The Un-Division Network ~ Subdivision Builder

Project Engineer: Jacob Darabaris

Sponsor: Deborah Hwang

Project Advisor: Deborah Hwang

Computer Science 497

University of Evansville

May 2, 2019

ABSTRACT

The goal of this project is to create a web application for which Home Owners Association (HOA) Members can design a subdivision and attach information to the objects inside of it. Members can then save the subdivision to a database, and the map becomes an organizational and record keeping tool that fills a need not yet accounted for. Once a subdivision is saved, this application also can be useful to realtors and prospective homebuyers, who can use the information provided by HOA members to make more informed decisions.

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor Dr. Deborah Hwang for keeping me on track, helping maintain sight of the project focus, and fielding all general questions. I would also like to thank my friend Turki AlHarbi for help with both the initial database set up as well as general back end structure planning. Finally, I would like to thank my friends Majd and Mohmmad Soufan for their indispensable advice regarding the back end, as well as fielding many database specific questions.

LIST OF FIGURES

INTRODUCTION

Across the US, two types of homes are being built: Large estates in the country, and Family Designed Homes in subdivisions. Subdivisions pop up everywhere with tens to hundreds

of similar homes, creating effective communities in each at the smallest possible level. These communities largely have been ignored and underrepresented on the Internet, and this project aims to change that. The goal is to create a tool for these community members, or Homeowners Association (HOA) Members, in various subdivisions across the US to make their current responsibilities easier, as well as open new avenues for meaningful connections and experiences. It would exist as a web app designed for larger screens. This application platform features a unique subdivision builder/viewer, containing housing and resident information on an interactive map.

This subdivision builder is the key focus of the project, and it is a program that lets users add a personal touch to their personal subdivision overview. They can digitally build and personalize the map of their subdivision in an interactive view, adding information such as: housing vacancy or basic resident information, not limited to: family name, if there are kids any outdoor pets, fences, contact info, etc. Undeveloped and proposed building info is visible on the map as well. In addition, there is potential access to resource consumption statistics, such as water and electricity usage, to support incentives amongst members to be the greenest as possible. The application is designed to be part of a much larger platform with a profile system to utilize the builder in different and more useful ways. Those who utilize this platform can make smarter buying decisions, better plan events, and make new connections with neighbors.

PROBLEM STATEMENT

The idea of a subdivision is simple, develop an area of land fit for modern homes, then produce homes similar in size and features across a grid structure in that area. In most growing towns and communities, subdivision birth and expansion are nearly inevitable, as the need for

fast and cheap housing is driven upwards by new families. In terms of housing, subdivisions are the most efficient use of land, save for towering skyscraper complexes [1]. The scope of subdivisions can be broken down to 3 areas: building and planning subdivisions, buying and selling subdivision housing, and post-sale maintenance. Historically these 3 areas have been loosely linked only as needed, and with few useful tools and resources available in each area.

A network needs to be established to link these distinct parties, and better serve their needs and responsibilities. For Building and planning, the two focus users are Builders and Developers. Developers lay the groundwork for the subdivision, whether it be electrical lines, plumbing, roads, and more as well as working closely with the town to satisfy rules and regulations. The builders then come in and develop the actual homes. Builders and developers need a platform to better communicate the present and future state of the subdivision to each other, as well as serve as a hub for pooled information [1].

For the buying and selling users, realtors and prospective buyers make up the mix. Many great homebuying solutions exist for both realtors and homebuyers to utilize, but most information they provide is impersonal and only deal with monetary value and physical features. There is not a way to learn qualitative details about a community without spending a lot of time there in person. Realtors also struggle with gathering information on homes, as there is no central database. A lot of time could be saved if there was only a unified source of information [2].

Post-sale maintenance users refer to people or groups who are now essentially on their own looking to upkeep their community. At this point, the developer is long finished with his work, and the builder is likely inactive as most of housing lots are finished. Home Owners Association Members and the Management Companies they employ make up this third category and they are truly in the most need of assistance. HOA members are ordinary people trying to

decipher cryptic building rules and home expansion regulations. They are concerned about their long-term home values but also would like a better way to organize each other. They need an outlet to answer all of their questions as well as keep in touch with key players in the organization and above [3].

A subdivision home flows through the following steps: A developer develops the land. A builder builds a home on the land. A prospective buyer seeks to buy a home. A realtor sells them a home. Then finally that homeowner becomes a HOA member, and likely works with a management company [4]. The problem lies in the fact that at each stage in the model each group quickly breaks ties with group that led them to the next step, and a lot of information and opportunities to communicate are lost (or exclusive to each group). A unifying social network with intuitive features is needed to avoid treading over the same grounds twice and allow for progress in previously impossible ways.

Specifications

The Un-Division Network will exist as a Web Application and will require the use of several different languages and implementation of several different features. Originally, both PHP and JS were planned to be used sparingly for both the back and front end respectively, and C# via the ASP.NET Framework was to be used for the actual web application—the subdivision builder. Unfortunately, a lack of research during the design phase caused considerable changes in implementation. It was originally thought that C# could be used to build web applications akin to the way desktop applications are created via WPF or UWP. Unfortunately, C# Web Code, also known as Razor Pages, are primarily designed for back end work, specifically linking data objects. JavaScript and jQuery libraries became the sole means of front-end implementation. The

project still utilizes MS SQL Server as a database for user information as planned. The database is functional but minimal.

Due to limitations in time and understanding of the database, a profile system was not implemented. Many planned features, such as hierarchical views dependent on user type and support for an unlimited number of subdivisions based on which subdivision a member is a part of, had to be postponed for later development. The focus also completely shifted to the Subdivision Builder, and the other three planned areas are completely absent. Even without the hiccups, it would have been extremely unlikely that these features would have been present and worked well.

With the focus set on the Subdivision Builder, specific requirements come into focus. Users should be able to design and a subdivision and save it, and view and edit it again at a later time. There should be one page strictly for building a subdivision, and other strictly for viewing the subdivision. In designing a subdivision, users should be able to add and position objects on screen and assign attributes to them. Objects should change depending on user inputs and attributes assigned to them. The interface should be easy to use and intuitive.

DESIGN APPROACH

The Subdivision Builder was approached by addressing the following questions: ‘how do features work’, followed by ‘how will someone break this’, and finally ‘is this fun and easy to use’. First a home screen was created, which does not offer any functionality or purpose other than a glimpse into the scope of the original project (see Figure 1).

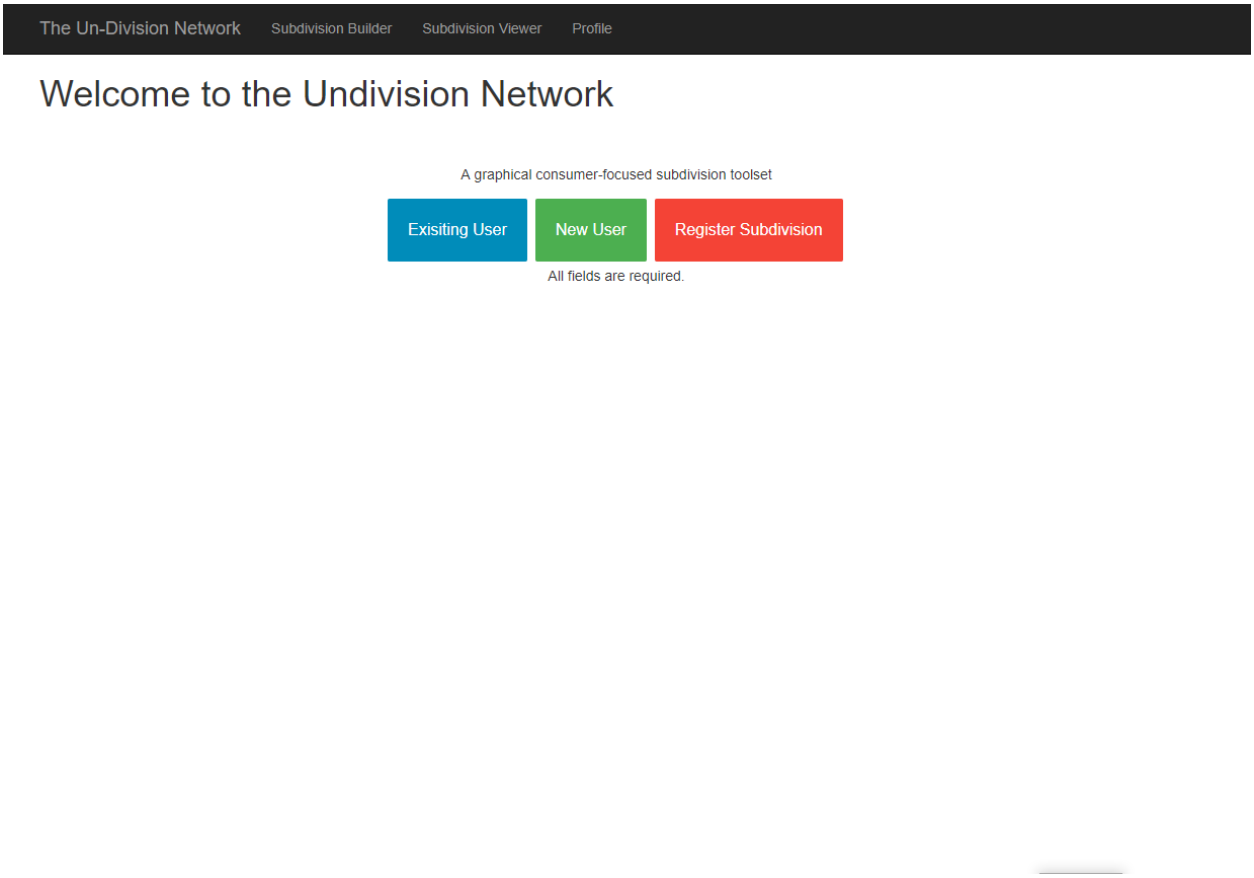

Figure 1: Application Home Screen

The first true functional problem was choosing a way to allow the addition of images to a web page, and then naturally, manipulation of said images. A JavaScript library called jQuery UI proved to be exceedingly useful early on, with a feature called ‘append’, which adds a specified element to either the start or end of a targeted parent object. Another feature called ‘draggable’ allowed users to reposition the image on the page within the confines of its parent object. Draggable also has a ‘snap’ attribute that will snap an object’s border to that of another adjacent object’s border. At this point, the base level desired functionality was achieved, and focus could be diverted to a number of different requirements.

The next goal was to work on the actual layout of the page. First a collapsible side bar, specifically a list-tree, was implemented containing a list of possible image objects to be added by the user, such as roads and homes. Then the frame for which the subdivision could be built on screen was created, and it was deemed best to only allow large desktop screens to be able to use the builder (see Figure 2).

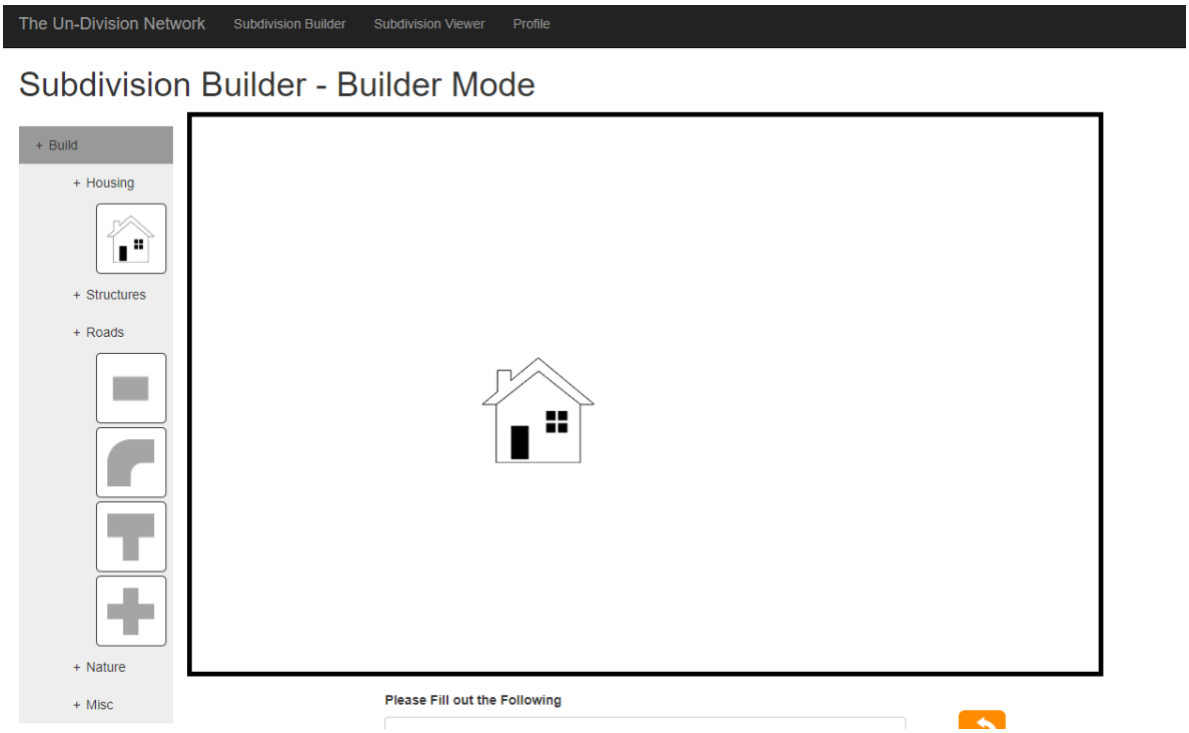

Figure 2: Builder Mode with House Object Placed

So rather than resize the frame on smaller screens, users with screens smaller than a designated pixel width would instead see a message informing the user to switch to a larger screen (see Figure 3).

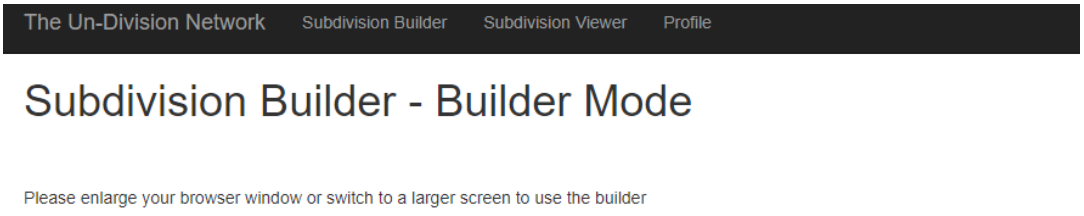

Figure 3: Builder Mode on a Small Screen

A feature that would have been ideal to implement here would be a resizable and zoomable frame. Resizable in the sense that the frame can be made bigger than default, but not smaller. Users would likely want to build a subdivision larger than that which could fit in the frame. This is where the main problem with the design began to emerge.

While the choice to use jQuery was ideal for bringing quick results to the project, it ultimately proved inflexible enough for the project's particular needs. jQuery provides only two rigid functions for adding element to a DOM, append and prepend, the latter of which was not used in this project. As stated, append adds a given element inside a specified parent element after the last child. What caused problems though, is even if an object is appended and moved around (using 'draggable'), the page still remembers the origin of that initial placement. Subsequent appends will act as if the last child is still present at its origin position and would then place it after what looks like empty space (see Figure 4).

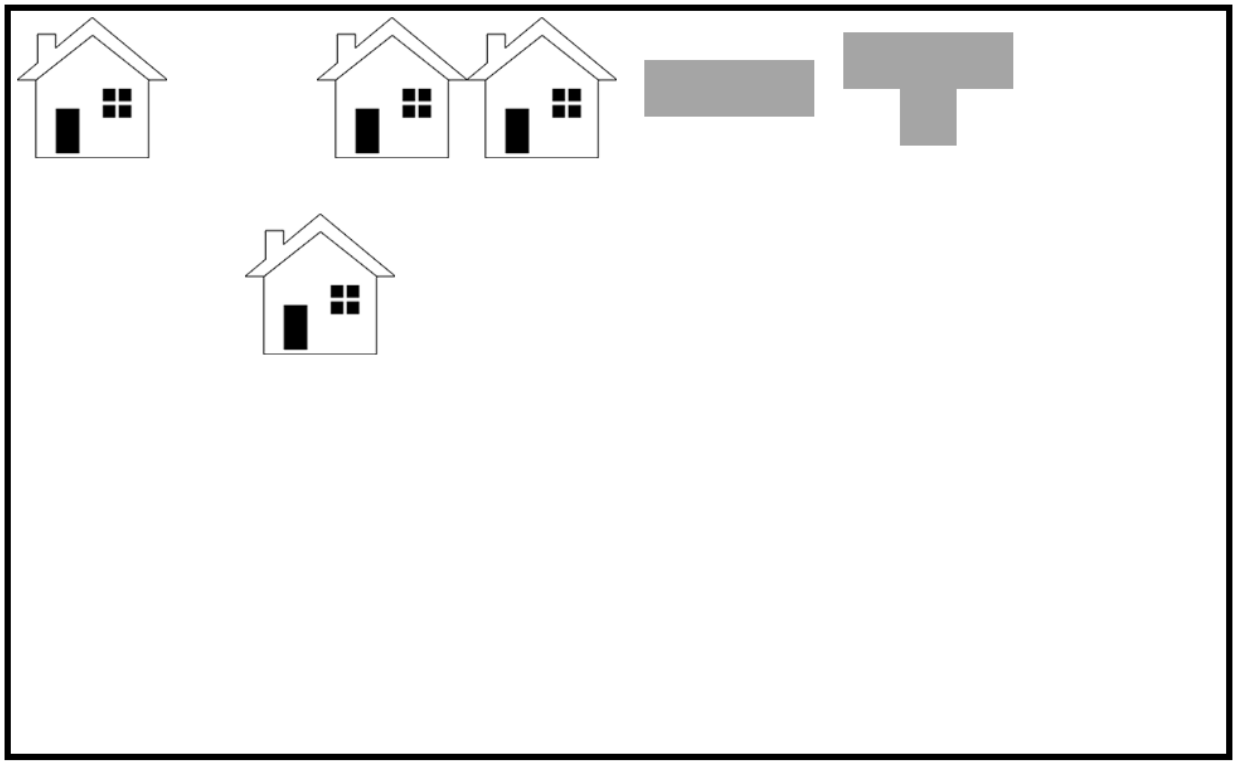

Figure 4: Example of ‘Invisible Last Child’ - jQuery Limitation

In this example, six objects were appended to the screen, but the second object appended was moved from its original spot. The third object (assuming top leftmost is the first object) was appended not immediately next to the first object, but after an empty placeholder space for the second object that was moved. Attempts were made to address this issue, notably applying CSS to each object to make its top and left coordinates zero. From all the forum posting and documentation (or lack thereof) available, it seems not possible. This would cause more problems later.

Aside from this issue, the next step was to make a rotate button as well as an onscreen form that are only present after a user clicks an object to add. Six fields for the form appear on click, three required, for the user to fill. Two of the required fields, Build Status and Owner

Status, are selection fields that physically change the object that was previously added to the screen, specifically its color and overlay (see Figure 5).

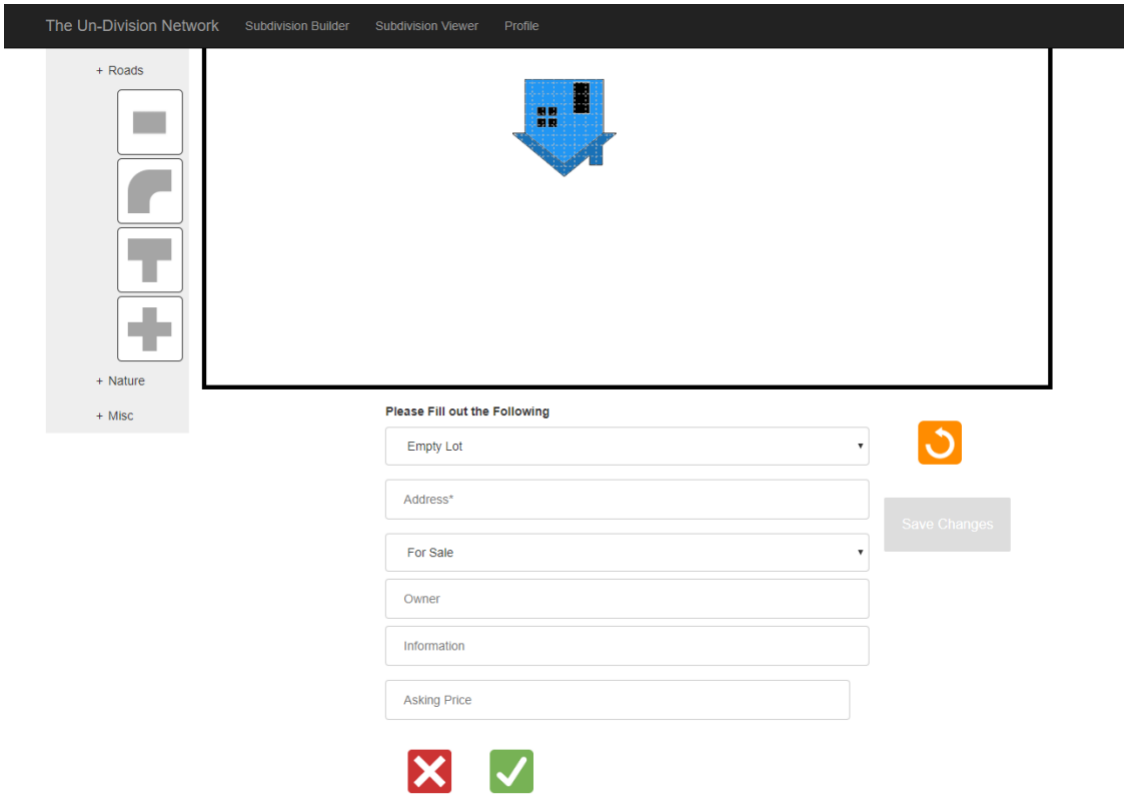

Figure 5: Builder Mode with Attributes applied to House Object

After filling the required fields, the user will be able to click a green ‘Checkmark’ button and the object’s attributes, rotation, and position will be locked into the page. Alternatively, at any point the user can click the red ‘X’ button and the object will be deleted. The functionality behind the green ‘Checkmark’ button was the first to require use of the backing C# Razor Pages, which stores all the form elements, as well as other relevant information, into appropriate holders (see Figure 6).

```
5 references | Jacob Darabaris, 24 days ago | 1 author, 1 change
public class Structure
{
    0 references | Jacob Darabaris, 24 days ago | 1 author, 1 change | 0 exceptions
    public int StructureID { get; set; }
    0 references | Jacob Darabaris, 24 days ago | 1 author, 1 change | 0 exceptions
    public int StructureDBID { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Style { get; set; }
    0 references | Jacob Darabaris, 24 days ago | 1 author, 1 change | 0 exceptions
    public string ImageSource { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Address { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Owner { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public double? Price { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string SaleStatus { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string BuildStatus { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Information { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public DateTime UpdateDateTime { get; set; }
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string UpdaterId { get; set; }

}
```

Figure 6: Structure Table Components

After the green checkmark button is clicked, the form and rotate button disappear and a ‘Save Changes’ button becomes available. The ‘Save Changes’ button is the link between the MS SQL Server database, and acts as an Ajax post request. The Structure table, which may have one or many objects in it, is saved to the database (see Figure 7).

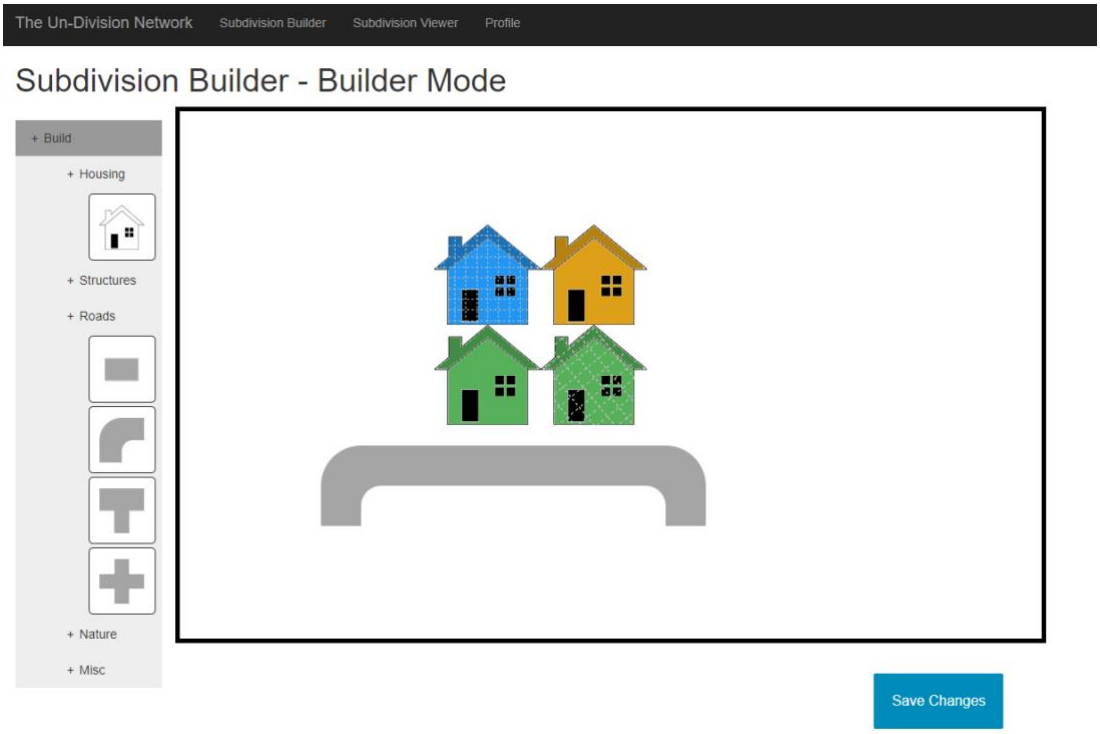

Figure 7: Builder Mode with Subdivision Ready to be Saved

Then the Viewer Mode page needed to be built, which is essentially a non-interactable Builder Mode page (see Figure 8).

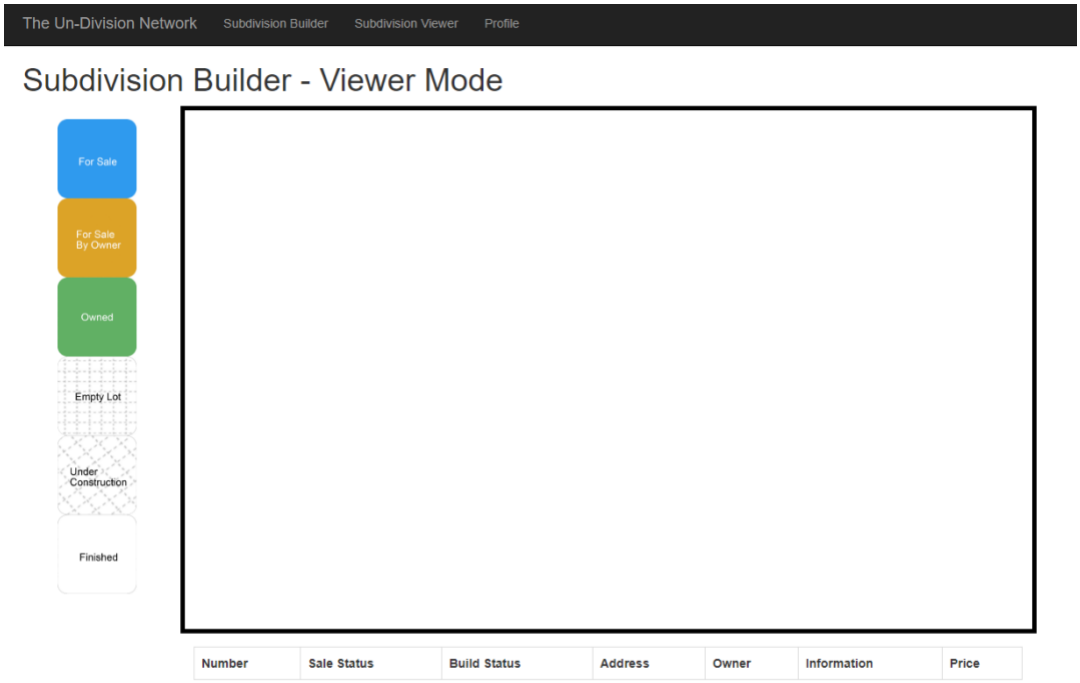

Figure 8: Viewer Mode with No Subdivision Saved

The intent was to allow the user to be able to make informational edits on this page, and this is where more problems came into play. As C# is running server side, and JavaScript is running client side, the vision of allowing users to edit information on the page became out of reach quickly. The previous map on the builder page was drawn entirely using JavaScript, while the map on the viewer page needed to be drawn user Razor code. Adding event handlers in the previous fashion was now impossible, and instead opted for a chart at the bottom of the page displaying all object attributes. The only problem remaining was that it is not entirely apparent which set of attributes belonged to which object on the map. Many attempts were made to mark or label homes and streets populated with Razor code, but nothing seemed to work. This feature is certainly possible to implement but has proved troublesome in the frame of the project. So, in a final effort to help the user distinguish homes and structures in the table below, a visual key

was added on the left side bar that can help the user reasonably deduce which attributes belong to which object (see Figure 9).

Number	Sale Status	Build Status	Address	Owner	Information	Price
1	For Sale	Empty Lot	250 Bald Eagle Lane	Jacob Darabaris	Great Family House	120000
2	For Sale By Owner	Finished	515 S Lincoln Park Dr	Michael Jones	Comfortable Single Home	90000
3	Owned	Under Construction	451 S Weinbach Ave	Jacob Darabaris	3 Bedrooms, 2 Bath	100000
4	Owned	Finished		Joseph Tiernen	Heated Flooring	110000

Figure 9: Viewer Mode with Subdivision Saved and Attributes Visible

One may notice that in comparing Figure 7 and 9, there is a slight offset in the positioning of the houses. While not drastic in this particular example, the oddity is rooted in the same limitation of jQuery previously mentioned. As the position of the object in CSS is pulled from the builder page to reproduce in the viewer page, any positioning errors become readily apparent in the viewer. As stated, each object appended to the initial place of the last child in a parent element, regardless of whether that last child has since moved. Each object appended

though has a relative top and left CSS position of zero, which resulted in out of bounds and displaced positions. The subdivision is saved in the database and is available for viewing until the table is emptied. Much of the remaining time was spent testing and finding different ways to break the application, and then address said ways.

While not the most functional design, this implementation was chosen due to time constraints after committing to a certain design path. An alternative path to address the jQuery positioning problem was considered where the frame would instead be composed of a grid of divs that objects could jump between. This would take some of the issues away with the offset of saved objects. Considerations were also made to do some of the front-end design with C# Razor rather than JavaScript, but documentation was much scarcer and the technology less adopted.

RESULTS

The final product came out to mostly everything the specifications envisioned, minus the profile system and un-editable saved subdivisions. The map builder and front end is relatively robust, well-designed, and very functional. The database back end is very limited but serves its purpose with room to grow. While it is only one piece of the network of epic proportions planned in the proposal phase, the subdivision builder serves as a solid foundation that all future components can branch from. For now, building a subdivision online and saving it is finally a reality.

CONCLUSION

The Subdivision Builder is truly just the first chapter in a huge undertaking with tremendous potential for development and capitalization. It does serve a niche purpose for HOA

members not currently available and does so reasonably well. Subdivisions can be designed and saved in an interactive way with ease of access to all. In order to realistically deploy and market the application though, the Subdivision Builder will need a much larger back end focus, as well as fine tuning to the positioning system.

REFERENCES

1.FreeAdvice Staff. "What Is a Subdivision?" *FreeAdvice*, real-estate-law.freeadvice.com/real-estate-law/zoning/subdivision.htm.

2. Debbie Braccio (Realtor at McColly Real Estate), interviewed by Jacob Darabaris via phone, September 24 2018.

3. Tanya Moffit (President of the HOA at the Preserve), interviewed by Jacob Darabaris via phone, September 10 2018.

4. Derek Hoots (Builder for Nantucket Cove), interviewed by Jacob Darabaris via phone, September 20 2018.

BIOGRAPHY

Jacob Darabaris is a man with more creativity than he can possibly express through his limited means. He does not enjoy programming, rather he moderately enjoys what he can achieve through programming. The next chapter of his life starts with a Software Engineering position at Raytheon in Indianapolis this Summer. With a steady stream of income, he intends to expand on the small but considerable success he has experienced as a seasoned computer technician in college and officially launch his own computer repair and refurbishing business online. Other notable plans include self-writing and recording a metal musical album; he has about 10 published songs to gus name, but nothing incredibly notable or in this genre. And in regard to actually using his degree he will be repaying the next few years, he hopes to produce a 3D video game with a small team of friends, and handle everything internally including character and map design, music, physics, and the crux of the actual programming functionality. He is sure the experience will be most akin to this one—he has no idea what he is getting into.