

Software Systems Engineering Capstone Project Report Document

Pencil.

Paul Hewitt & Ian Quach
200343079 & 200367522

Section 1 - Project Description	4
1.1 Project	4
1.2 Description	4
1.3 Revision History	4
Section 2 - Overview	5
2.1 Purpose	5
2.2 Scope	5
2.3 Requirements	6
2.3.1 Estimates	7
2.4 Business Model	7
2.5 Competitive Advantages	7
Section 3 - Design Methodologies	8
3.1 ScrumBan Process	8
3.2 User Story Map/MVP Management	10
3.3 Design Patterns	11
3.4 Customer Personas	11
Section 4 - Software Architecture	12
4.1 Tech Stack	12
4.2 Angular	13
4.3 Serverless	13
4.4 Backend	13
4.4.1 S3	14
4.4.2 API Gateway	14
4.4.3 Lambda	14
4.4.4 DynamoDB	15
4.4.5 CloudWatch Logs	15
4.5 REST API	16
4.6 Data Models	19
4.7 Authentication	20
4.7.1 User Roles	21
Section 5 - Frontend Design	21
5.1 Application Structure	21
Section 6 - User Stories	23
6.1 Basics	23
6.2 Andy/ChungCuts	24

Section 7 - User Interface Design	25
7.1 User Interface LoFi Designs	25
Section 8 - Business Decisions	31
8.1 Monetization	31
8.2 Statistical Analysis	32
Section 9 - Testing Plan	32
9.1 Code Testing	32
9.2 User Testing	33
Section 10 – References	34

Section 1 - Project Description

1.1 Project

Pencil: Scheduling Made Simple

1.2 Description

Pencil is a software as a service application (SaaS), that simplifies the once complicated task of customer scheduling. By allowing business owners to easily create their own online schedule, potential customers can see available time slots, and book desired appointments. Users are able to find other businesses that are also using Pencil, making it a one of a kind, online business aggregator. Lastly, Pencil provides booking data analytics, and visualization to business owners. This gives owners valuable business intelligence, and allows them to see what is, and what isn't working.

1.3 Revision History

Date	Comment	Author
2019-10-30	Initializing the document, populating section 2	Paul Hewitt
2019-11-22	Populating document for Project bazaar	Ian Quach
2019-11-30	Adding Business Sections	Paul Hewitt
2019-12-20	Adding And Compiling Scrap Notes	Paul Hewitt
2020-01-09	Adding Sprint/MVP Timeline	Paul Hewitt
2020-01-25	Adding in Scrumban discussions	Ian Quach
2020-02-10	Adding in jot notes	Ian Quach
2020-02-15	Adding System Architecture Information	Paul Hewitt
2020-03-14	Adding the REST API section	Paul Hewitt
2020-03-27	Adding Auth Section	Paul Hewitt
2020-03-30	Adding User Story Maps, Application Structure	Ian Quach
2020-04-01	Adding Design Patterns	Paul Hewitt
2020-04-03	Adding Business plan discussion	Ian Quach

Section 2 - Overview

2.1 Purpose

We are beginning to see more and more entrepreneurs and small businesses pop up everyday. It is important that this market is supported by scaleable, and simple technology. Pencil provides any industry that requires scheduling with a platform that is enjoyable for both customers, and business owners alike. Applications that currently exist lack key features that are necessary for an enjoyable experience.

With Pencil we include basic gamification features, third party integration (Facebook and Google Maps), and customer tracking. The gamifications aspects help motivate continual involvement and use by the customer with the business. Maintaining a customer's persistent memory allows the business to track their needs and wants. By providing graphs to each owner, it allows them to quickly identify trends at a glance.

The intended audience of Pencil is both business owners and their customers. Large, and small businesses alike will be able to leverage the vast array of features designed to streamline the scheduling experience. Customers will be able to use Pencil to find specific businesses in their area, and then book appointments with each one. Business owners will have additional access to tools in order to keep track of, and reward their customers. These features will be encapsulated away from the customer, included only in the business owner dashboard. It is hoped that one day Pencil's data could be available in API form, allowing developers to work with, and manipulate the data, similar to services such as Google Calendar.

2.2 Scope

Pencil is an all encompassing customer scheduling, and management application. On top of this, it will work as a business aggregator, working like a phone book, in which users can search for businesses, and then book desired appointments.

All the data used in Pencil will be user generated, and it does not pull in any outside business data. Data will be tagged by the user, in order to be searched more efficiently. Pencil will integrate with certain third parties, in order to streamline the user experience. Social media integration will function as user login and account management. Square can be integrated to handle payments. Amazon SNS will be the platform used to send out SMS messages to customers.

2.3 Requirements

Pencil needs, based off of our interview with our client, to be a lightweight, easy to use client. Since the client will be a nontechnical person it also needs to be very low maintenance which requires sound software development principles. The customers will also need to be able to access the clients services at all times to book appointments. In tandem with that, a web application that is accessible anywhere serves those needs. The entrepreneurs will also need a secure login for their users as personal information may be stored on our service. Another important feature is the gamification elements which support the users to remain with the business. Social media integration will also help bolster the continued support from the customer. On the UI/UX front, the web application will need to be modular or easily configured to fit the client's needs quickly. A separate view for both customers and owners is necessary.

2.3.1 Estimates

Description	Hrs. Est.
UI/UX prototyping	8
Frontend	25
Misc.(Meetings, etc)	10
Gamification aspects	1
Data visualization and analysis	5
Backend/AWS	45
Facebook API Integration	2
Google Maps AI Integration	5
FullCalendar API Integration	16
Documentation	40
Total:	155

2.4 Business Model

The business model for Pencil will be a 'Freemium' business model. You are able to use a free version of the application but it will lack certain quality of life functionalities. Using the premium version will include all of those features. The business model will be discussed more in-depth later in the document.

2.5 Competitive Advantages

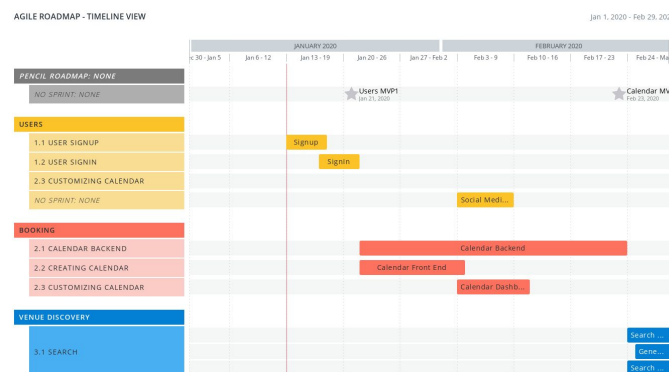
Currently the main competitors consist of SquareSpace and YouCanBook. SquareSpace offers a template based website builder that can be used to book clients into a calendar and take payments. Beyond the use of a basic day planner it does not offer any of the innovations that Pencil offers. No SMS alerts, third-party integration, customer tracking, or gamification elements. YouCanBook is a bit more robust and friendlier in the UX department. YouCanBook offers SMS alerts and third party integration, like Facebook and Instagram. While it does offer quite a decent scheduling

application there is still no customer tracking. Pencil intends to offer all of that with the addition of contactless booking. The gamification will motivate clients to continually use the business earning rewards that the business owner can dictate. The customer tracking allows for a persistent memory of the client's needs and wants so there is no guesswork if the client gives vague requirements. Finally, none of the aforementioned products offer any feedback or analysis on your schedule. Pencil aims to provide valuable insights to owners, allowing them to create smarter, more efficient calendars.

Section 3 - Design Methodologies

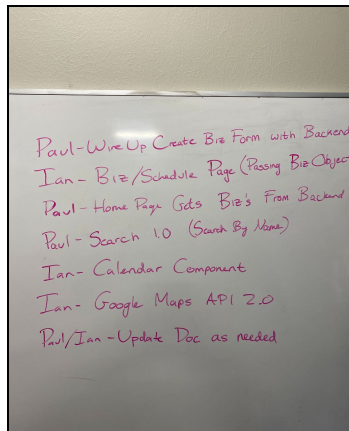
3.1 ScrumBan Process

Due to our small team size, two members, we are able to use an agile design process. In practice, each feature was specified, designed, developed, and tested together as a team. Before starting coding, we got together and scrummed to determine what tasks needed to be completed for the day. The features were all determined using the Pencil feature roadmap. Over the course of the project the roadmap was updated and corrected to properly reflect the scrums goals.

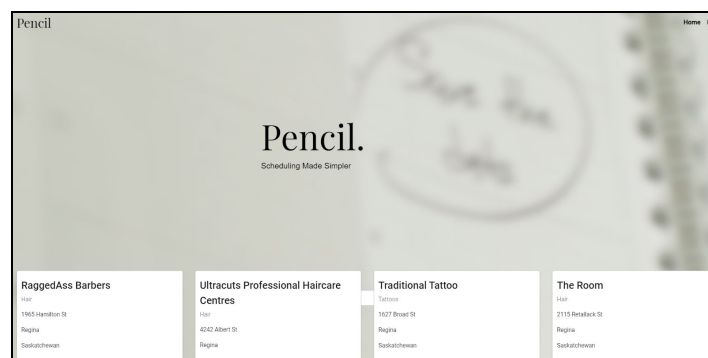


We began with a stand up meeting discussing all the aspects of the features that would need to be done. Once the discussion was completed, the feature would then have a lofi design created to further help with visualization. Since we both had different strengths and different goals we wanted to achieve with the project we would break off and tackle components that aligned with those strengths and goals. Now with proper

visualization the coding portion could be completed, a 1.0 of webapp components would be created. The components would vary from entire frontend pages, backend services, or investigating APIs that would be used in the project. After the 1.0 of the component is completed we would then rejoin each other and critique the coding decisions. Paired programming was then used for redesigning the different features to finalize the design for MVP 1. An example of this scrumBan process can be seen with the included pictures. Initially, a plan for the day is written up, covering the different features that we wanted to cover. The figure displays part of our scrum planning for the 7th of March.



The LoFi designs in Section 8 - User Interface Design were used during the scrum for the prototype designs when programming began. During that critique and paired programming issues were corrected with the design decisions, for example certain issues with the responsive user interface design were overlooked during prototyping. This can be seen in the following figure. Once the following issues were corrected the feature would be checked off as completed.



3.2 User Story Map/MVP Management

Themes	Epics	MVP 1 Tasks	MVP 2 Tasks	MVP 3 Tasks
--------	-------	-------------	-------------	-------------

Basic user functionality	
Login	Business Sign up
Login page	Basic HTML/JS business sign up
FB only signup, temp button	name, email, password, business type
pull info from fb login	attaching business with owner user
User profile information populated with facebook info	
Google Maps integration to display local businesses	

Business Functionality	
Business searching	Calendar
search 1.0, parse businesses based on business name	Calendar scheduler displaying appointments
Design DB objects diagram	deleting the appointments made
Search with business name, type, location	
Use unique business ID to link user with business	
Search with business type, location	draggable events on calendar for better UX
	Expandable to multiple years, currently operating only on 2020-2021
	edit appointments to change name and times

Business Functionality
User Profiles/User Homepage
Currently displaying owned businesses for owners
Display upcoming appointments

UI/UX
CSS/Website flow

Improve UI/UX

make site look less ugly

3.3 Design Patterns

Two key design patterns were implemented with the development of Pencil. Pencil's backend is a REST API, which is further explained in Section 4.5. For the frontend, the MVVM design pattern was followed. MVVM stands for Model, View, and Viewmodel. As we used Angular, we were at the mercy of its functionality and limitations, but Angular's component system can be interpreted as MVVM. Pencil uses Angular services to handle all of its data. Because of this data handling, some consider an Angular service to be similar to the Model, or business logic. Angular components use templates to display pages to users, rendering html and css files to form webpages. These templates can be interpreted as the View, as this is what the user sees, and interacts with. Using Angular's two way data binding, the View is able to communicate with the component TypeScript file. This would make the TypeScript file the ViewModel, as it is in between the View and Model. In Pencil's case, this ViewModel uses the Angular service to handle the data, and then serves said data to the view, completing the definition of MVVM.

3.4 Customer Personas

Pencil had identified two major customer personas before development was started. By clearly defining these two personas, it further helped to identify some required functionality, or functionality that might be missing. With these two personas defined, it allowed us to define two user roles, which are explained in Section 4.7.1

Name	Andy - Business Owner	Steve - Looking for a new barber
Goals	Andy is looking for a new, online product to handle his scheduling. He is looking for something lightweight, that users can quickly visit, and book an appointment	Steve has been looking for a new barber. He would like to support a local business, one that is not too far away
Motivations	<ul style="list-style-type: none">• Would like to get more customers to book with him faster, and more efficiently• Would like an intuitive, easy to learn interface	<ul style="list-style-type: none">• Would like to see a comprehensive list of barbers in his area• Needs a service provided to him by a professional
Frustrations	<ul style="list-style-type: none">• His current system requires customers to create accounts before booking• His current system does not allow for custom time slots• His current system does not track users	<ul style="list-style-type: none">• There is no cohesive list of local service providers• Making accounts on other scheduling apps is tedious, and it is annoying to keep track of login information

Section 4 - Software Architecture

4.1 Tech Stack

Pencil is built as a SaaS web application. Using an Angular frontend, and an AWS backend, we are utilizing an industry standard cloud platform, combined with one of the world's most popular frontend frameworks. The backend itself is programmed using Go. We decided to use a modern, powerful tech stack for a couple of different reasons. Both AWS and Angular are extremely powerful, allowing you to build small test applications, up to industry ready, scalable software capable of handling millions of daily active users. We also wanted to consider security. AWS is one of the most secure platforms in the world, and we wanted to trust Pencil's backend with one of the world's

leading cloud platforms. Programming with security is a fundamental skill for any software engineer, and crucial in the real world.

4.2 Angular

Angular 8 is used as our frontend framework. This is a TypeScript based framework, which uses components instead of controllers, thus losing the concept of scope. Angular was chosen due to developer familiarity with the framework, as it would allow for rapid prototyping. Angular can be extremely powerful for large, enterprise ready applications, which would be Pencil were to ever be operating at scale.

4.3 Serverless

Using the Serverless Framework, we were able to make Pencil completely “Severless”. Again, this framework is industry standard, trusted by large companies such as Expedia, and Electronic Arts. There are a couple of reasons why Pencil went Serverless. The first is to reduce the amount of time and effort involved in backend deployments. Serverless abstracts away most of the tedious work normally required to deploy to a cloud platform like AWS, for example configuring Trust Policies. This allowed Pencil to be rapidly developed and prototyped, which worked perfectly with our Agile development practice. Another reason is scalability. If Pencil were to scale suddenly, Serverless would be able to handle the sudden influx, and the application would see no real hit in performance. Serverless also has the advantage of working well with AWS right out of the box, which is the cloud provider Pencil is powered by.

4.4 Backend

Pencil’s backend is written in Go, sometimes referred to as GoLang. Go is a very new language, first released in 2009. It is a statically typed, compiled language, with syntax similar to C. There were a couple of reasons why we used Go for our backend. The first, is it is a very popular, upcoming language, making it an invaluable skill to learn before heading out into industry. Go can also compile into .bin files, making it very useful when deploying to AWS. Finally, Go integrates well with third party platforms. It

has libraries that are ready to use out of the box that makes backend work a breeze. We used AWS' official Go library for writing our backend. Our backend is a REST API, with further explanation outlined in Section 4.5. Pencil utilizes AWS to host the REST API, as well as hosting the database, DynamoDB. The exact AWS Services used by Pencil are found below:

4.4.1 S3

Amazon Simple Storage Service (S3) is an online object storage service. Data of any kind is able to be stored in "buckets", where it is then able to be accessed by any of the other Amazon Services. The Pencil backend services are written using Go. By using a makefile, the Go files are compiled into .bin files. These .bin files are then uploaded into a S3 bucket, where they are accessed by API Gateway.

4.4.2 API Gateway

API Gateway is the service that allows us to build, manage, and deploy our REST API. This service is essentially where the 'magic' happens with regards to the Pencil backend. API Gateway unpackages our .bin files from our S3 bucket, and creates and deploys our API. Contained in our API are all the different routes and endpoints that Pencil hits. When Pencil makes an HTTP request, it is sent to API Gateway. The request is then forwarded to the proper endpoint, which are actually Lambda functions. Each endpoint has its own Lambda function, which is configured using AWS Lambda.

4.4.3 Lambda

As Pencil is serverless, using Lambda was the obvious choice. Lambda allows you to run code without first setting up, or managing servers. Lambda functions will only run when triggered. This keeps computing time, and resources down. This way, you only pay for computing time that you are actually using, as opposed to paying for a server to be always online. In our case, a trigger is when API Gateway receives an HTTP request. Depending on the trigger, the proper Lambda function will initialize. When a trigger is received, Lambda will spin up a new instance of that function, and it will execute the desired functionality. This is what is called a cold start. That same instance

will remain available until no triggers are received for roughly 30 minutes. Each of these Lambda functions interface with the appropriate database, which is another Amazon Web Service. The service we use is DynamoDB.

4.4.4 DynamoDB

The database we use is DynamoDB. Again, this service is serverless, maintaining one of Pencil's requirements. DynamoDB automatically scales up or down depending on capacity, and load. It is well known for its scalability and low response time.

DynamoDB is a NoSQL database, however it is still structured unlike other NoSQL databases, like MongoDB. Each table represents a collection of objects, like a user object, or business object. Each of these objects are required to have member variables or attributes, similar to a struct or class object. DynamoDB uses primary keys to identify each unique item. These primary keys can be connected to a certain item's attribute, and that can be defined upon the table's creation.

4.4.5 CloudWatch Logs

The final service Pencil uses from the AWS suite is CloudWatch Logs. This is just a simple logging service, that allows us to monitor and record how the other services are working, and interacting with one another. This can be extremely handy for debugging, as the logs will output the exact service that is throwing an error, and why. Furthermore, these centralized logs provide powerful tools, such as visualization in the form of graphs and dashboards, as well as log querying and filtering. Again, this can be invaluable, as the user can see logs for very specific use cases.

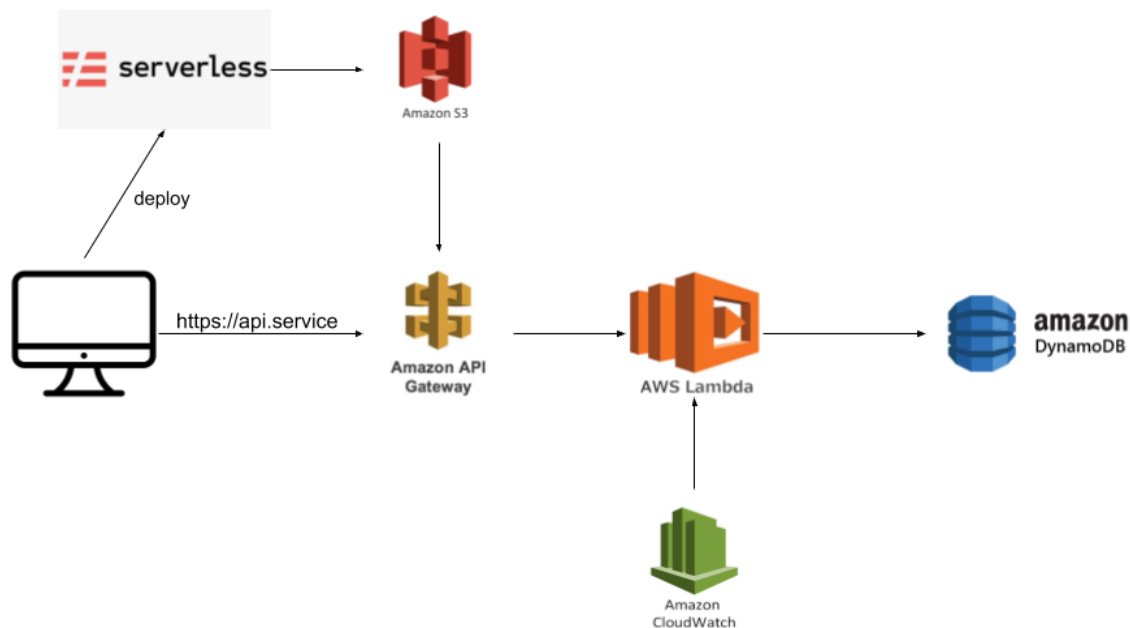


Diagram showing how the different Amazon Web Services interact

4.5 REST API

Pencil has its own REST API that is hosted on AWS. REST was chosen for its simplicity, and its speed. In order to make a request to the API, Pencil hits a URL, known as a route. This route is mapped to the appropriate service. This URL is secret, as there are security concerns associated with exposing your API URLs. Each route has different endpoints, associated with the traditional HTTP verbs: GET, POST, PUT, DELETE. An incoming request will have all of the standard data, including:

- Headers
- Method
- Path
- Params
- Body

This data is encoded into JSON, and the HTTP request is made. Our Go backend is able to read this JSON, either marshalling or unmarshalling as needed. Our API is capable of returning 2XX, 4XX, and 5XX HTTP status codes, indicating okay, client error, and server error respectively. We included this to help with debugging. If our

API was not working correctly, it is helpful to know if it is server side or client side before starting the debugging process.

Our API uses the Service design pattern, as opposed to MicroServices, or a Monolith. This was chosen to minimize the amount of Lambdas required in order for our API to run. Furthermore, less services mean less time deploying, which means more time developing and prototyping. The one downside of using the Service design pattern is that it can make debugging more difficult, as each service is responsible for multiple methods, as opposed to MicroServices. Each service is mapped to a Lambda function, that Lambda represents the route that will be hit by the client. The Lambda function identifies what HTTP method was sent to the route, and executes the correct endpoint.

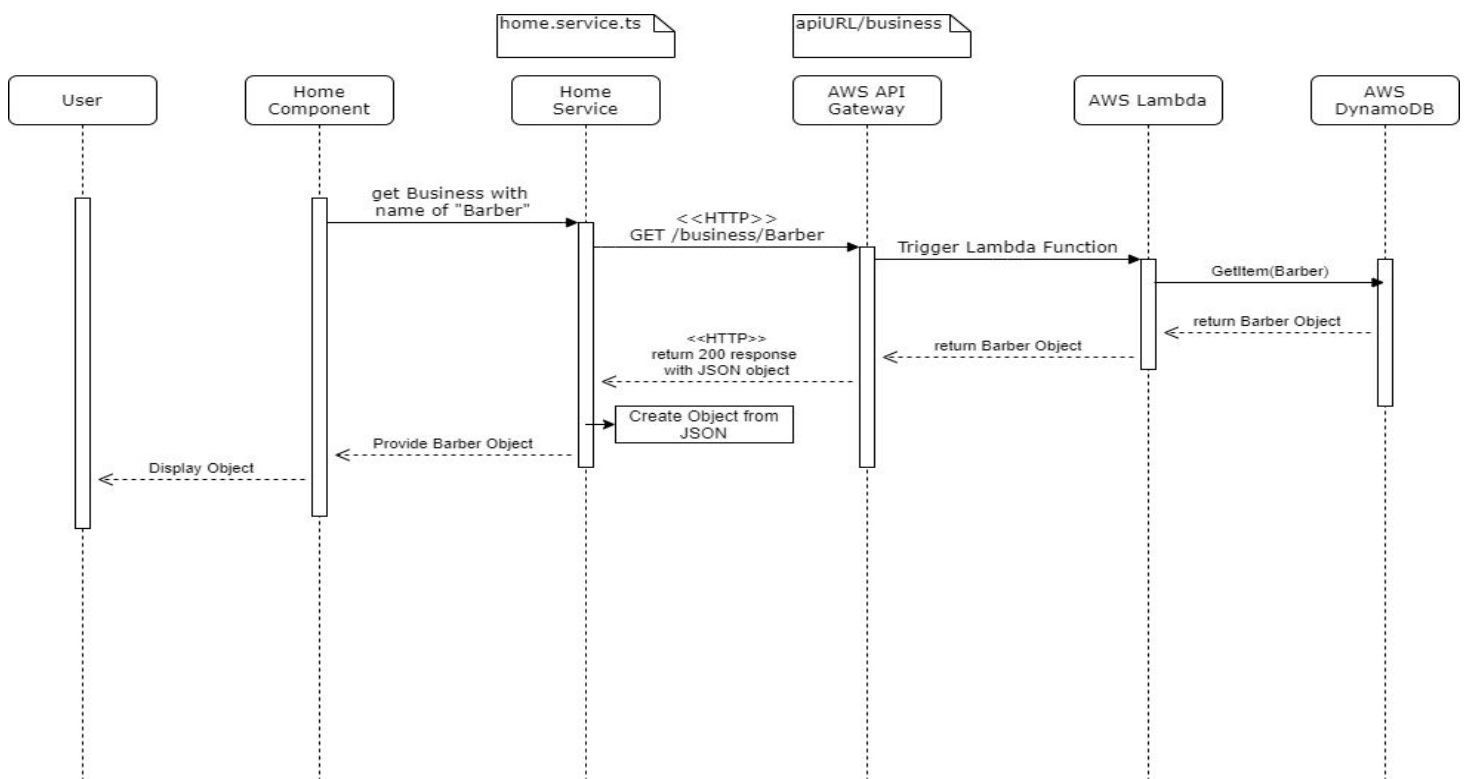
Pencil's API is currently composed of 4 services, each with its own corresponding route, and database. The services are as follows

- Service: Business
 - Purpose: To retrieve individual businesses from the database
 - Route: /business
 - Methods Supported: GET
 - Params:
 - GET: The name of the business you want to retrieve
 - Headers: Content-Type, application/json
 - URL: Pencil API URL
 - Body: N/A
 - Table: Businesses Table
- Service: Businesses
 - Purpose: To modify the database of businesses
 - Route: /businesses
 - Methods Supported: ANY
 - Params: N/A

- Headers: Content-Type, application/json
- URL: Pencil API URL
- Body:
 - POST: The business object you want to modify
- Table: Businesses Table
- Service: Schedule
 - Purpose: To modify the database of schedules/events
 - Route: /schedule
 - Methods Supported: ANY
 - Params:
 - GET: The name of the business whose schedule you are wanting to modify
 - Headers: Content-Type, application/json
 - URL: Pencil API URL
 - Body:
 - POST: The schedule object you want to modify
 - Table: Schedule Table
- Service: Users
 - Purpose: To modify and track users
 - Route: /users
 - Methods Supported: ANY
 - Params:
 - GET: The user's ID Number
 - Headers: Content-Type, application/json
 - URL: Pencil API URL
 - Body:
 - POST: The user object you want to modify
 - Table: Users Table

```
public createBusiness(businessForm): Observable<any> {  
  const url = `${environment.apiUrl}businesses`;   
  const headers = this.createHeaders();  
  return this.http.post<any>(url, businessForm, headers);  
}
```

An example of how we create a new business in the database. We are hitting the POST endpoint on the businesses route. We pass in the business object in the body of the request



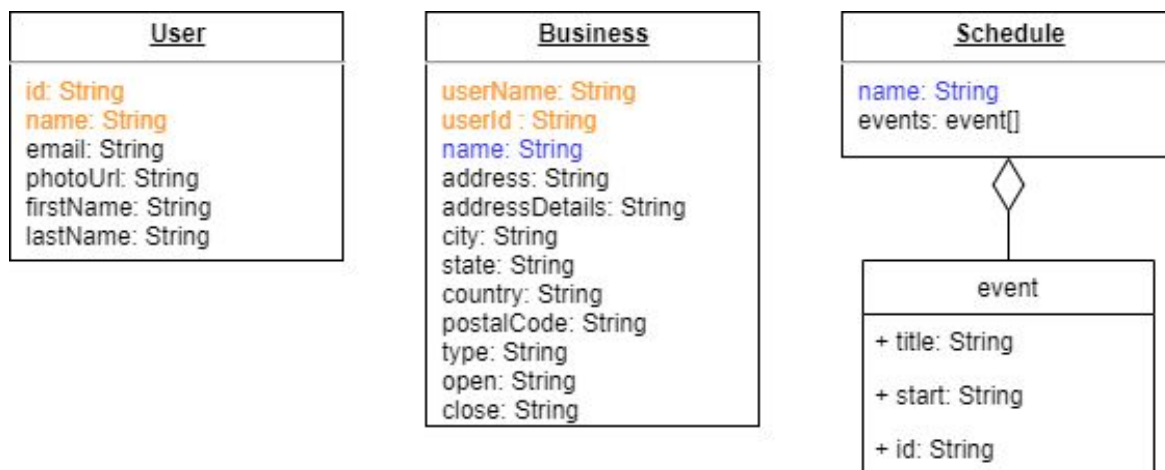
Sequence Diagram showing how our frontend interacts with our REST API. CloudWatch Logs have been omitted from this diagram

4.6 Data Models

Pencil has three main database tables. Each serves a different, unique purpose. The Businesses Table contains all of the businesses that have signed up with Pencil. The items in this table are Business objects, which are defined below. The Users Table

Pencil

contains all of Pencil's Users, and the Schedule Table holds the schedule and appointments for each business. Because Pencil uses Facebook's API to handle it's users, our user object emulates what the Facebook user object returns. The "primary key" for all of these tables together is currently the name of each business. In DynamoDB, this key is referred to as the partition key, and works a little different compared to a SQL primary key. Businesses are linked to users by their username and user id. Schedules are linked to businesses by the name of the business. In the future, our tables should be linked using a composite primary key, which consists of a partition key, and sort key. This would allow us to handle businesses with the same name much more efficiently.



Pencil's three primary models. Mapped values are color coded.

4.7 Authentication

Pencil has integrated with Facebook to handle all login, and user authentication. This way, only those with an activated, valid Facebook account are able to use Pencil. It was determined that using a major social media integration such as Facebook, not only legitimizes Pencil as a product, but may put users at ease as they do not need to give Pencil any usernames, passwords, etc. When a user clicks on the "Login with Facebook" button, they are redirected to a Facebook webpage. Once they login to the

webpage, they are redirected back to Pencil. The Facebook API returns the user object to our front end, and we send it to our database. Almost all of our user handling is done for us by the Facebook API, making this integration a very valuable addition to the application. In future iterations, it would be helpful to add more social media integrations, such as Google, to make Pencil accessible to as many users as possible.

4.7.1 User Roles

Pencil has two types of users. Customers, and owners. Customers consist of the average person using Pencil in order to discover a new business, and/or book an appointment. Owners are those that own a business that they would like to list on Pencil. The associated access rules can be found below:

Customers	Owners
<ul style="list-style-type: none">• Only have access to the home and booking views• Can book appointments at the desired business• Can browse different businesses	<ul style="list-style-type: none">• Have access to the owner view• Can customize and create the schedules corresponding to each owned business• Gain access to data analytics and visualization

Section 5 - Frontend Design

The frontend of Pencil is designed and programmed using the Angular 8 framework. Angular allows us to create Pencil as a single page web application. It allows the web app to be very dynamic generating pages and data in real time as the user requests it. This gives Pencil a smooth and clean experience to navigate through.

5.1 Application Structure

Though Pencil operates as a single web page application, it is split into several different components. The different components are responsible for the different screens and functions for the website. The website still has a main index.html that contains the web application. Angular uses metadata from components to define how the Angular template will modify the HTML before it is rendered for display.

The many components in the application are:

- Calendar
- Home
- Login
- Nav
- Owner
- Owner-calendar
- Signup

The different components will be rendered over the main HTML file. The Nav component is the only component that exists on top of every other component page. Links to the Home component and the Login component exist on the Nav. The Home component is what the users will initially see when they visit the website. It contains the search bar and different links. The links display the businesses that appear from search.

The Login component must be visited and used before the customer is able to book appointments. The business owner must also be logged in before they are able to register a business. To make the experience more seamless, the only way to login is to use a Facebook account. Additionally, this places the account security responsibilities on Facebook.

The Calendar component is used to contain the business info and schedule when a customer selects a specific business. The Owner-calendar is a variant of the Calendar component that only business owners will be able to view; this view allows them to edit appointments and book them. Additionally, in this Owner-calendar the owners will be able to view analytics on their businesses.

The Owner component is used as the owner's dashboard. From the homepage if the owner clicks on his name or profile image, received from Facebook, they will be taken to the Owner component. They will be able to view the businesses that they have registered. Clicking on one of the business cards will take them to the Owner-calendar.

The last page is the Signup component. To sign up a business all the fields will need to be filled out. A caveat of the Signup component is that the users will need to be logged in to be able to register a business.

Section 6 - User Stories

6.1 Basics

Customer

- As a customer I want my own login information
- As a customer I want to be able to see appointments offered by a service
- As a customer I want to book my desired appointment
- As a customer I want to receive notifications about my appointment
- As a customer I would like to prepay for my appt
- As a customer I would like to rebook easily
- As a customer I want to be able to make changes to my appointment or cancel if necessary
- As a customer I would like a way to contact the owner
- As a customer I would like to be rewarded for reusing a service
- As a customer I would like to see the various services offered around me

Service Provider

- As a service provider I would like to create appointments to be filled by customers
- As a service provider I would like these appointments to be customizable and flexible
- As a service provider I would like to be able to contact my customers
- As a service provider I would like to accept payment
- As a service provider I would like to track all existing/previous customers
- As a service provider I would like to offer rewards
- As a service provider I would like to limit who can book

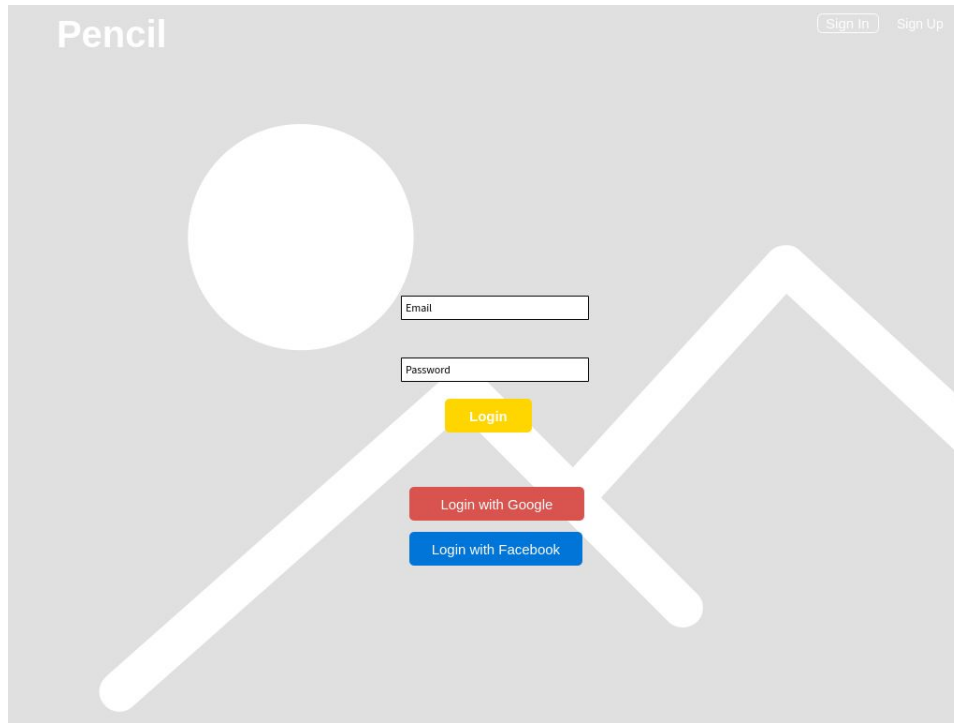
6.2 Andy/ChungCuts

- No login for users
 - Slow and tedious. Users should be able to book quickly, and efficiently.
 - Can use Phone Numbers to verify the users
 - Use Phone numbers to track the users
 - First, Last, and Phone number should be the only required fields
- Ensure different 'types' of time slots
 - YouCanBook only has one type of time slot
 - Would like to see:
 - Regular Appt
 - Hair Dye
 - Lunch
 - Personal
 - Etc.
- Be able to have control over his employees schedule as well
 - Essentially have some sort of hierarchy system in place
 - Owner of the business could have admin rights
 - Employees could have their own pages/schedules, the owner should have control over them
- User Tracking
 - See how many times someone has been in that month, how many times total, etc
- Be able to pre pay for appts
 - Would be great for last second cancels or no shows
- Automatically send reminder texts to customers
 - Can set and forget
 - Send a text one day before, one hour before, etc

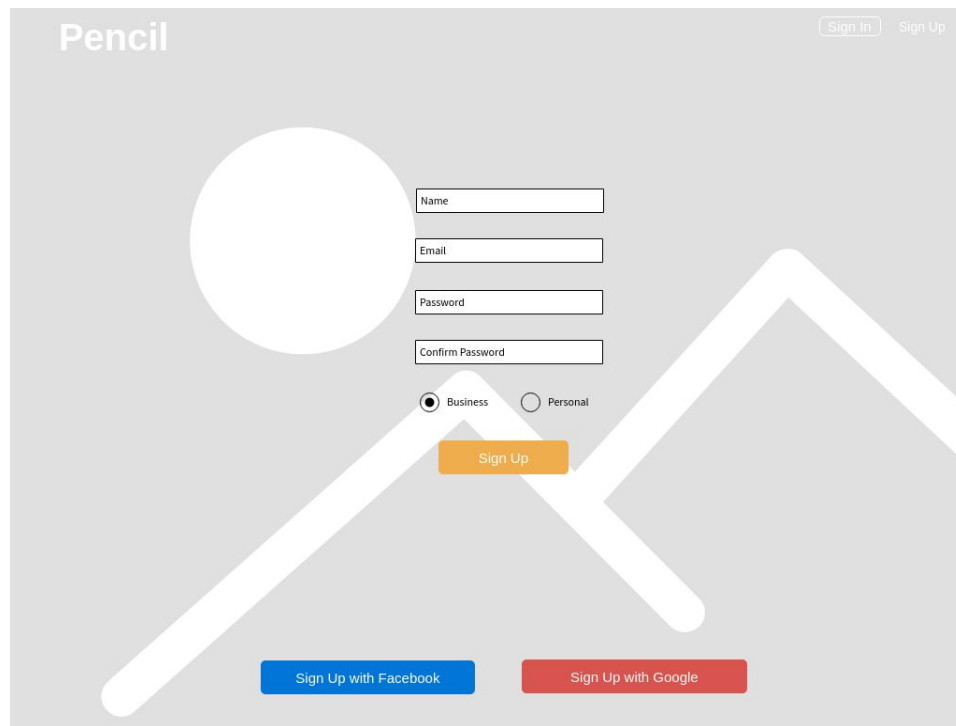
Section 7 - User Interface Design

7.1 User Interface LoFi Designs

Customer sign in



Customer sign up



The image shows a web form for signing up on the 'Pencil' platform. The form is set against a light gray background with a large white graphic of a mountain and a sun. The 'Pencil' logo is in the top left, and 'Sign In' and 'Sign Up' links are in the top right. The form fields include Name, Email, Password, and Confirm Password. There are radio buttons for 'Business' (selected) and 'Personal'. An orange 'Sign Up' button is centered below the fields. At the bottom, there are two buttons: 'Sign Up with Facebook' (blue) and 'Sign Up with Google' (red).

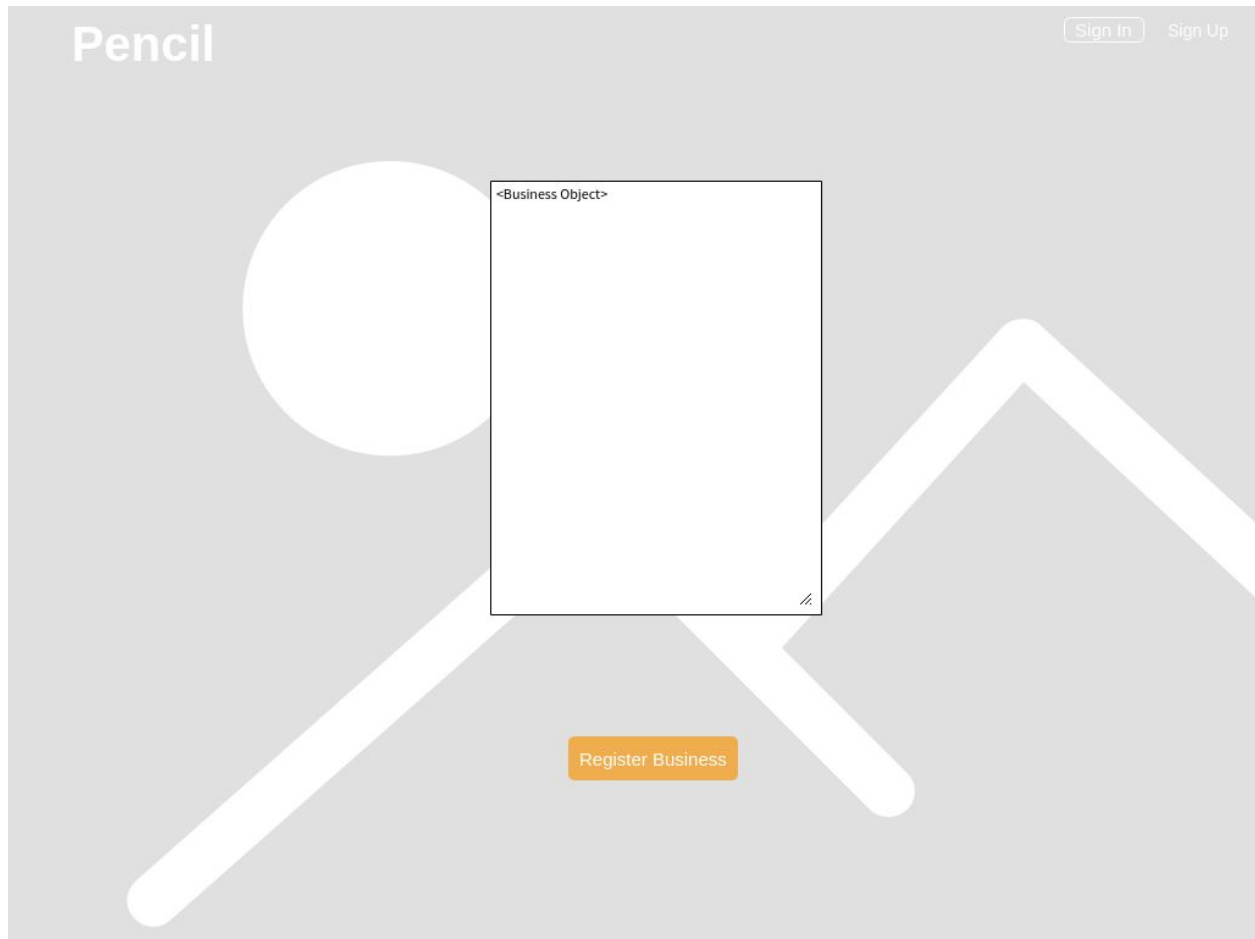
Pencil [Sign In](#) [Sign Up](#)

☒ Business ☐ Personal

[Sign Up](#)

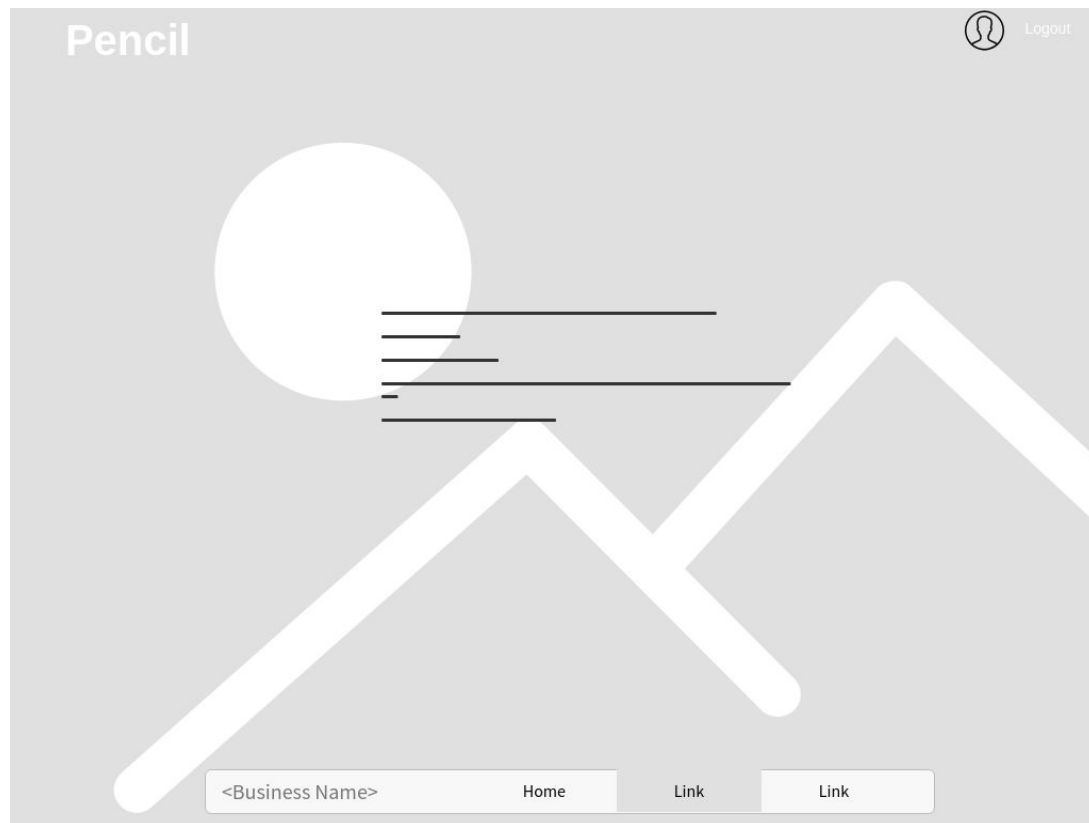
[Sign Up with Facebook](#) [Sign Up with Google](#)

Business sign up




Pencil

Owner view/owner tools



Pencil


 Logout

<

January 2020

>

Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				



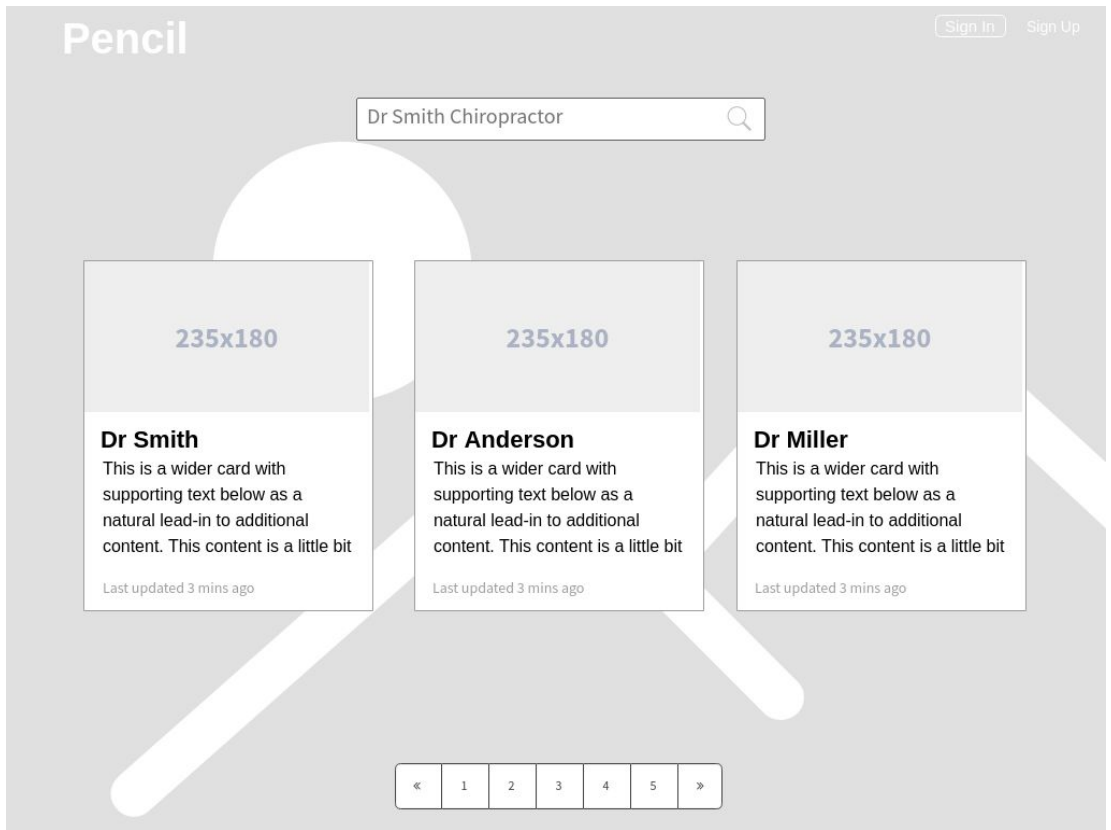
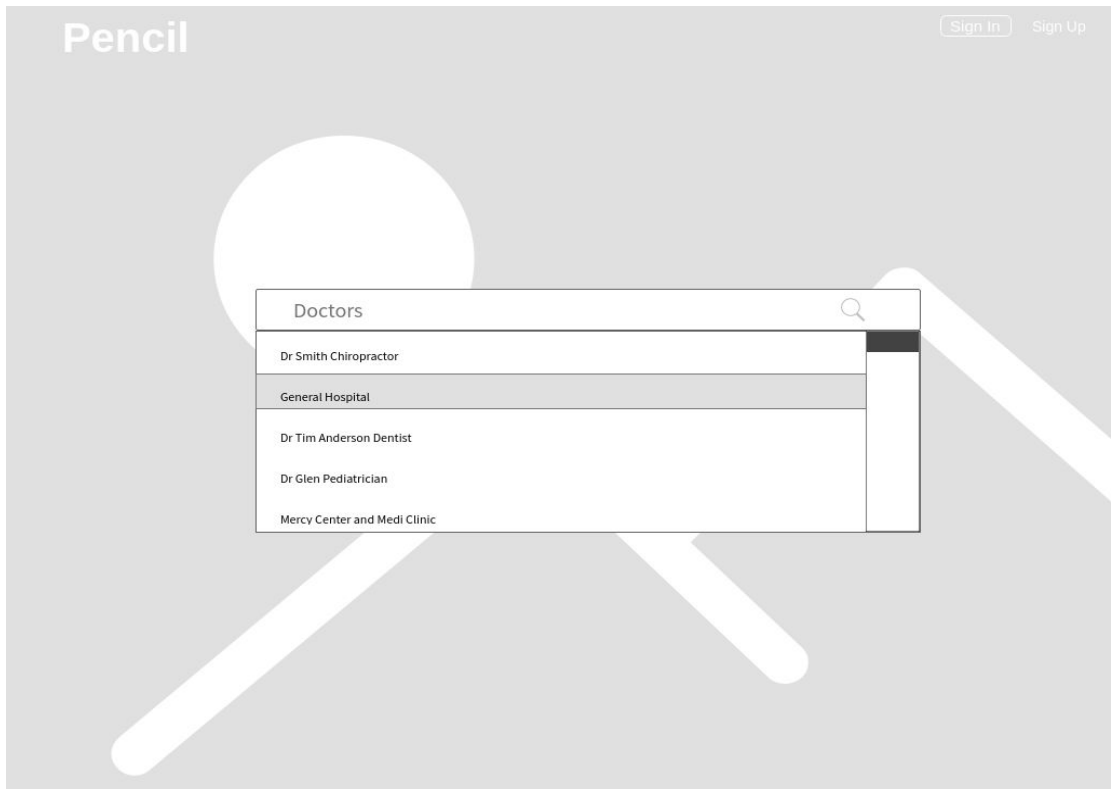
<Business Name>

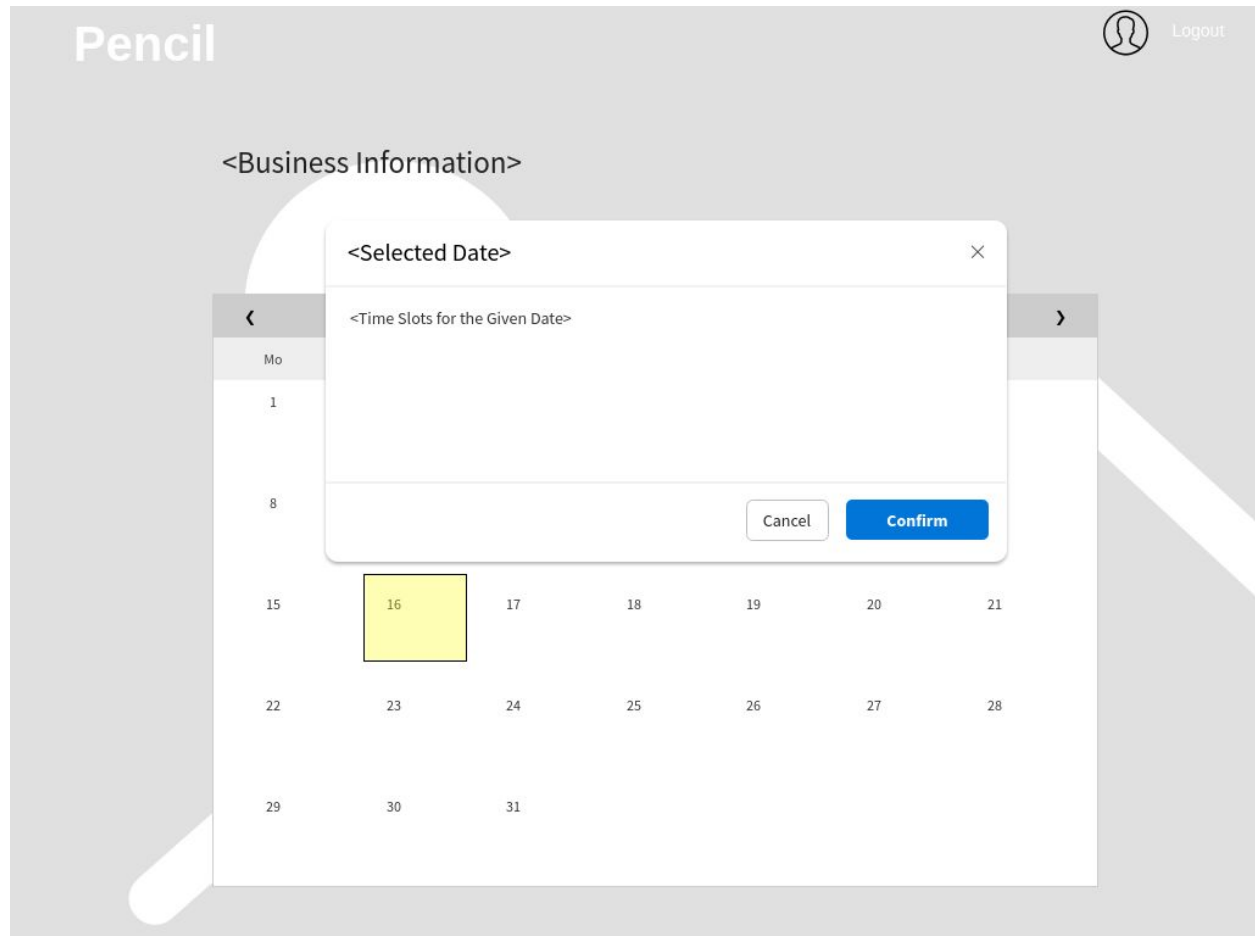
Home

Link

Link

Search





Section 8 - Business Decisions

8.1 Monetization

Due to the fact that Pencil has a small user base the chosen business model was to keep it as a “Freemium” app. A freemium application is an application that is free for all users but has premium features that they are able to purchase. The base functionality of Pencil is available to the free user but extra features could be purchased on a subscription basis. A full list of features that would be considered premium features is not yet complete but for example improved analytics can be a purchasable functionality.

8.2 Statistical Analysis

The analytics available to all business owners are simple, easy to track values. Free plans would include analytics like the number of visitors to your business page and the number of successful appointments made. This would be useful for any business using Pencil as it gives an estimate of how customers require their needs, and if it's successfully being advertised to the correct audience. More advanced analytics could be specifics about the customers that visit your business. It would allow for more personalization with the customers. Additionally, average time spent on the business page would be an advantageous premium feature. The average time spent could be an indicator that user time needs to be sped up, streamlining their business model, or that not enough is being offered to customers. Pencil's statistical analysis could be a very advantageous feature that Pencil would be able to offer over its competitors.

Section 9 - Testing Plan

9.1 Code Testing

During the development of Pencil, the Jasmine test framework was implemented in order to perform both unit, and integration testing. By using the command `ng test`, the Karma test runner would be launched, and the test results would be output to the console. A web page would also open automatically, with the Jasmine HTML Reporter displayed on the screen. The tests themselves would be written in each components `.spec.ts` file, allowing for testing of that individual component. One of the ways testing could be implemented would be to test our different Angular services. The purpose of these tests would serve two purposes. The first, to ensure that our services were configured properly, and were actually able to hit our API, and get a result back. The second purpose would verify that our API was working correctly, returning what it was supposed. As a result of these tests, an issue with CORS (Cross-origin resource sharing) was discovered in the early implementation of the API, and the correct actions were executed in order to fix this error.


```
describe('HomeService', () => {
  const service: HomeService;
  beforeEach(() => { service = new HomeService(); });

  it('#getBusiness should return the name Ragged Ass Barbers',
    (done: DoneFn) => {
      service.getBusiness('Ragged Ass Barbers').subscribe(value => {
        expect(value.name).toBe('Ragged Ass Barbers');
        done();
      });
    });
});
```

Code snippet of a possible Jasmine test

9.2 User Testing

From the beginning of Pencil's development the goal was to create a seamless and enjoyable to use web application. Due to the nature of Angular, Pencil is a single page application. The single web page gives it a very snappy and responsive feel when visiting the different subpages within the webapp. With Paul Hewitt's experience from working in industry and Ian Quach's experience from course work with ENSE496AB the Pencil team has a very strong foundation for creating good user experiences. The initial lo-fi mockups designed online were tested with Paul's parents, Ian's girlfriend and her family. Having a wide range of ages allowed for testing how different age groups navigate through the webapp. Feedback from the test groups were then used for the initial website design. Due to SARS-CoV-2 other test groups that were planned had to be cancelled. The results from the user tests provided vital information to help create a sleek website experience. Users are able to navigate through with few headaches easily providing the information the customers and business owners need.

Section 10 – References

Andy Meeting - 1