will manage the application over time. The second is the **computer-savvy users** who are familiar with application interfaces and have plenty of apps, files, and websites that they wish to manage in specific ways for their daily workflows. Third is the **computer-inept users** who do not have much computer intuition and may not have a lot of links to manage, but still want an easy way to access their everyday applications, files, and websites.

Form 1: Developers/system administrators

Developers/system administrators

Application:

GitHub, Visual Studio Code, JavaScript, HTML, CSS, SQLite, NPM, Vue.js, Electron

Potential Users: Developers/system administrators

Software and Interface Experience:

The user should be familiar with JavaScript, HTML, and CSS which are used in the creation of the application design and functionality. The user should also be familiar with Vue.js, Electron, and Github for the coding of the LinkTailor app. Familiarity with file system management and online syncing will also be helpful.

Experience with Similar Applications:

The user should have used code management software like GitHub, coding software like Visual Studio Code, and have familiarity with the command line, dev tools, and web development languages. Familiarity with well-designed computer apps and other link-managing solutions is also very helpful for inspiration in designing the LinkTailor app.

Task Experience:

Should have experience in the items listed above for the creation and proper maintenance of LinkTailor.

Frequency of Use:

When the application is created this user will interact with the software shown above, but only when needed. Overtime, the user will interact with the application after its creation to manage any bugs that are discovered after launch and provide any updates to features based on feedback of the end user.

Key Interface Design Requirements that the Profile Suggests:

This user will need to have a good understanding of the languages and software mentioned in this section to properly update and manage the application when needed.

Form 2: Computer-savvy users

Computer-savvy users
Application: LinkTailor Application
Potential Users: Computer-savvy users
Software and Interface Experience: The user should have good computer intuition, being familiar with using context menus and hotkeys.
Experience with Similar Application(s): The user needs no prior experience with other applications, but will likely be familiar with the Windows Start Menu, browser bookmarks, the file system, and possibly other linking applications such as Raindrop.io or Pocket or RocketDock.
Task Experience: More advanced application clicks – using hotkeys and context menus to add, edit, delete, drag/drop and resize the links as well as customize the links by organizing them into various categories and layouts and assigning them custom tags.
Frequency of Use: Whenever the user wants to open or add links/files, most likely many times in one day.
Key Interface Design Requirements that the Profile Suggests: It is important that the application is customizable and provides options that satisfy the user's desires for how to organize and quickly access their links including hotkeys, tag management, highly customizable styling/theming, and widgets.

Form 3: Computer-inept users

Computer-inept users

Application:

LinkTailor Application

Potential Users: Computer-inept users

Software and Interface Experience:

The user should be familiar with the concept of applications, files, and websites and of saving links to them for the purpose of conveniently accessing them later.

Experience with Similar Application(s):

The user needs no prior experience with other applications, but it is expected they may have had experience with the Windows Start Menu and/or browser bookmarks and/or the file system. Thus the team may model the look and function of the application after these apps so these users can more easily intuit the interface.

Task Experience:

Simple application clicks – using the mouse to add, edit, delete, drag/drop and resize the links and files. The user should also be able to use the keyboard to type in link/file names and uploading images to save for files.

Frequency of Use:

Whenever the user wants to open or add links/files. This could be daily, weekly, or even monthly.

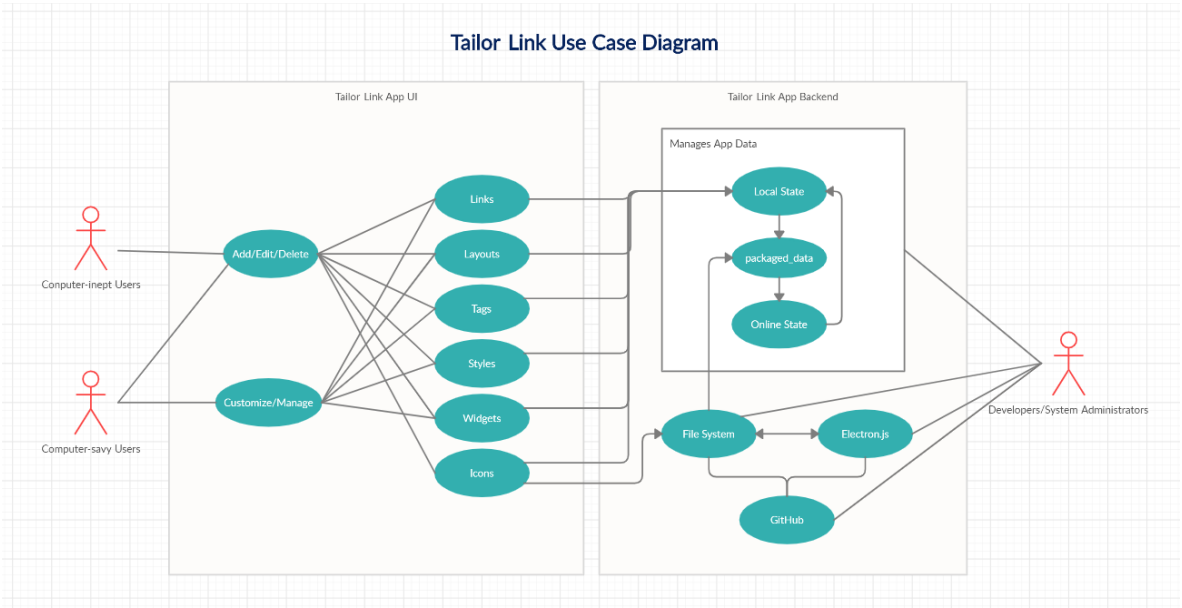
Key Interface Design Requirements that the Profile Suggests:

It is important that the application is easily accessible and able to navigate easily by the user. It should also provide high value with little time invested in the app, meaning the user has minimal decisions to make (default options will fill-in what doesn't have to be explicitly given by the user). Also the app can give recommendations for how to best organize and tag links.

Use Case Diagram

The following use case diagram shows how the three different types of users will interact with LinkTailor and how LinkTailor interacts with the online state, the local file system, and GitHub. Computer-Inept Users and Computer-Savvy Users will both interact with the LinkTailor Interface with Computer-Savvy Users making use of the more detailed features LinkTailor has to offer, customizing their experience. Developers and System Administrators will be interacting with the backend of LinkTailor, managing app data in the local state and the online state and changing code that is managed through GitHub.

This diagram includes the hypothetical, yet-to-be-implemented plans for an online state where user data (such as layouts of links and styles) could be saved online and synced across devices. The diagram also gives a peek into the branding history of LinkTailor, using the name “Tailor Link” which was the working name of the project at the time.



4 Use Case Diagram for LinkTailor

Technical Discussion

The team worked hard to keep technical debt to a minimum. They carefully selected every library, tool, and methodology to improve the modularity of the code while also avoiding big complex libraries and tools to decrease the complexity.

Node.js

The team used Node.js as their runtime and NPM as their package manager. Node is an open-source runtime that uses the V8 engine to execute JavaScript code. It also has an amazing repository of community created open-source libraries and frameworks that are available with the Node Package Manager (NPM).

Electron

Electron is the main back-end library that helps the team create native desktop applications for Windows, Mac, and Linux. It uses chromium browser creating the interface allowing the use of any technology available for web development. Vue.js was the choice for a front-end framework.

Vue.js

Vue is an amazing component based front-end framework that enables the creation of components and modular interfaces. It makes it very easy to organize the front-end code base.

Vue-grid-layout

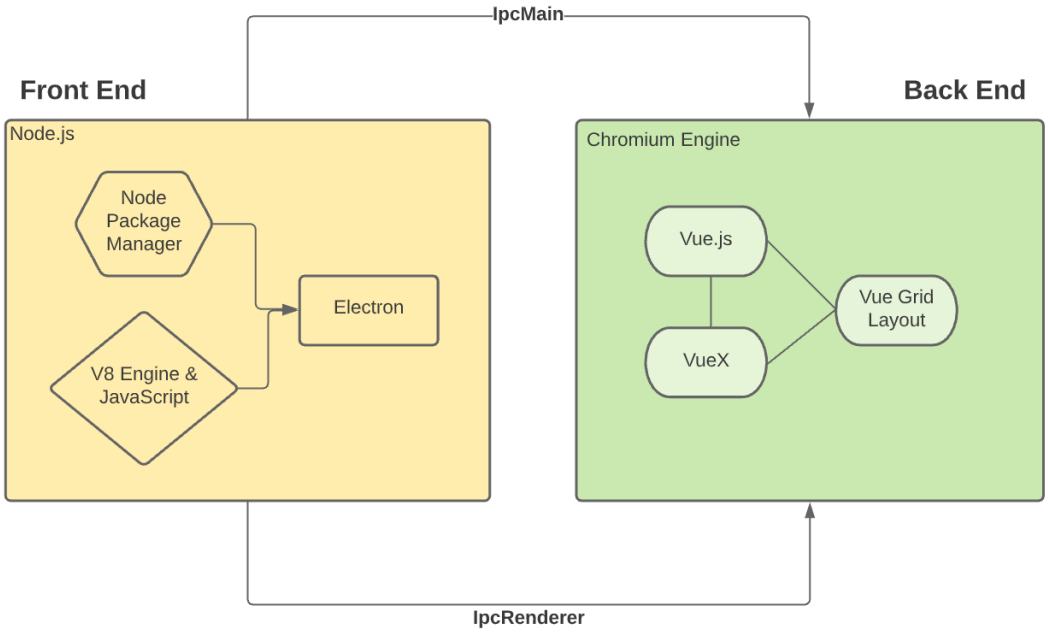
Vue-grid-layout is a grid layout system for Vue. The team used this library to create the grid system.

These are the most notable technologies used by the LinkTailor team. All the interface elements and component structures were built by the team members themselves. They did not want to use any massive design library, instead they learned to make their own designs.

The LinkTailor app is mostly offline except a couple API calls here and there. It runs natively on the operating system that it was installed on. In the future the team plans to implement a cloud solution like Firebase to store user data in a recoverable database.

When running native commands that can modify the operating system LinkTailor uses preloaders and the class that Electon has called ipcRenderer to send events to the back end. This means the node integration is not available to the front end just like a web browser making the app secure from any potential web-based attacks.

The team is using VueX for storage. They created a state object that has all the required data that a user interacts with. The state is then written to the app directory.



5 Diagram of Technologies used by LinkTailor

Testing

The LinkTailor team performed user testing to find bugs and receive feedback to improve the app. Friends and family were recruited to be testers, though testing was (and is still) open to anyone since the app download and user testing form are both clearly available on the public website for LinkTailor.

Beginning the Testing

User testing for LinkTailor began March 29th with v0.1.0, available for Windows and Mac. Potential testers could download a .exe file for Windows and a .dmg file for Mac (from the TailorLink website or GitHub), and the application would install when the file was run. User testers were asked to fill out a Google Form that gave them tasks to complete, so the team could have hard data as well as free responses.

Live Observation: Kemal observed some testers real time as they completed the tasks on the form to learn their thought-process and get inspiration for how to make the app more intuitive. Indeed, several new tasks were created from just the first live observation.

Installer: While the team considered (and partially developed) an installer to give users customization on their install, they decided to opt for a simple executable which installs in a default location for A) consistency in user testing B) ease of releasing new versions as the app was continually updated. In the future, it is likely an installer will be created for LinkTailor (with customizations such as install directory) but at the moment it is not a high priority. (See Appendix A for more on the Installer)

Version Control: Versions of LinkTailor are tracked using GitHub Releases and the standard vMajor.Minor.Patch best practice. Whenever a new version is released, the team updates the LinkTailor.app website so users downloading from there receive the latest version.

User Testing Form and Methodology

The team asked users to consider the ways in which LinkTailor is (or is not) **convenient**, **functional**, and **aesthetically pleasing** in an effort to encourage their thoughtful assessment of the app. The user testing form includes seven tasks for users to attempt to ensure they try a range of features of the app and to receive concrete feedback on whether the intended functionality is indeed working.

Tasks in the form have only two answers a user can select: “completed without a hitch,” and/or “other,” in which case the user is prompted to enter some explanation. This is done to provide quantifiable feedback to the team (in the form of completion as indicated by the first checkbox) as well as encourage

users to give feedback throughout the testing process. Note that users are able to select both options if they wish to indicate completion of the task as well as give additional feedback.

6 User Testing Form screenshot of the "Zoom out" task

Each test has a description which

A) provides more detail for the task (if detail is helpful for clarity) and

B) helps explain to the user how to achieve the task (if they don't adequately understand from the task name).

The amount of description is intentionally short in order to:

A) not create stress within the user from a detailed criteria set or deter the user from reading the full description

B) to leave some intentional vagueness so the team can assess the intuitiveness of the application (i.e. were the users able to complete the task with limited information?).

Each task is strategically aimed at testing a particular functionality within the application and combined are intended to force each user to experience a range of features so they can give feedback on a broad experience of LinkTailor.

The User Testing Form for LinkTailor can be found on the LinkTailor website: https://linktailor.app

Results

The LinkTailor team modified their last weeks of development during the semester based on feedback received in the early stages of user testing. Much of the changes aligned with the initial values for the app, exposing deficiencies in implementation of those values. User feedback for the first two weeks of testing is summarized below with feedback themes and specific examples:

Intuitive. In interface and navigation. Non-confusing. Users navigate without frustration. (for Computer-Inept Users)

Kemal's observations were incredibly helpful in determining default user habits, including what ineffective routs (to accomplish a task) were initially tried. The app is able to be adjusted to accommodate for those user habits, and some changes have already been implemented.

Guiding. Quick to gain benefit from. Helps to promote good organization habits. Little customization needed up-front to start benefiting from the convenience of LinkTailor. (for Computer-Inept Users).

Some users expressed a desire for concrete instructions on features such as hotkeys and even full tutorials to help them understand the conveniences offered to them by LinkTailor.

Full-Fledged: Expected features and conveniences are present. (for all users)

One user expressed a desire for hover tool-tips to explain what buttons would do when hovered over, a feature the team had listed to implement but had yet to accomplish. Another user

desired the ability to “group” links like is done with Android apps on a phone, a feature the team scrapped (greatly reduced in priority) for the time being due to the high time investment required and low return on functionality. With each of these pieces of feedback, these features take on more weight/priority, increasing the chance they will be completed sooner.

Customizable. Users can do what they want with the app to benefit them. They can make it look the way they want. (for Computer-Savvy Users)

No users reported complaints on customization, however this is the sort of value that becomes more salient the more an application is used. The team does not expect first-time users to have much comment on customizability. This is why the team also looks forward to feedback in the future from users who have dived into using the app regularly for weeks or months.

Aesthetic. The app is pleasing to look at. Doesn’t appear buggy. People feel good going back to the app. (for all users)

No complaints were made about the aesthetic of the app, which shows a definite win on design choices.

Budget

The following is a hypothetical budget for LinkTailor containing initial considerations of how money might be generated by LinkTailor and the cost of ongoing development.

Project Asset Type: Revenue Generating: LinkTailor is a brand new application. It is on GitHub, open source, and available for anyone to work with. Revenue will be generated by donations. There is also the possibility that extra features may eventually be made available behind a paywall of some sort.

Funding Source: Self: LinkTailor will initially be developed by the project team and then potentially by others who are interested in investing in such a solution. The code is open source and the team is eager to cooperate with contributors. The team is also open to the possibility of outside investors.

Below is the team's Risk Identification, displaying a reasonably high risk score of 3.6, calculated by considering the project's work effort and complexity. This is calculated from the team's assessment of the project's moderate work effort (3) due to the intensity of developing new software with a small team and high complexity (4) due to the number of integrations desired to make LinkTailor a highly effective application.

Risk Identification

	Risk Rating 1-5 (5 is high)*	*Comments*	*Weight*	*Score*
Work Effort (days)	3	Custom software and small team.	40%	1.20
Complexity	4	Integrations (browsers, OS, file system, online syncing)	60%	2.40
Project Risk Score:				**3.60**

Below is the team's rough Cost Estimate for the project, which comes to $8,000 to develop the initial application and $5,100 annually for ongoing maintenance, bug-fixes, communications, and mild improvements. The calculations use a standard $20/hour pay for software development labor, though the team may be willing to invest in completing LinkTailor for much less. The hardware cost accounts for some amount of server space and processing power that would be required if many users are syncing and backing up data. The ongoing annual cost is highly dependent on the funding model used for LinkTailor and the number of users invested in and making use of LinkTailor.

Estimated Cost Rough Order of Magnitude:

				Ongoing Annual		
	Rate Per/Hr	Work Effort (Hours)	1 X Costs	Rate Per/Hr	Work Effort (Hours)	1 X Support Cost
Labor - IT	20	400	$ 8,000.00	20	250	$ 5,000.00
Labor -External	0	0	$	0	0	$
Software - External			$			$
Hardware - External			$			$ 200.00
TOTAL			**$ 8,000.00**			**$ 5,200.00**

Timeline

First will be shown the initial timeline created for LinkTailor at the beginning of the first semester of development. Then will be shown the actual feature development timeline re-constructed from the team's Kanban Board.

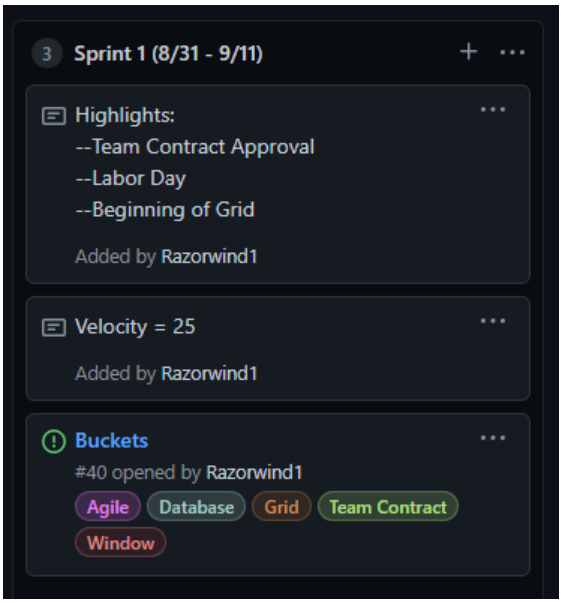
Below is the timeline created for the LinkTailor project before development started. This timeline was intended to be a rough estimate, and development plans changed significantly as the project progressed, in accordance with the deepening understanding the team acquired with each Sprint, and priorities were re-evaluated via the Agile methodology.

Task Name	Duration (Days)	Start Date	End Date
1.0 Project Management and Deliverables	233	8/24/2020	4/13/2021
1.1 Team Building	7	8/24/2020	8/31/2020
1.2 Project Idea Brainstorming	7	8/24/2020	8/31/2020
1.3 Fall Semester Assignment 0: Team Members and Project Name	1	8/24/2020	8/24/2020
1.3.1 Project Name	1	8/24/2020	8/24/2020
1.3.1 Project Logo and Branding	1	8/24/2020	8/24/2020
1.4 Fall Semester Assignment 1: Team Contract	7	8/24/2020	8/31/2020
1.4.1 Project Approval	7	8/24/2020	8/31/2020
1.4.2 Quick Project Timeline	7	8/24/2020	8/31/2020
1.5 Fall Semester Assignment 2: Project Abstract for Tech Expo	29	9/14/2020	10/12/2020
1.6 Fall Semester Assignment 3: Team Contract Resubmission	7	10/05/2020	10/12/2020
1.7 Fall Semester Assignment 4: User Profile	7	10/12/2020	10/19/2020
1.8 Fall Semester Assignment 5: Use Case Diagram	7	10/12/2020	10/19/2020
1.9 Fall Semester Assignment 6: Draft Report	22	10/19/2020	11/9/2020
1.10 Fall Semester Assignment 7: Final Report	22	11/9/2020	11/30/2020
1.11 Fall Semester Oral Presentations	29	11/2/2020	11/30/2020
1.11.1 Practice Presentation	29	11/2/2020	11/30/2020
1.12 Spring Semester	93	1/11/2021	4/13/2021
1.13 IT Expo	1	4/13/2021	4/13/2021
2.0 Research	43	9/7/2020	10/12/2020
2.1 Software Requirements	7	9/7/2020	9/14/2020
2.1.1 Determine Front End Development Language	7	9/7/2020	9/14/2020
2.1.2 Determine Back End Development Language	7	9/7/2020	9/14/2020
2.2 Miscellaneous Research	7	9/14/2020	10/5/2020
2.2.1 Average Number of Websites Booked Marked	7	9/14/2020	10/5/2020
2.2.2 Conduct Interviews with Mock Stakeholders	7	9/14/2020	10/5/2020
2.2.3 Budget Analysis	7	10/5/2020	10/12/2020
3.0 Environment Set-Up	14	9/14/2020	9/28/2020
3.1 Install Libraries for Development	7	9/14/2020	9/21/2020
3.2 Install Electron	7	9/14/2020	9/21/2020
3.2.1 Setup Electron	7	9/14/2020	9/21/2020
3.3 Setup Vue.js	7	9/14/2020	9/21/2020
3.3.1 Install and Configure	7	9/14/2020	9/21/2020
3.4 Setup Github	7	9/21/2020	9/28/2020
3.5 Create Kanban Board	7	9/21/2020	9/28/2020
3.5.1 Create Project Tasks	7	9/21/2020	9/28/2020
3.5.2 Assign Tasks	7	9/21/2020	9/28/2020
4.0 Software Design	21	9/28/2020	10/19/2020
4.1 Create System Diagrams	7	9/28/2020	10/12/2020
4.1.1 Create Network Diagrams	7	9/28/2020	10/12/2020
4.1.2 Create Database Diagrams	7	9/28/2020	10/12/2020
4.1.3 Create Wireframe Diagrams	7	9/28/2020	10/12/2020
4.2 Create Legal Documentation	7	10/12/2020	10/19/2020
4.2.1 Draft Legal Disclaimers and Privacy Policy	7	10/12/2020	10/19/2020

5.0 Development (Back End and Front End)	120	10/5/2020	2/1/2020
5.1 Create Main Page Grid View	14	10/5/2020	10/19/2020
5.2 Create Navigation Side Bar	14	10/5/2020	10/19/2020
5.3 Create Buttons	14	10/5/2020	10/19/2020
5.3.1 Add Link	14	10/5/2020	10/19/2020
5.3.2 Create View	14	10/5/2020	10/19/2020
5.3.3 Edit Link	14	10/5/2020	10/19/2020
5.3.4 Add Folder	14	10/5/2020	10/19/2020
5.3.5 Settings	14	10/5/2020	10/19/2020
5.5 Create Search Bar	14	10/5/2020	10/19/2020
5.5.1 Configure Filtering	14	10/5/2020	10/19/2020
5.6 Design and Develop UI Color Scheme	30	10/19/2020	11/19/2020
5.6.1 Design Multiple Themes	30	10/19/2020	11/19/2020
5.7 Develop App Features	92	11/2/2020	2/1/2020
5.7.1 Develop Notification Functionality	21	11/2/2020	11/23/2020
5.7.2 Develop Syncing Devices Functionality	71	11/23/2020	1/11/2020
5.7.3 Develop Toggling Functionality	22	1/11/2020	2/1/2020
6.0 Testing	30	2/1/2021	3/1/2021
6.1 Functionality Test	30	2/1/2021	3/1/2021
6.1.1 Add Link	30	2/1/2021	3/1/2021
6.1.2 Add Folder	30	2/1/2021	3/1/2021
6.1.3 Create View	30	2/1/2021	3/1/2021
6.1.4 Edit/Remove Links and Folders	30	2/1/2021	3/1/2021
6.1.5 Change Between Views	30	2/1/2021	3/1/2021
6.1.6 Choose Theme	30	2/1/2021	3/1/2021
6.1.7 Search Bar with Filtering	30	2/1/2021	3/1/2021
6.1.8 Open Links with Hot Keys	30	2/1/2021	3/1/2021
6.1.9 Syncing Between Devices	30	2/1/2021	3/1/2021
6.2 Perform User Acceptance Test	30	3/2/2021	4/2/2021
6.2.1 Create Multiple Files, Folders and Views	30	3/2/2021	4/2/2021
6.2.2 Conduct Usability Test	30	3/2/2021	4/2/2021

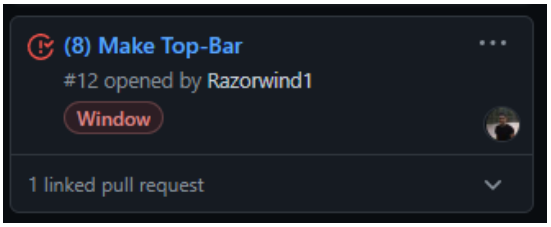
7 Original Project Timeline

Below is a timeline of actual features completed, with images from the team's Sprint Boards which summarize each Sprint with "Highlights," "Velocity" (the amount of work and value provided to the application) and "Buckets" (the categories of work completed). A specific task from the Sprint may also be pictured, to give insight into what features were being implemented that Sprint, and what the team's tasks often looked like:



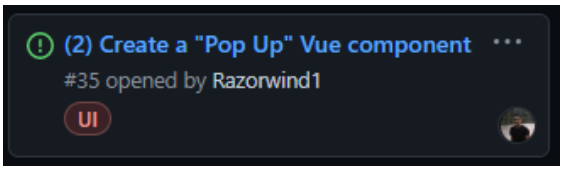
Sprint 1

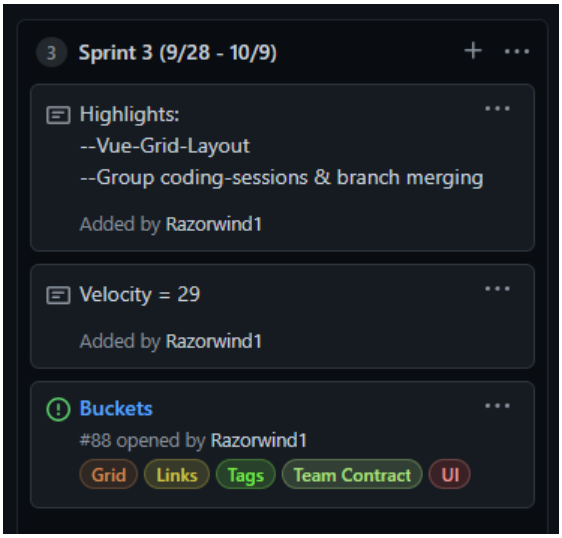
The team began development in the Microsoft Studio Code environment they each set up, using GitHub Desktop for version control.



Sprint 2

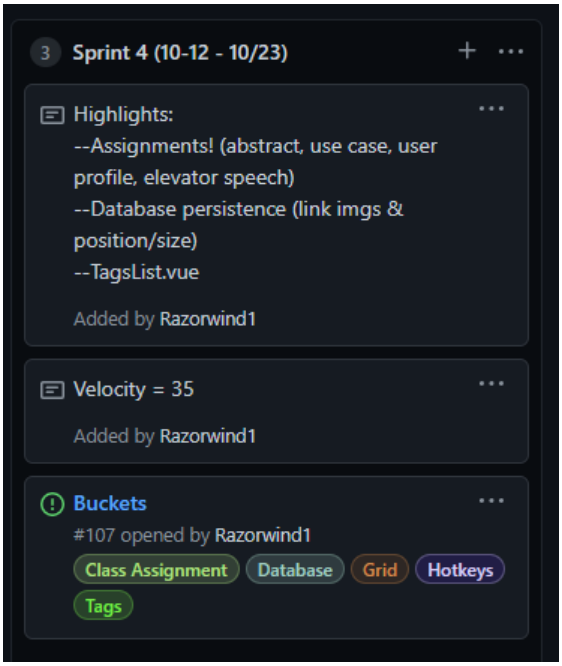
Kemal determined much of the app's look and feel during early development.





Sprint 3

The team helped each other grow in understanding through intentional group coding sessions.

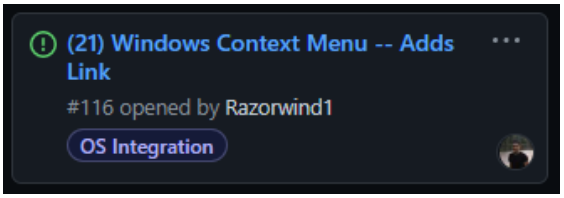


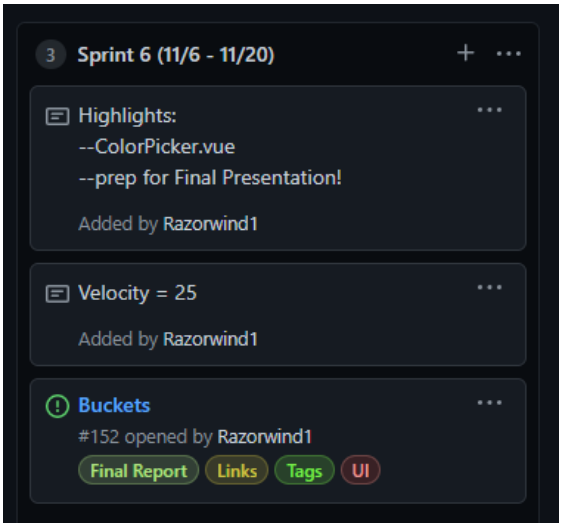
Sprint 4

The vuex store was saved in a state.json file in a user_data folder whenever the store was updated to allow for persistence across closing and re-opening LinkTailor. (See Appendix A for code)

Sprint 5

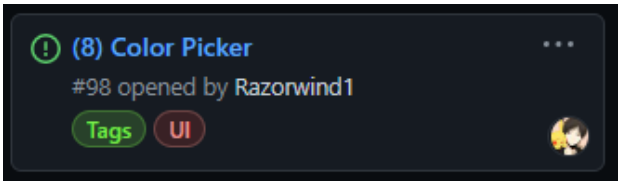
Typically a task with Velocity of (21) should be broken down into smaller tasks, yet for this task it was effective since Kemal completed it in only one Sprint! (See Appendix A for more on this task)

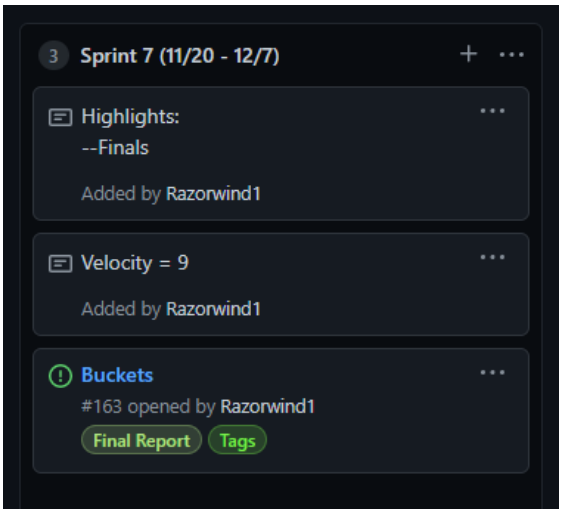




Sprint 6

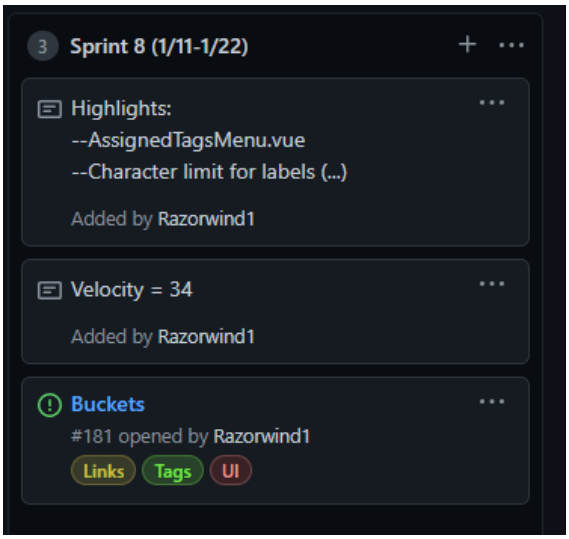
Because of Vue's modular programming, components such as a Color Picker can be easily re-used throughout the application.





Sprint 7

Class assignments were tracked in the Kanban Board, yet only given a Velocity of (1) since they do not directly provide value to LinkTailor.

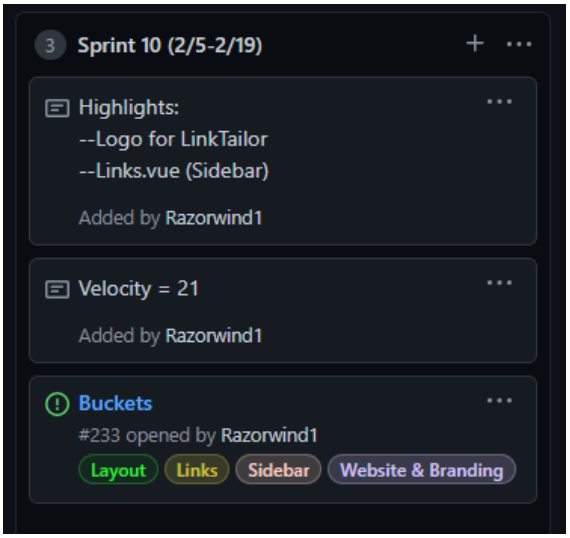


Sprint 8

The team dealt with numerous edge-case and non-ideal scenarios, such as determining how long labels will display.

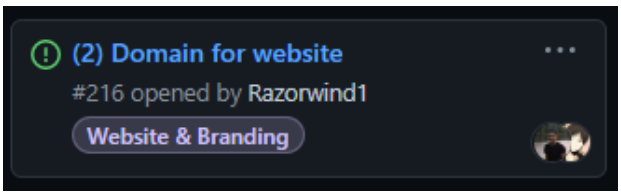
Sprint 9

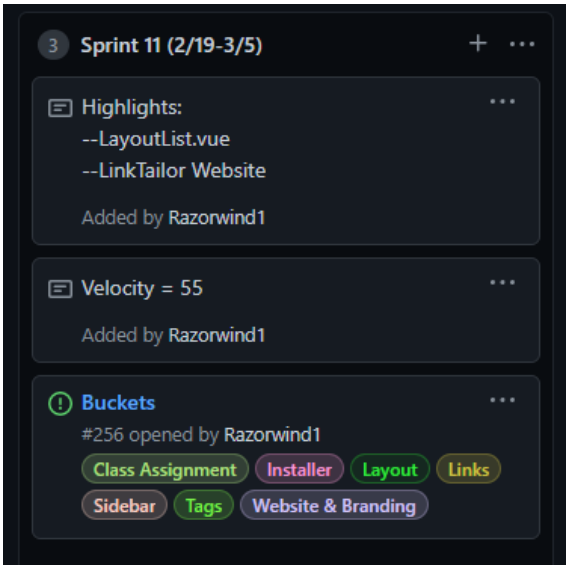
Refactoring and stripping away wasteful code was an essential part of healthy, clean developing.



Sprint 10

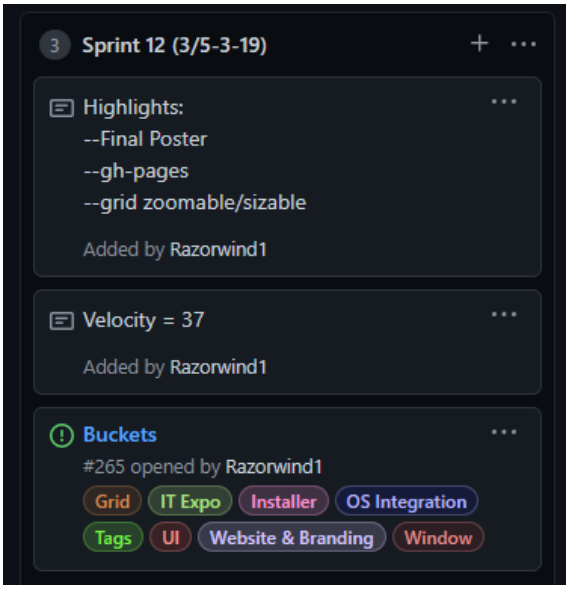
Branding for LinkTailor took thoughtful effort. (See Appendix B for more details)





Sprint 11

This Sprint had the highest Velocity, due to the team's dedicated work building upon previous research and established features.



Sprint 12

The last Sprint covered the most "Buckets" of work as the team added many small features and corrected bugs, while continuing to update the LinkTailor website and prepare for the IT Expo.

Analysis of Problems Encountered

The most consistent difficulty was deciding which work to complete first, in an effort to provide maximum value to LinkTailor and avoid discarded work. There were times in which the team had to undo and discard work because of a change of plans. For instance, in the first week of development, much time was spent implementing the Muuri grid library, but by the third week new discoveries of Muuri's limitations compelled the team to, with some resistance, change to a different library – the Vue-Grid-Layout library. This meant the team had to completely undo their previous work installing the Muuri grid library, install the new library, and modify any work they had done with the Muuri grid to match the new library.

Code refactoring is another source of discarded work. For instance, the team initially used chained-events to communicate up to parent components, and later decided it was better practice to mutate the store as a way of communication. The team learned that it is important to do significant up-front research and plan ahead, holding off on development of work that is not researched or appears likely to change. Also the team learned to make peace with the fact that sometimes, work needs to be discarded. This could be because of poor planning, but also may actually be the most effective path for the development process, given that planning (just like re-working) also takes time (so it is possible to over-plan) and sometimes the only way to discover a need for a new solution is by trying a different one at first.

Recommendations for Improvement

With so many ideas and so little time, the team was forced to prioritize certain functionalities over others. If there was more time they would have included the ability for users to export and import their state and share them with other users of the application. Creating a widget for the application was also high on the priority list. A widget would be a special kind of layout that displayed in a certain area of the screen when the user hovers over an area of their desktop. This would allow the user to have quick access to the widget without having to launch LinkTailor's typical interface. Other functionalities that did not fit into the team's two semesters of development are creating a chrome extension, detailed styling of links, and small user experience enhancements.

If the team were to start from scratch again and knowing what they know now, more functionality would have been completed. There was a lot of time in each Sprint focused on researching and learning how to implement certain ideas which took up a good chunk of time. In the end, this is the reason why all the envisioned features were unable to be implemented.

The team received suggestions from others that LinkTailor could use more description and explanation within the app. Some users did not know what buttons did before trying them and would have preferred explanatory labels to be displayed when they hover over buttons. Users also recommended the documentation of hotkeys and functionality, displayed within the app, so they can become aware of the full experience obtainable with LinkTailor.

In the future, the team would like to continue work on the application, implementing ideas that they were unable to complete this year, continuing to learn and mature as developers, and witness the maturation and growth of LinkTailor.

Conclusion

Throughout this project, the team learned how to work together in an Agile environment, to research and implement new technologies, to follow development best practices, and to spend time contemplating code structure and UX before diving into coding. The team has experienced the pulls of the battlegrounds of both planning and reporting, becoming more aware of how to plan, yet not over-plan and how to report, yet not get caged in documentation. The team did not err on the side over-communicating or over-researching and will likely benefit from continuing to increase engagement in these areas.

The team has worked together to overcome great challenges and is satisfied with the work accomplished. Yet there is still a yearning for more. The team will continue to journey with LinkTailor and would love to see it flourish.

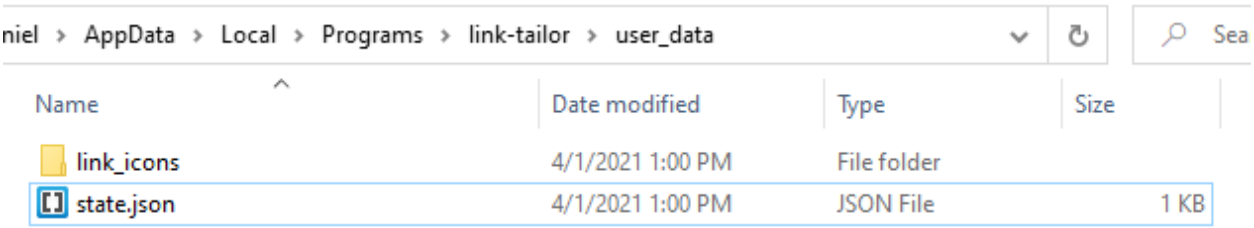
Appendix A. Code

Appendix A contains code snippets, screenshots, and discussion about the technical code of LinkTailor.

The Vuex Store

LinkTailor uses Vuex Store for saving user data. Code snippet below. Link data is not populated by default since users will add their own links (in which case the Store will be updated with a mutation).

User data is saved in a generated file called state.json as shown below.

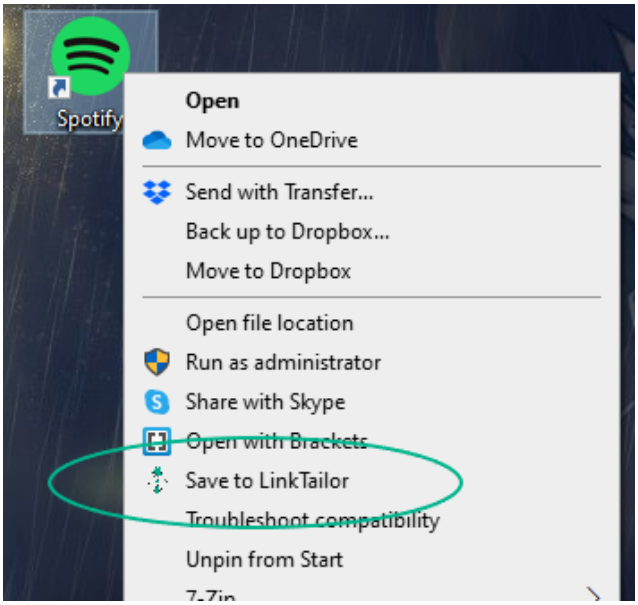


Windows Context Menu – Registry Integration

To integrate LinkTailor with the Windows context menu, the Windows Registry must be edited. Kemal discovered how to run a script that would update the registry, adding an option within the Windows context menu (right-clicking on folders, files, and a background shell) to “Save to LinkTailor.”

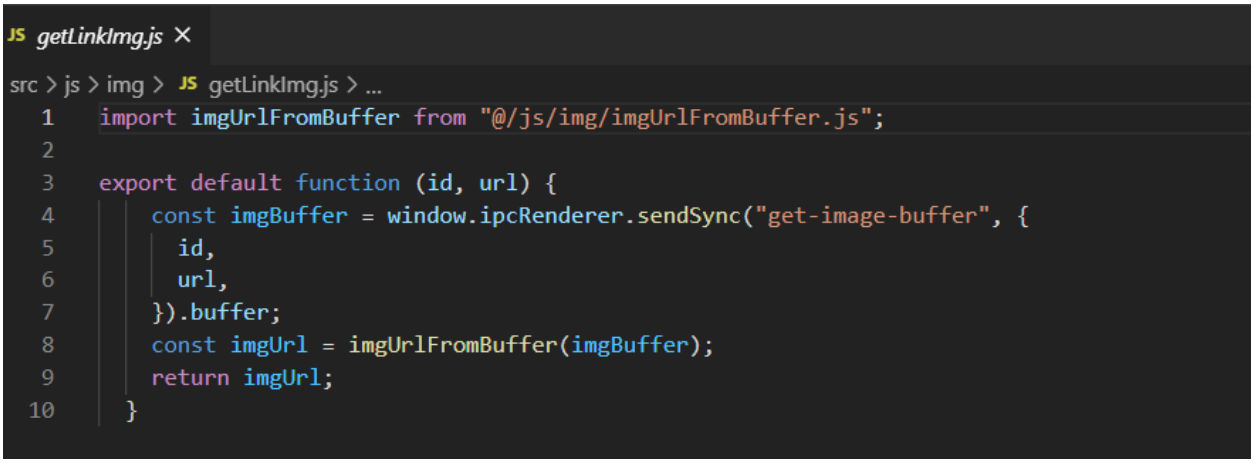
```
Windows Registry Editor Version 5.00


[HKEY_CLASSES_ROOT\*\shell\TailorLink]
@="Save to TailorLink"
"Icon"="D:\\Custom-Links-Project\\dist_electron\\win-unpacked/Link Tailor.exe"

[HKEY_CLASSES_ROOT\*\shell\TailorLink\command]
@="\"D:\\Custom-Links-Project\\dist_electron\\win-unpacked/Link Tailor.exe\" \"--open_dir=%V \""


[HKEY_CLASSES_ROOT\Directory\Background\shell\TailorLink]
@="Save to TailorLink"
"Icon"="D:\\Custom-Links-Project\\dist_electron\\win-unpacked/Link Tailor.exe"

[HKEY_CLASSES_ROOT\Directory\Background\shell\TailorLink\command]
@="\"D:\\Custom-Links-Project\\dist_electron\\win-unpacked/Link Tailor.exe\" \"--open_dir=%V \""


[HKEY_CLASSES_ROOT\Directory\shell\TailorLink]
@="Save to TailorLink"
"Icon"="D:\\Custom-Links-Project\\dist_electron\\win-unpacked/Link Tailor.exe"

[HKEY_CLASSES_ROOT\Directory\shell\TailorLink\command]
@="\"D:\\Custom-Links-Project\\dist_electron\\win-unpacked/Link Tailor.exe\" \"--open_dir=%V \""
```

Below is an example of what the feature looks like after this code is run.



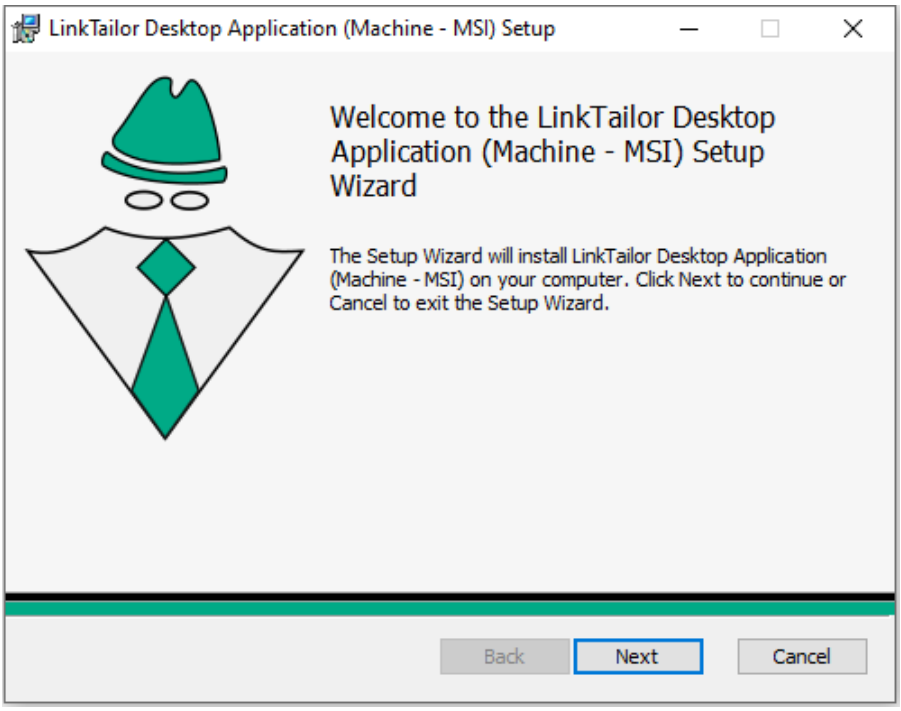
Saving Images

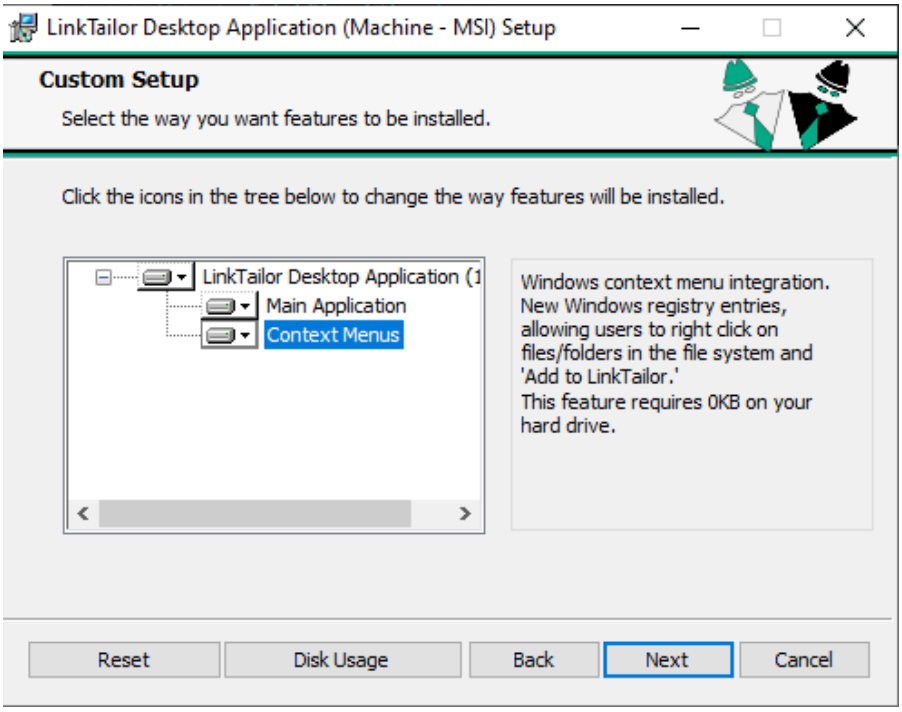
Saving images to local storage and fetching them proved more difficult then the team anticipated. Images had to be converted to buffers and then sent through the event system that put together the front and back end of the application. The team had to be very careful practicing modular coding principles to find a solution to handle the custom images that users uploaded. They also learned about encoding tools and structures like Base64, Buffer, and Blob.



Windows Installer

A windows installer .msi was created for LinkTailor using the WiX toolset in an effort to give the user more customization such as installation directory and to give users the expected features provided by a Windows Installer (such as change, repair, and remove). The other agenda was to provide a way for the Windows Registry to be edited upon install (for the “Add to LinkTailor” context menu feature). Daniel learned a lot about the registry, Wix tools, installation components, and GUIDs, yet was unable to find a convenient way to create a .msi that did not involve manual adjustment. As such, while a functional Windows Installer exists, it is not yet updated for the latest version of LinkTailor and more work will be required to make such an installer test ready.


LinkTailor Desktop Application (Machine - MSI) Setup
Welcome to the LinkTailor Desktop Application (Machine - MSI) Setup Wizard
The Setup Wizard will install LinkTailor Desktop Application (Machine - MSI) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel


LinkTailor Desktop Application (Machine - MSI) Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
LinkTailor Desktop Application (1
Main Application
Context Menus
Windows context menu integration.
New Windows registry entries, allowing users to right click on files/folders in the file system and 'Add to LinkTailor.'
This feature requires 0KB on your hard drive.
Reset
Disk Usage
Back
Next
Cancel

Obtaining unique IDs

To generate unique ids for the entries a well know library in the node package manager called UUID was used. This means every entity had a unique property and the code would never have difficulty differentiating between links even if they had similar properties or had the same name.

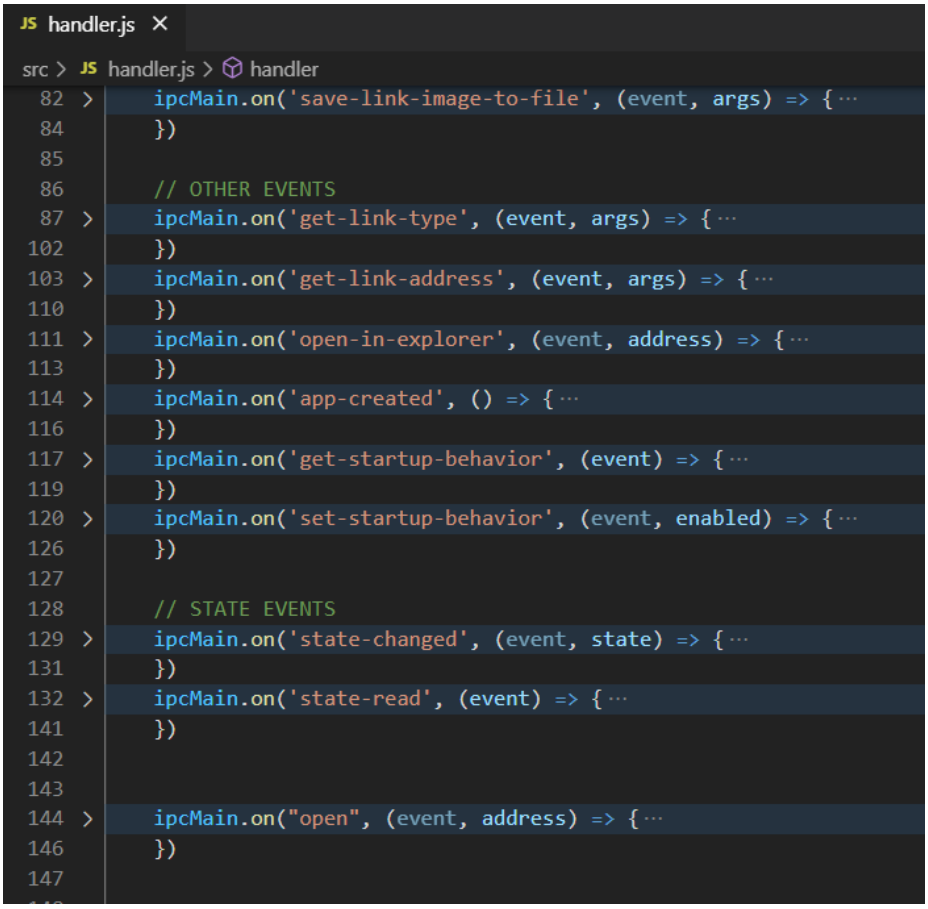
Fetching Website Icons

Fetching website icons proved difficult at first because of the CORS policy and the security requirements of basic HTTP requests. After doing more research, the team found a free web API to help them with this task. This API from Clearbit allows LinkTailor to query for website logos from an extensive database.

```
grabLogo: function () {
  if (this.type === "url" && this.address !== "" && !this.address.match(/^[a-zA-Z]+:\/\//)) {
    this.address = "http://" + this.address;
  }
  if (this.customImg === true || this.type !== "url" || this.address === "") return;
  let host = new URL(this.address).host;
  request(
    { uri: `https://logo.clearbit.com/${host}`, encoding: null },
    (err, res, buffer) => {
      if (!err) {
        this.imgBuffer = buffer;
        this.imgSrc = imgUrlFromBuffer(buffer);
        this.imgLabel = host;
        this.imgFetchClearbit = true;
      }
    }
  );
},
```

Front-End Back-End Hand Operations

By default, Electron does not allow node integration in front-end due to security reasons. The team could bypass this by telling the app to allow front-end node integration, but this would cause security issues since attacks like Code Injections or Cross Site Scripting could result in attackers gaining valuable permissions like executing native cmd commands. This would allow them to have access to the operating system resources like file system and the network. Obviously this would not be ideal, so the team used a preloader to load the ipcRenderer class to the front-end to send events with arguments to the back-end. This meant the front end would not have access to the operating system but rather would send events to the back-end handler to do the tasks for it.



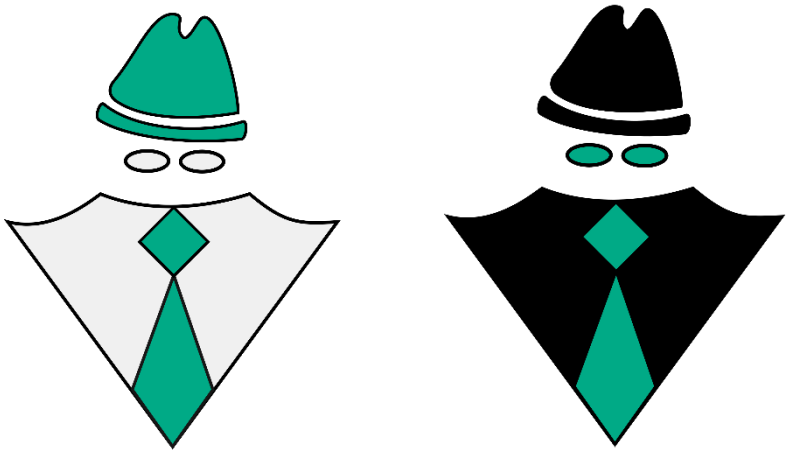
Appendix B. Branding

Appendix B contains logo, branding, and website information and screenshots.

Logo

The LinkTailor logo shown below was created by Daniel using the open-source vector graphics software Inkscape. The logo itself was intentional designed to reflect the purpose and mission of LinkTailor. LinkTailor is “personal” and for the benefit of individuals, hence the logo is a person. LinkTailor is stylish and professional and intended to increase productivity, something communicated by the person’s professional tie and hat. LinkTailor is intended to help the user “tailor” their computer experience to their “fit” their needs, so the person in the logo is a tailor themselves. The logo is also shaped, generally, like a downward pointing arrow (which may be the only distinctive feature if the logo is a tiny icon), indicating that LinkTailor acts as a “pointer” to the various destinations to which the user wishes to travel.

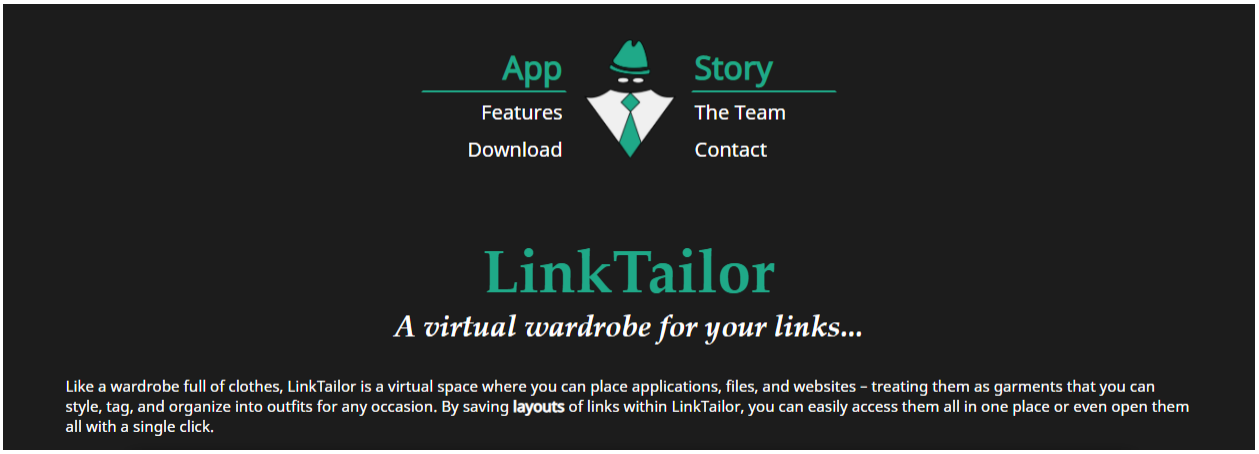
A light and dark version of the logo was created for versatility, with the light version being the default, universal icon of choice since it displays well on both light and dark backgrounds.



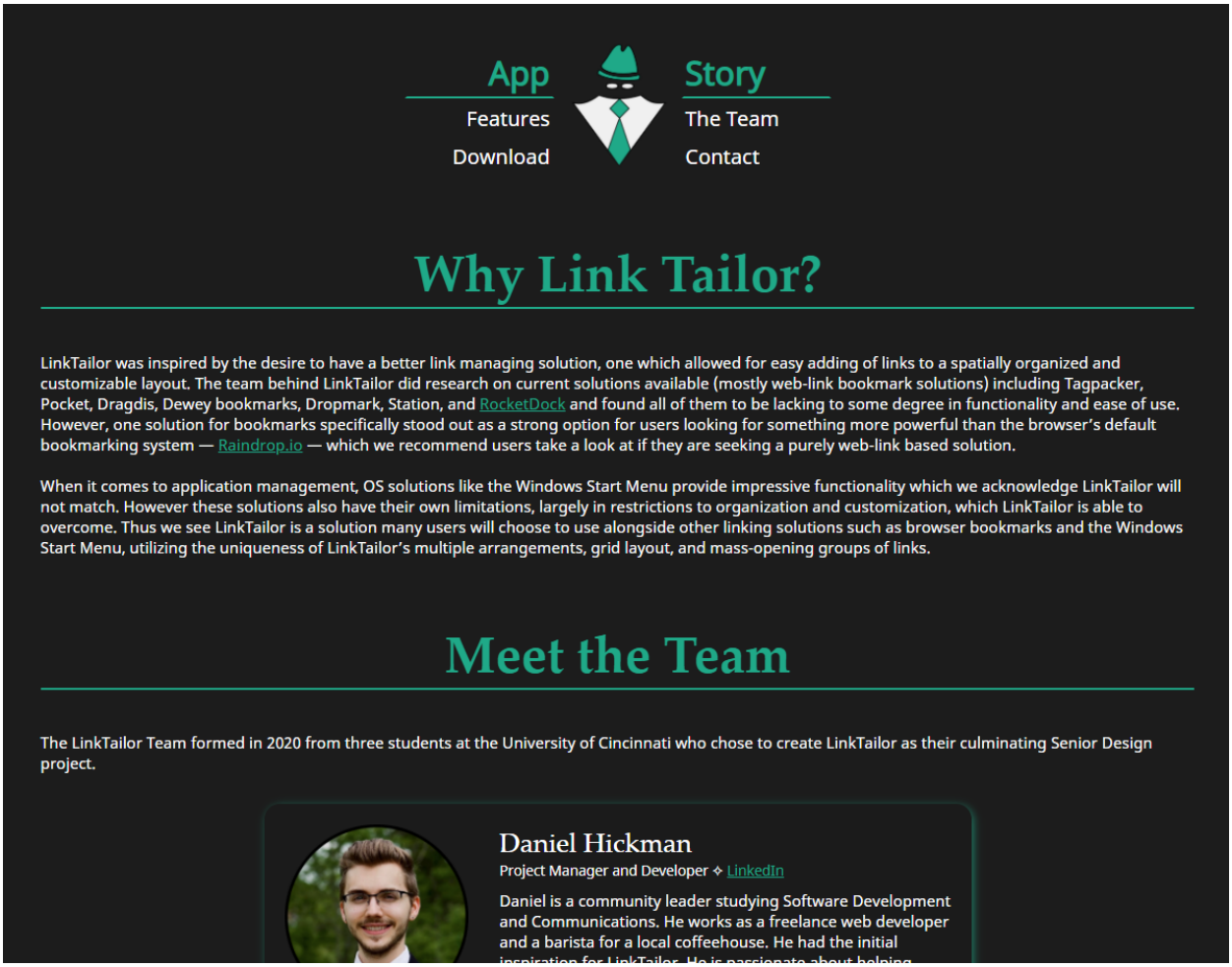
Website

The LinkTailor website was created to publicly tell the story of LinkTailor and allow users to download the application for themselves. Daniel created and hosted the website for free with GitHub Pages, so the website and code live side-by-side. The simple two-page website is made with html and css.

The “App” page explains LinkTailor’s purpose and features and allows for users to download the app.



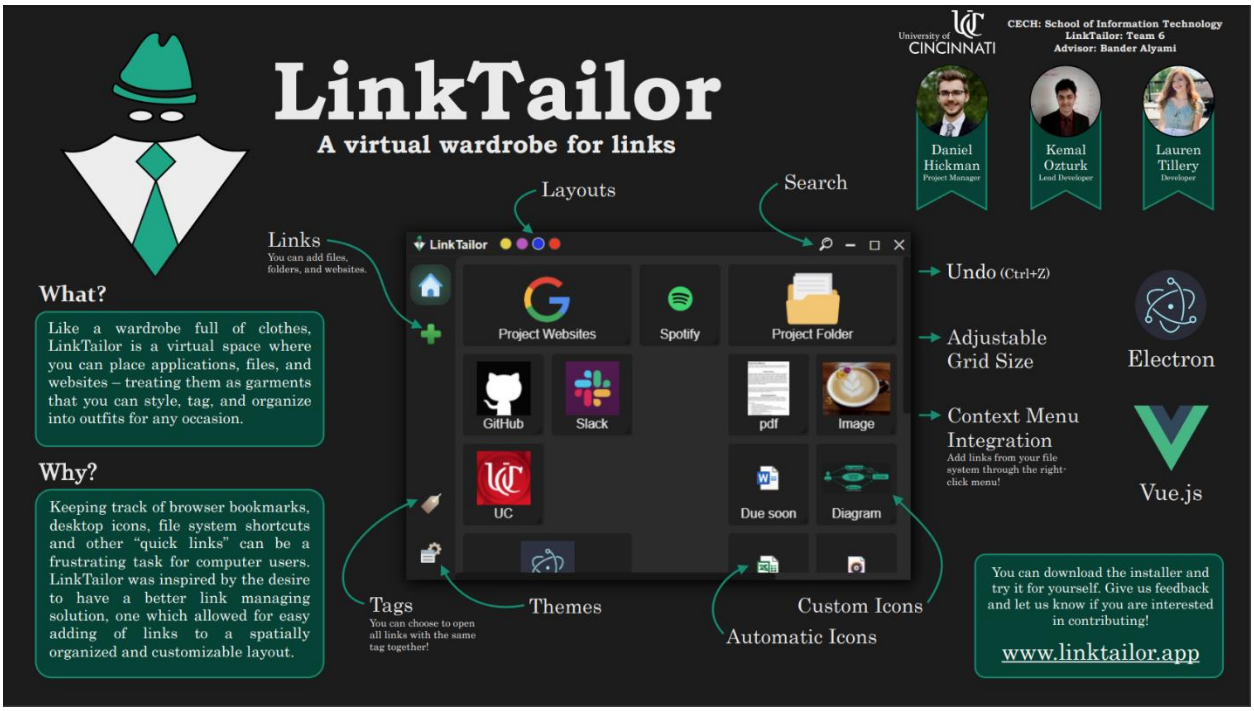
The “Story” page gives background on the inspiration for LinkTailor, the LinkTailor team, and where LinkTailor may go in the future.



The LinkTailor website url is **https://linktailor.app**

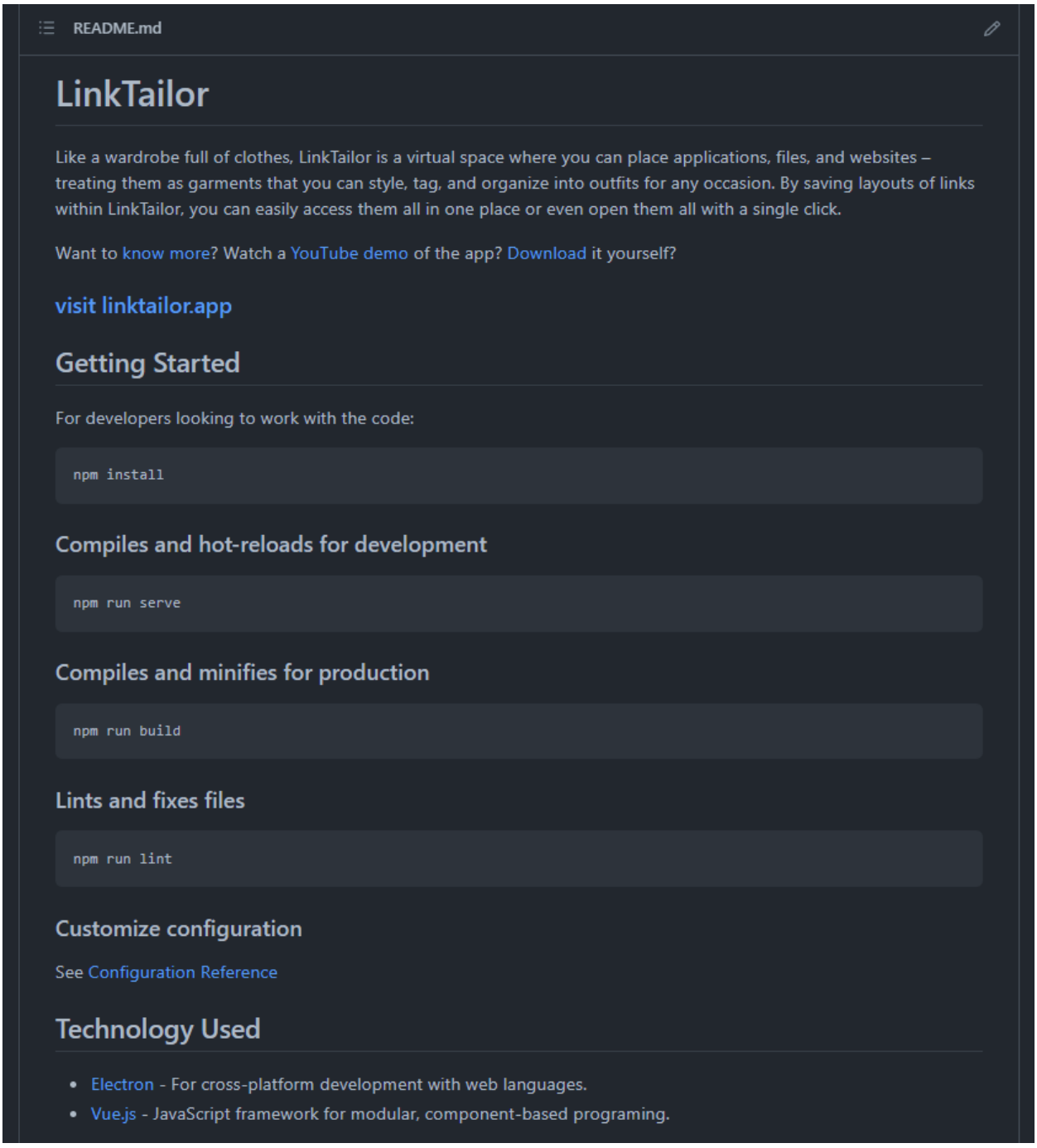
Poster

The LinkTailor poster was created for UC's IT Expo and contains a concise display of the purpose and function of LinkTailor.



ReadMe

The GitHub ReadMe file cannot be forgotten as an important way that LinkTailor is displayed to the world. With so much information on the website, the ReadMe is short and sweet, linking to the website for more detailed information. GitHub url: **https://github.com/Razorwind1/Custom-Links-Project**



README.md

LinkTailor

Like a wardrobe full of clothes, LinkTailor is a virtual space where you can place applications, files, and websites – treating them as garments that you can style, tag, and organize into outfits for any occasion. By saving layouts of links within LinkTailor, you can easily access them all in one place or even open them all with a single click.

Want to know more? Watch a YouTube demo of the app? Download it yourself?

visit linktailor.app

Getting Started

For developers looking to work with the code:

```
npm install
```

Compiles and hot-reloads for development

```
npm run serve
```

Compiles and minifies for production

```
npm run build
```

Lints and fixes files

```
npm run lint
```

Customize configuration

See Configuration Reference

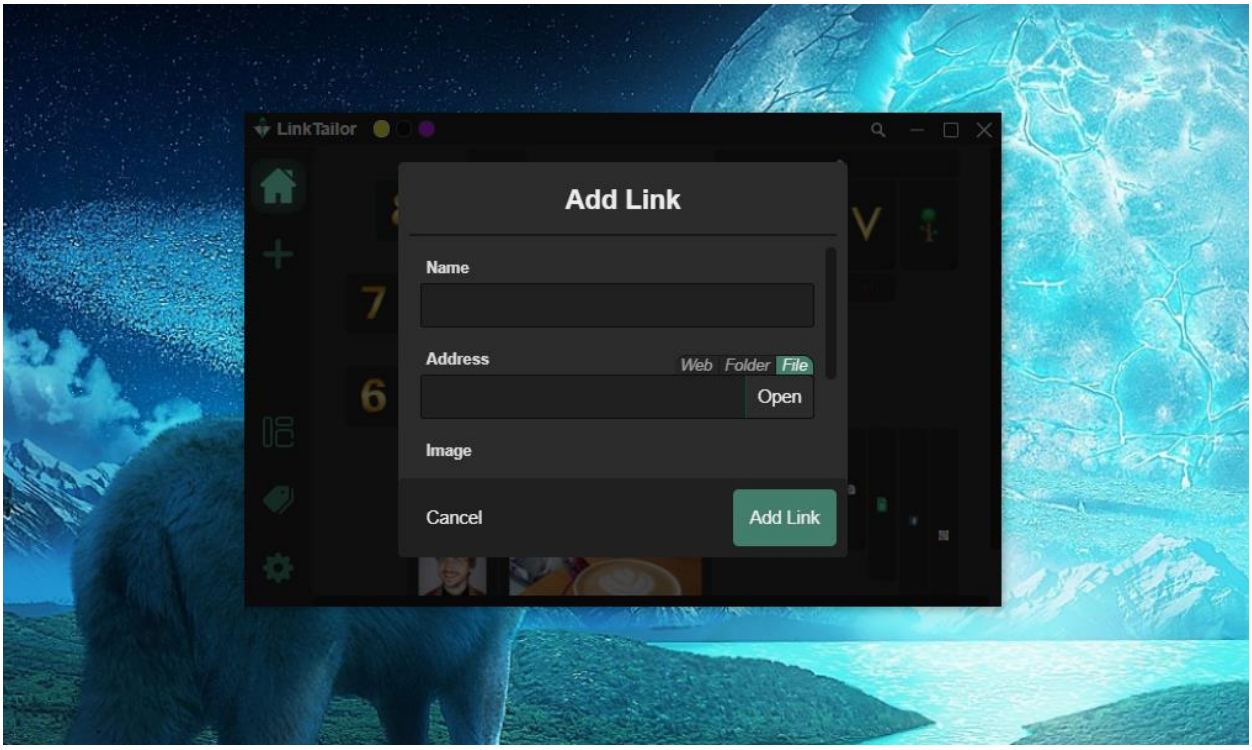
Technology Used

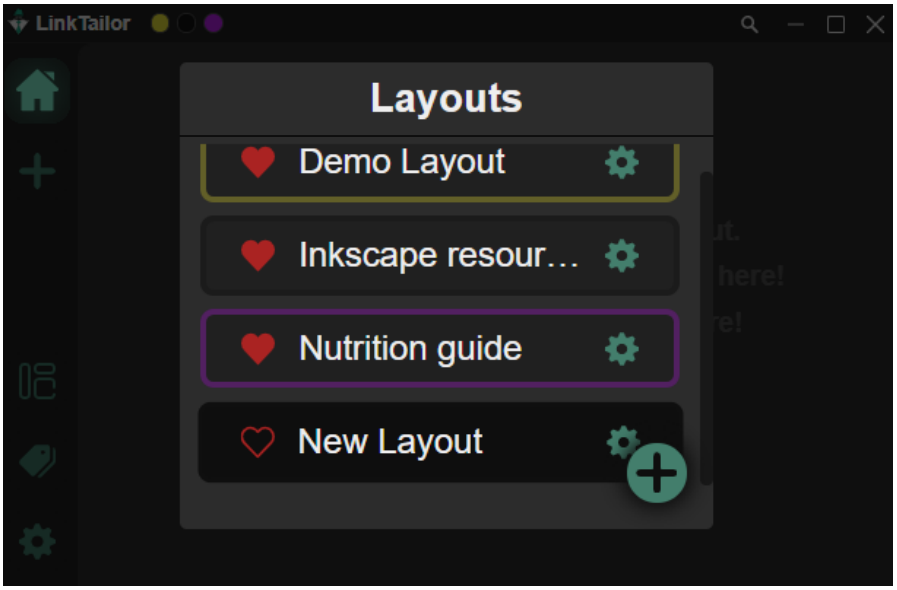
- Electron - For cross-platform development with web languages.
- Vue.js - JavaScript framework for modular, component-based programing.

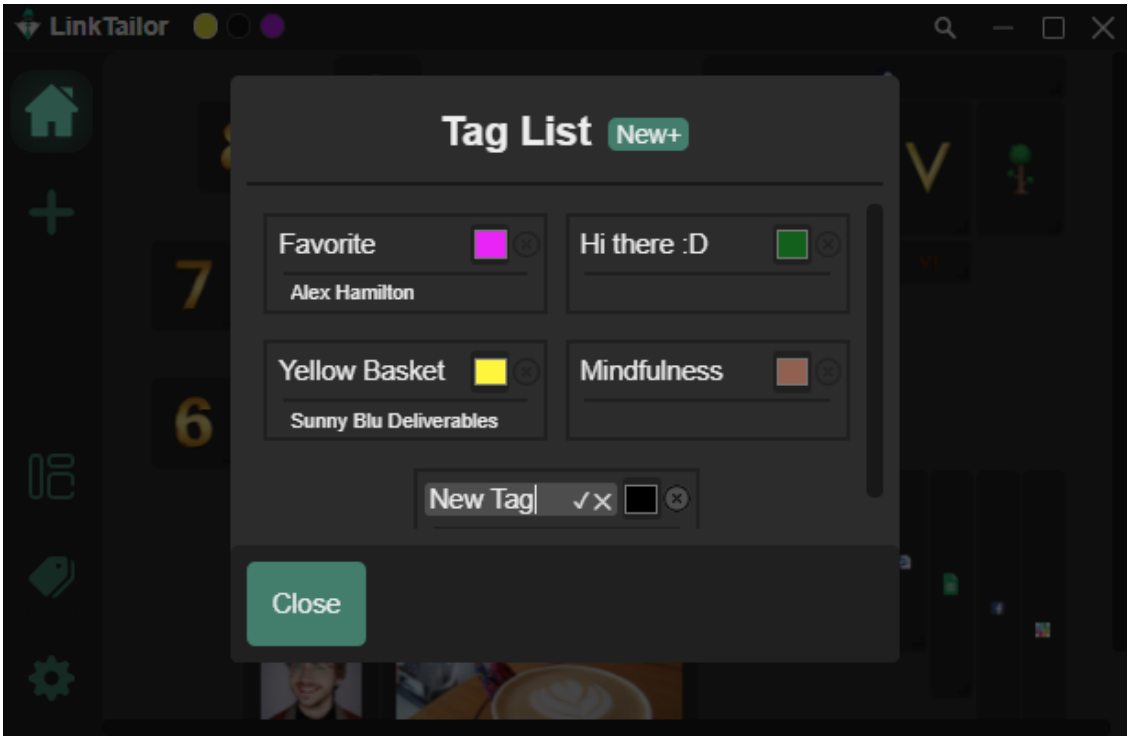
The App Interface

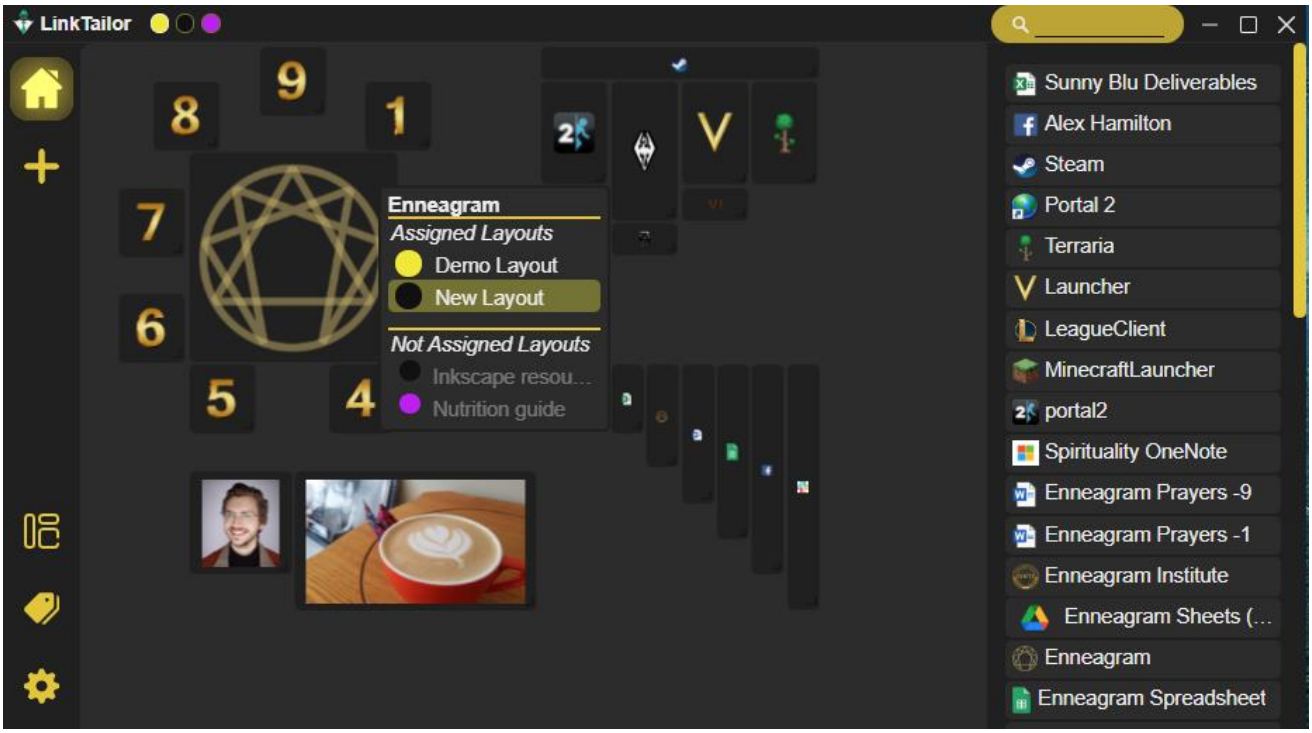
It is only fitting that with so much talk of the LinkTailor app that the interface be shown in this report.

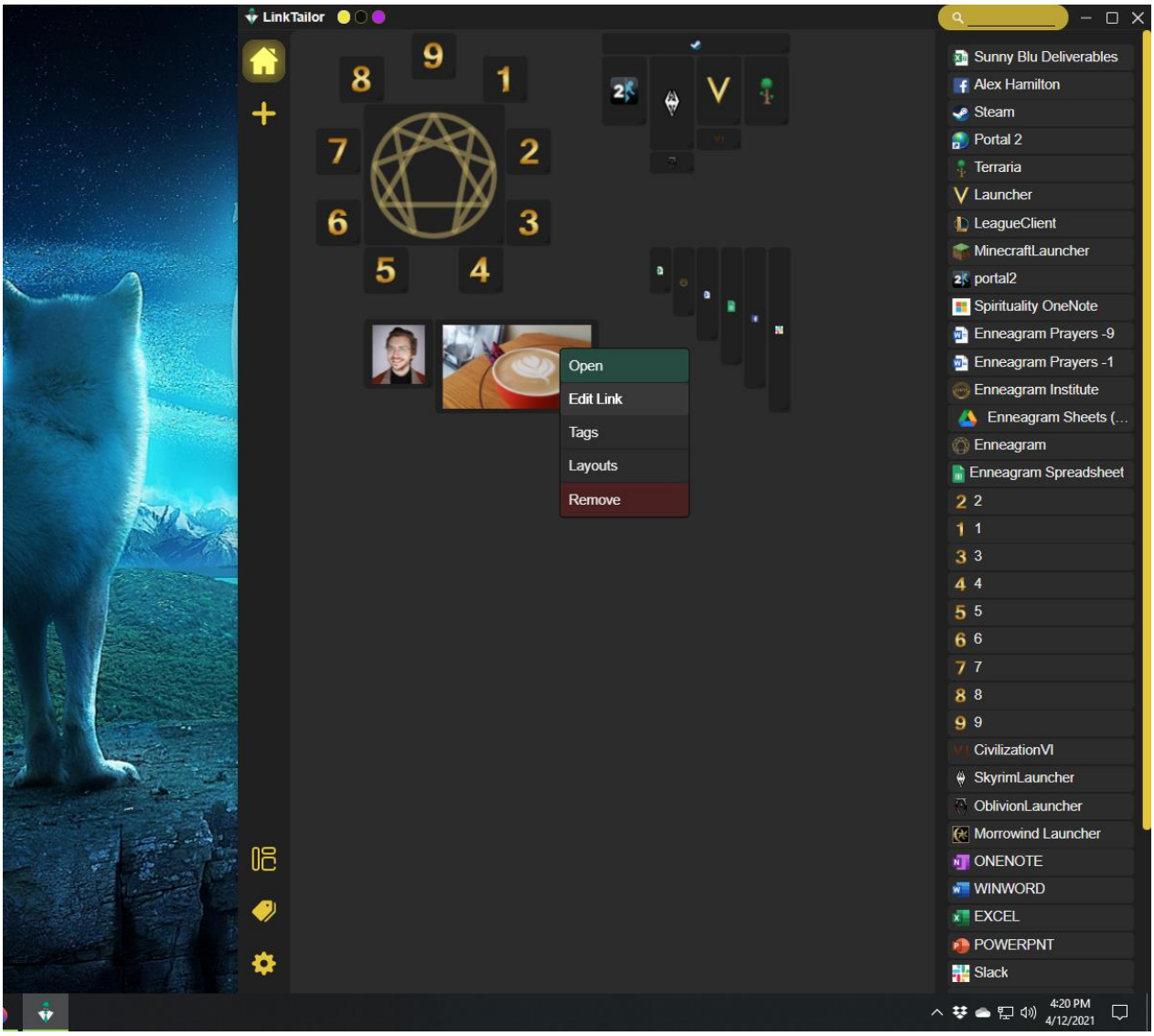
Below are several screenshots displaying the most recent version of the LinkTailor app.






LinkTailor
Tag List
New+
Favorite
Alex Hamilton
Hi there :D
Yellow Basket
Sunny Blu Deliverables
Mindfulness
New Tag
Close


LinkTailor
Enneagram
Assigned Layouts
Demo Layout
New Layout
Not Assigned Layouts
Inkscape resou...
Nutrition guide
Sunny Blu Deliverables
Alex Hamilton
Steam
Portal 2
Terraria
Launcher
LeagueClient
MinecraftLauncher
portal2
Spirituality OneNote
Enneagram Prayers -9
Enneagram Prayers -1
Enneagram Institute
Enneagram Sheets (...
Enneagram
Enneagram Spreadsheet


LinkTailor
9
8
1
7
2
6
3
5
4
Open
Edit Link
Tags
Layouts
Remove
Sunny Blu Deliverables
Alex Hamilton
Steam
Portal 2
Terraria
Launcher
LeagueClient
MinecraftLauncher
portal2
Spirituality OneNote
Enneagram Prayers -9
Enneagram Prayers -1
Enneagram Institute
Enneagram Sheets (...
Enneagram
Enneagram Spreadsheet
2 2
1 1
3 3
4 4
5 5
6 6
7 7
8 8
9 9
CivilizationVI
SkyrimLauncher
OblivionLauncher
Morrowind Launcher
ONENOTE
WINWORD
EXCEL
POWERPNT
Slack
4:20 PM
4/12/2021

References

[i] L'Angelier, Eliot. (2019) *Google Chrome bookmarks – a UX case study*. Medium. https://uxdesign.cc/google-chrome-case-study-bookmarks-83a32b7c754

[ii] Descy, Michael. (2018) *Re-committing to Pinboard, after many months away*. Snippets. https://mjdescy.me/2018/07/10/re-committing-to-pinboard-after-many-months-away/

[iii] Dubroy, Patrick. (2009) *How many tabs do people use? (Now with real data!)*. dubroy.com. https://dubroy.com/blog/how-many-tabs-do-people-use-now-with-real-data

[iv] Alsop, Thomas. (2020). Computer penetration rate among households worldwide 2005-2019. Source/Publisher: ITU. Release Date: November 2019 https://www.statista.com/statistics/748551/worldwide-households-with-computer/

[v] Statista Research Department. (2015) Number of websites visited by new and established internet users in the UK 2014. https://www.statista.com/statistics/322946/number-of-websites-visited-by-new-and-established-internet-users/

[vi] Fyodor. (2019) *How do you manage your browser bookmarks?* Dev. https://dev.to/fyodorio/how-do-you-manage-your-browser-bookmarks-1hcb