Friend Finder

University of Virginia CS Capstone Spring 2021

Pablo Weber
University of Virginia
pw9ev@virginia.edu

Matthew Hunt
University of Virginia
mjh3nv@virginia.edu

ABSTRACT

Many universities offer dormitories to incoming college students to facilitate the transition to a new home. This provides a natural ecosystem for connecting with peers, as students share the same living-space. When students move out of dorms and into off-campus housing there is a noticeable disconnect between students as they become more spread out and isolated in distant apartment complexes. Not only are apartment complexes generally far from campus and each other, the nature within an apartment complex is quite different from dorms; students go from knowing everyone on their floor to possibly only the 2 or 3 people they live with, if they live with anyone at all. From personal experience, apartments do little to enable residents to meet and interact with each other, especially in comparison to university-run dorms. On top of this, the current pandemic disables residents from physically interacting with each other. And so it is reasonable to believe that a virtual, technological social environment could fulfill this apparent void in student life.

Current social networks, applications, and media are targeted at enabling people to find who they know, stay updated, and share information about their lives. People can to an extent also find new people that might know someone they know. Platforms like Facebook allow users to create virtual groups and subscribe to specialized topics and people. However, what most current social apps are missing in order to solve the aforementioned problem is taking current location and residence into account. Modern dating apps allow people to find potential matches based on relative distance and various matching algorithms. These apps solve privacy and security concerns by obfuscating precise location. An app for off-campus university students would function in a similar way, allowing students to connect based on the combination of building and distance, but for the purpose of forming platonic relationships. The web application would be built using a modern front-end framework, such as React, and a cloud-based backend infrastructure supporting a social network. This application would help students living off grounds to connect with nearby people they would otherwise not be interacting with.

1 Introduction

Student housing is usually depicted and thought of as being a community where young and eager students are able to meet and connect. This is true for most people when living in dormitories, in fact, it's where many students make their first few friends in college. Indeed, you are more often than not paired with a roommate, common rooms are provided and widely used, you share the laundry room with others, and people tend to leave their doors open in order to meet new people. Because of all of these shared spaces, it is very easy to meet people and make new friends. However, when students move out of university-provided housing and into apartments, many find themselves having a hard time meeting new people. This is because apartment buildings are inherently private by nature. There are very little shared spaces, and almost no one leaves their door open due to the lack of trust that was once present when living in dorms. The only time one sees their neighbors is when passing by them in the hallway or when sharing the same elevator as them. There is very little opportunity for meeting new people. In fact, the only way that one can currently actively meet their neighbors is to knock on their door. This is problematic for a few reasons. First of all, the neighbor might not even be home. Second, you could be inadvertently disturbing your neighbor, say for example if they were sleeping or even taking a test. Third, you have no idea who could be behind that door, they could be someone dangerous or someone you simply wouldn't want to be friends with. Lastly, other than your direct neighbors, it might seem strange to knock on random doors in your apartment building.

Having lived in apartments for 3 years now and experienced difficulty making friends with those living closest to us, we felt the need to create a solution. We wanted to create a platform that allowed people living in apartments to connect specifically with others living in close proximity to them. This would be based not only on distance alone, but also on the specific apartment complex, floor, room number, or personality matching algorithms.

2 Background

We chose to create our platform via the format of a web app. We chose to go this route instead of a mobile app because a web app does not need to be downloaded or manually updated, is easier to maintain, does not require app store approval, and most importantly, is platform independent and thus much more accessible – which is what we really tried to focus on.

The app is built using a React front end (React being a JavaScript library), and a Dgraph backend/database. The app is hosted on Heroku. The client communicates with the database through a GraphQL API generated by Dgraph.

3 Related Work

A similar idea within the same social space is the popular app Bumble. While Bumble is primarily known and used as a dating platform ("Bumble Dating"), it does also have a lesser-known Bumble Social section which allows users to connect with each other for purely amicable reasons. While this is similar to our application, it differs in that it matches people purely based on raw distance. Our idea is to match people based on a combination of factors such as apartment complex, floor, room number, ensuring that users are able to meet people that are specifically residing in the same building or even floor as them. Additionally, we chose technologies that make it possible to match users on a variety of other factors such as similar interests and shared connections. Many people in the business sector make connections on LinkedIn through recommended friends and degrees of connection. LinkedIn relies on an internal graph-based database that allows deep levels of connection degrees to be computed efficiently. Database read times and joins have been though about for decades now, with many companies adopting NoSQL technologies optimized for reads. In a social network like Bumble or LinkedIn, there is not much data that constantly needs to be updated, but rather the relationships between them, so it makes sense to model off native graph technology. Bumble Social is not widely used because it is simply overshadowed by the main purpose of the app, Bumble Dating. What's more, the Bumble Social section is fairly hidden and hard to get to within the app. It is for this reason that we wanted to create a dedicated, apartment-based application to meet new people. Often times users only use applications for their primary purpose and do not realize lesser emphasized features.

4 System Design

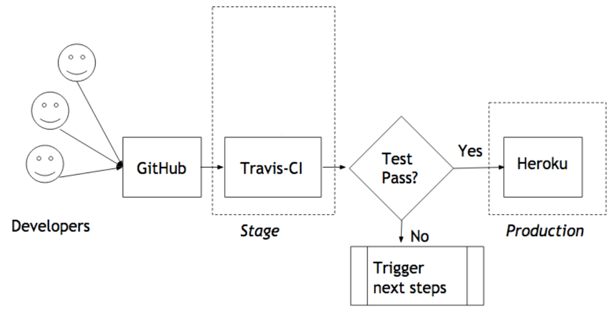
The product of the project the team worked on is a publicly accessible web application. This could also be deployed to an end-user's mobile device with a few changes to the front-end and networking layer. All the data and application-layer processing belonging to the application runs on servers in the cloud. Like many modern-day internet applications, the system design relies heavily on the ideas behind the client-server architecture. Client computers request information necessary for the application from the servers, which is then displayed and formatted once received. Some implementations differ on which computer creates the user interface file, but applications using this architecture will request and fetch application and user data from a backend server and database. It is therefore natural to break the system design commentary into frontend and backend sections.

The data for the application is stored in a graph database, Dgraph. Graph databases are good for hierarchical, directional data schemas. Dgraph takes advantage of the power of native graph processing by precomputing relationships into an index-free adjacency file in the database engine's memory. This means the graph's nodes can be traversed by going through these adjacency pointers and that query times are independent of the graph's size. Rather, they are instead proportional only to the amount of the graph searched. This makes native graph databases an excellent choice for social networks, where the application may be required to find friends-of-friends. Additionally, it can be useful for making recommendations based on items' similarities, which could be how many friends two users have in common. Furthermore, the database can precompute the geo-spacial distances between nodes containing location data in an effective manner and use that in combination with graph algorithms to find shortest paths. Based on these features, a NoSQL graph database was chosen over a traditional RDMS system. The database is used to store user and resident information. A user can put in their living space, and it will form a relationship between the two in the database. The system is also set up to allow users to find and add friends. This could be through geo-search, a similarity-based algorithm, or through mutual friends.

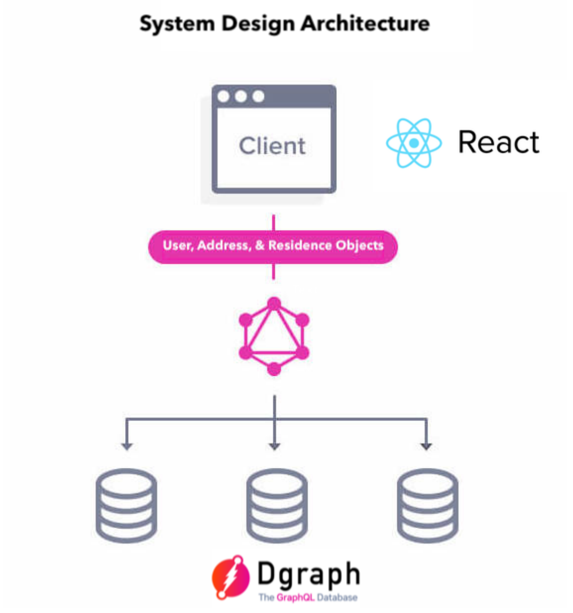
The second part of the backend is the GraphQL server. This API layer defines a data schema for the system to use and a mechanism for the client to retrieve data in a RESTful manners. Dgraph requires developers to simply define in GraphQL, along with Dgraph directives, which types of data to use and how they are related. Once this is defined and deployed, the database automatically starts storing data in the correct structure, and CRUD data accessibility query patterns are generated. Updates to the schema are frictionless and much easier to manage than most SQL systems. The GraphQL server sits in-between the client and server to relay data requests and responses in a simple manner. GraphQL is seen as an improvement to plain REST systems because it allows clients to only request for specific data fields of resources it needs, rather than entire resources. The client adheres to the syntax that matches the data schema, makes calls to the API, and receives JSON data in return. The API calls can create, update, delete, and read data over HTTP or even gRPC. Additionally, it can establish a subscription connection using the WebSocket protocol, which allows a client to receive updates to certain data without it requesting it. This allows the application to be real-time with live updates if needed. Authorization to access the API, and therefore the database as well, can be achieved by using a cloud service such as OneGraph or Auth0, or by using JSON Web Tokens (JWTs). For simplicity's sake, this was left out of the prototype.

The frontend client was developed using the React JavaScript library and the Apollo Client for React. React is a good tool for rapidly developing fast, reactive single-page applications (SPAs). Relying on modern JavaScript, React keeps track of client data and interface state. Once a device has downloaded the required HTML for the web application and the JavaScript functionality to go along with it, it does not need to download any more files for layout purposes. Rather than making direct database connections and returning a layout file based on that, the architecture handles rendering processing on the client computer, supplemented by requests for data through the API layer. Furthermore, data can be cached on the client computer using the functionality provided by the Apollo GraphQL client, which can save trips to the database. The client relies on a token-based authorization mechanism, where it stores access tokens that grant use to login authentication services and the servers storing the application data. Certain content within the application will not be accessible if the client does not have a token stored, and all network connections will be denied if a token is not valid. On top of the React and Apollo libraries, a variety of others were used such as Firebase – for uploading profile images to cloud storage, Facebook & Google social login – for quickly signing up for the application, Material UI – for pre-made Google Material Design-compliant user interface components, and Jest – a JavaScript testing framework.

The application prototype was developed by running all the servers locally. The Dgraph database and GraphQL engine was run from a local Docker container, while a React bundler kept source code updated and running on localhost. Source code versioning was managed through Git, and code sharing through GitHub. GitHub was connected to TravisCI for continuous integration, which itself was connected to Heroku for deploying the static web app.



Continuous integration and deployment with GitHub, Travis CI & Heroku


System Design Architecture
Client
React
User, Address, & Residence Objects
Dgraph
The GraphQL Database

5 Procedure

The first step in using the platform is signing up and creating an account. We used both Facebook and Google's respective SSO to ensure an easy and seamless sign-up process. The Google SSO in the prototype is strictly through virginia.edu Google email accounts. Once a user has an account, they would be able to fill in their apartment details: building, floor, and room number. Of course, the user would have the option to either show or hide any of the latter. In the prototype, users can also upload a photo of themselves to our cloud storage and choose a username. The user would also have the ability to add any of their roommates in the fully developed application. Next, the user can begin meeting people. After navigating to the "discover" page, a user will be shown a stack of "cards". These cards each represent a potential friend that is near – either in the same apartment complex, or even the same floor. Or it could show similar users based on connections or interests. A card would show relevant information about a person in the form of a "bio". The application might have preset "ice breaker" questions to answer or options to choose from, which have been shown scientifically to make meeting new people easier. The user can either swipe left or right on a card. Swiping left signifies that the user is not interested in meeting the person on the card. On the other hand, swiping right means that the user would like to connect with the person. If a user swipes right on a person that also swiped right on them, then a match is made. The user will then have the ability to message the person they matched with. Users can also view a match's roommates and connections, and can choose to privately message them if they accept the request. Users can also view the other users who live on a match's floor. This not only helps the user to familiarize themselves with the faces on their floor, but also helps build a sense of community within the apartment complex and virtually within the app.

If at any point a user feels uncomfortable when messaging another user, or for really any reason at all, they have the ability to unmatch with that particular user. This means that they match will be deleted, and the user will have no further access to the information of the person they unmatched with. In the "profile" section of the application, users can update their profile picture(s), bio, phone number, address, and roommates. The ability to update one's address is important because students tend to change apartment complex every year. Users also have the ability to filter the users that are shown on the stack in the "discover" section. These can be filtered by not only building and floor, but also things like hobby, school (ie. SEAS, College…), and age. Perhaps they could select a matching algorithm for the database to use.

6 Results

To make sure that the application was functioning properly, we had a few of our friends sign up and make accounts so that they could experience it on their end. They were able to see accounts that we pre-populated into our database, but we did not have enough time to put in the connection functionality. As of now, the prototype just shows all the users in the app in a stack of cards, with another screen for updating basic profile information. As of now, the React Webpack dyno on Heroku is hosted for free and takes a few seconds to spin up from sleep. The beta testers noted that the UI was simple but effective, and that they thought the app could potentially be a great way to meet people in one's apartment building. We could imagine how much quicker it would be to meet, see, and talk to people that live nearby than the way it is during the pandemic.

7 Conclusion

In conclusion, the aim of our project is to make it easier for people living in apartments to meet other people living in the same area as them. We accomplished this by creating a web application that shows a user potential friends nearby that may have something in common with them. From there, the user can choose to connect with the people that they would like to become more familiar with. This improves on the current way of meeting one's (direct or indirect) neighbors which involves either knocking on their door or hoping to meet them in the hallway or elevator. It allows users to not only expand the number of people that they can become friends with, but also makes the process much smoother and less intimidating, as everything is virtual. Our platform improves on the only other viable option out right now, Bumble Social, by matching users not only based on raw distance, but also by specific apartment building, level, and room number. This allows the user to specifically meet people who reside in the same apartment complex as them, ie. their neighbors. We believe that our platform will help solve the problem of social isolation that is common in apartment buildings, and help people make more friends with the people that live nearest to them.

Here is a link to the website, which takes a few seconds to start up: https://cs-capstone.herokuapp.com/ . If someone is curious about the prototype database schema, here is a link our fully unprotected GraphQL endpoint: https://long-rain.us-west-2.aws.cloud.dgraph.io/graphql .

8 Future Work

If we had more time to work on the project, there are a few features that we could add. We could allow users to browse by building or apartment, so they only see people living in their own building or people across the street. Once matched with someone, we could incorporate a virtual video chat feature, which seems to be trending in technology. We could organize users into group messages or matches as well. Users within a group within a building could manage sharing real resources like cooking materials or social events in real life. More research could be done investigating what visualizations of profiles lead to the most friendships. Research could also be done into what graph database algorithms are good for determining the users that show up in the application.