BookIt

Final Technical Report - Group Nine

ECE492: Senior Design

Team Members: Fernando Mendez (fm372@drexel.edu), Cheyenne Dwyer (cgd35@drexel.edu), Jason Houghton (jgh48@drexel.edu)

Team Advisor: Dr. Mark Boady (mwb33@drexel.edu)

Abstract

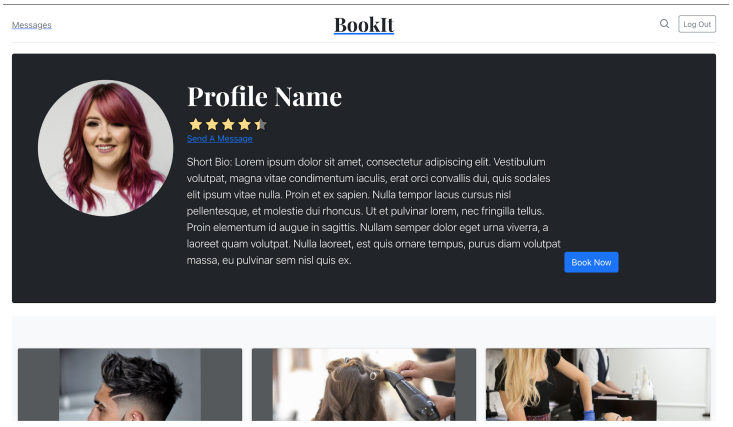
With the rise of the internet and mobile technology came waves of innovation that changed the way people do things. Today, people order taxis with Uber, food through GrubHub, and even crafts from Etsy. However, there still exists no such technological equivalence for scheduling appointments. This was especially highlighted by the COVID-19 pandemic where professionals like barbers, hairdressers, and tattoo artists saw a decline in their revenue as they were forced to operate on an appointment-only basis.

To better understand why an effective solution has not yet arisen, a market survey of existing scheduling softwares and service provider marketplaces was conducted. From this research, BookIt was created to be the most fitting marketplace solution for modern-day service providers. BookIt removes the barrier to entry that is imposed by existing platforms by offering a market-unique pricing model in the form of a monthly fee that is capped and proportional to the amount of appointments scheduled by the service provider. Further, BookIt more appropriately models the relationship that service providers have with their place of business and it offers free schedule-management software with which service providers can optimize their schedule. Service providers can also be able to list their schedules on its marketplace for potential clients to view and insert themselves into an available time slot. These innovations that BookIt provides will transform how people schedule appointments.

1. Executive Summary

Objective

- Develop a platform that is easy for both service providers and clients to schedule appointments
- Users are able to view reviews, different services in their area, and message service providers all from one platform
- Add ability for users to contact service providers, or vise versa, via a messaging feature to allow easy communication



Approach

- Develop a login page with the ability for the user to log into an existing account, register for a new account, or reset the password of an already existing account
- Create a messaging feature where users and service providers can communicate with one another, and reference old messages
- Add a profile page for business to display information like location, services offered, photos, and user reviews.
- Allow user to search for service providers and book appointments

Features

- Login page
  - Password Reset
  - Create Account
- Profile Pages
- Messaging
- Search for service providers
- Book appointments
- Schedule Management

2. Introduction

2.1 Motivation

A significant portion of the service and personal care industry has been lagging behind other markets when it comes to modernization and innovation, especially in relation to the rise of social media and mobile tech. Currently, there exists no technological equivalence to platforms like Uber and OpenTable with regards to the service/personal care industry; instead, niche technologies that are suitable for limited audiences exist. Such a platform would greatly benefit barbers, hair stylists, nail technicians, beauticians, photographers, physical trainers, and whoever else that provides services on an appointment basis, and thus may be referred to as *service providers (SPs)*. The places in which SPs provide services can be referred to as *shops*, e.g. hair salons for hair stylists. Further, those in need of a service may be referred to as *clients*. The issue to address here is multifaceted: how can clients more easily connect with local SPs, and how can SPs increase their clientele and consequently their weekly earnings while better managing their business?

Web technologies like social media and personal websites exist to help address this issue. Although platforms such as Instagram and Facebook have become key for SPs to generate more business, SPs have to share a platform with other content creators, thus making social media a non-optimal solution. Personal websites are also non-optimal as they are costly and are more suitable for “made” SPs or shops as a whole. Since not all shops (and most definitely not all individual SPs) will have websites, search engines are not suitable for connecting with SPs.

Unsurprisingly, some start-ups have already created alternative solutions in the form of calendar management software coupled with an online marketplace, akin to eBay or Etsy, where potential clients can find and schedule services. However, these solutions are not suitable for many SPs. Their pricing models are too inhibitive or pricey, or they misrepresent the shop-SP relationship, or both. Thus, after having completed a thorough analysis of existing solutions, Group Nine had developed a market-unique

pricing model coupled with a marketplace and schedule management solution to meet the needs of modern SPs.

2.2 Problem Statement

To help SPs grow their business, the number of appointments they schedule must be increased. SPs typically do this by showcasing their skills and presenting their availability to their followers on social media. Because their posts may not reach many people outside their friend groups, SPs need an elevated platform so that they may be more visible to their local population.

Schedule management is a major inhibitor to the growth of SPs' business. How do SPs manage their schedules today? While some solutions include the use of costly in-house software or a receptionist, most SPs manage their own schedules by hand, which is time-consuming and unproductive. For instance, clients and SPs must undergo an often lengthy exchange just to confirm an appointment. SPs need to avoid the frustrations that come with schedule management so that they can focus on their business.

2.3 Stakeholders Needs

The target market for BookIt can be split into two categories: SPs and clients. BookIt is addressing two fundamental problems faced by SPs: lack of visibility to potential clients where SPs struggle to find ways of attracting new clients outside of social media (apart from walk-ins), and schedule management, where schedule management is tedious and prevents them from promoting themselves.

The other group of stakeholders are clients, which are the users who will be driving demand on this marketplace. They are made up of anybody who is in search of a service. Clients need an easier way to discover and schedule SPs.

2.4 State of the Art

The ideal solution addresses both of these issues in a single mobile app that is intuitive to use. BookIt provides a solution in the form of a marketplace that is essentially free-to-enter while also providing free-to-use schedule management software.

Current marketplaces charge a high, flat monthly fee that is costly for SPs. In contrast, BookIt will only charge a fee at the end of each month that is proportional to the amount of appointments SPs make through the marketplace. BookIt will also impose a monthly cap on this fee to be fair to high-performing SPs. This pricing model is unique to the market and is more fair to SPs than existing models.

Another way in which BookIt's marketplace differs from existing solutions will be in the way in which shop-SP relationships are modeled. Current platforms like Booksy, theCut, and Fresha [4] do not allow SPs to create separate profiles apart from their shops' profile. This means that SPs do not own their own reviews or photos that may be posted to the shop's profile. What BookIt will do is appropriately model this relationship by allowing SPs to link their profiles to a shop, thus allowing SPs to retain their data if they switch shops. In essence, BookIt allows SPs to create and maintain *portfolios*.

2.5 Approach

BookIt was created as a single-page mobile web app (SPA) with extensive use of Javascript. One clear cut advantage that SPAs have over traditional websites is the entire application is fetched from the server at once. Any subsequent calls to the server are for fetching user data. This generally leads to quicker render times as the client browser does not have to wait to fetch and re-render entire HTML pages. Instead, SPAs only re-render HTML components that change, which leads to a more pleasant user experience than traditional websites.

2.5.1 Client Features

Clients can discover SPs via the search feature. They can create a query for any type of SP; their query results are sorted by distance. Clients can also navigate to SPs' pages from which they can view the SPs' work, send them messages, and request appointments. To request an appointment, clients can select from available times in an SP's schedule. Their request will appear as pending in their schedule submission.

2.5.2 Service Provider Features

SPs can create a page and list their offered services and prices, reviews, and location in adherence to BookIt's default page template. SPs can also message their clients, view pending or cancelled appointments, view their past appointments, set new schedule items, and view and edit their upcoming appointments.

3. Software and Methods

3.1 NodeJs

NodeJs and its package handler, NPM, was used extensively to develop the SPA and the backend server. Many open-source libraries and packages are made available through NPM. For instance, the popular Axios and Sequelize Node libraries are used by BookIt to make HTTP requests and perform database queries, respectively.

3.2 The Graphical User Interface (GUI)

The following were used to develop BookIt: HyperText Markup Language (HTML), used to structure how the SPA's appearance; Cascading Style Sheets (CSS), to style the SPA; and Javascript, use to dynamically change the appearance of the SPA and to fetch user data. Frameworks and libraries like ReactJs, Bootstrap, and Axios were used to facilitate development.

3.2.1 ReactJs

The UI Library used for this project is ReactJS, a popular framework created by Facebook for developing SPAs. As React is backed by a large community, many issues were easily troubleshot. Further, React's extensive UI library allowed for cohesive design, which reduced time spent coding common objects like widgets and forms. React also enables BookIt to be highly interactive via *hooks*, *state variables,* and *context variables*. These concepts are used to dictate when an HTML component should re-render, e.g. when new data is fetched or a notification is received.

3.2.2 Bootstrap

Bootstrap is a popular, open-source CSS framework developed by Twitter and used by many web developers. Rather than defining BookIt's CSS from scratch, Bootstrap was used to style the appearance of BookIt. Bootstrap offers many stylings for components like forms and buttons. There is even a NodeJs Bootstrap library that works nicely with React, which further facilitated development.

3.2.3 Axios

Axios is an alternative to the Javascript native function for making requests which is Fetch. BookIt uses Axios to make requests to the back end server whenever it needs to fetch or modify data in the database. Compared to Fetch, the syntax and functionality of Axios allows for easier handling of server responses.

3.3 The Server

BookIt's server satisfies the data needs of the front end SPA, which includes but is not limited to:appointments, photos, messages, and profile information. Thus, the server must be capable of processing HTTP requests and executing database queries.

3.3.1 ExpressJs

BookIt uses ExpressJs, a popular open-source routing library that is available via NPM. ExpressJs allows BookIt to parse HTTP requests to perform a set of actions on database tables.

3.3.2 Sequelize

Database queries are executed with SequelizeJs, a Javascript library used to interface with a database. Compatible with databases like MySQL and Postgres, SequelizeJs abstracts database queries to Javascript functions. With these abstractions, raw SQL did not have to be written which simplified the codebase. Sequelize provides input validation which provides for data integrity.

3.3.3 Bcrypt

Each user must provide a password upon registration so that they can login to their account. To maintain security, BookIt must encrypt passwords before storing them in the database. To encrypt passwords, BookIt uses the Javascript implementation of the Bcrypt algorithm which was designed specifically for password encryption. The Bcrypt algorithm allows passwords to be “salted,” or have a random string hashed with the password to ensure that two identical passwords will not result in the same hash. Thus, even if attackers gain access to the database, they will not be able to recover every user’s passwords.

3.3.4 JSON Web Token (JWT)

JSON Web Tokens (JWTs), were used to support the “reset password” feature. A JWT is a JSON that is cryptographically signed. The JWT becomes invalid if its contents are modified, prompting the JWT issuer to reject the JWT. BookIt uses JWTs to reset passwords by first embedding a JWT in a URL that is sent to the end user in an email. This URL will take the user to a reset password form that submits the JWT along with the new password. If the JWT is valid, BookIt will change the password.

4 Results

4.1 Specifications, Constraints, and Standard

As BookIt stores user data, it is essential to hash each user's password to foil unauthorized database accesses. Further, a scheme was devised so that users won't have to continuously provide their username and password to log in to BookIt. Lastly, a mechanism was created to prevent access to secure content by unauthenticated users.

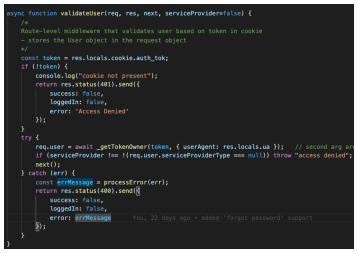
4.2 Concepts

The password each user provides to BookIt upon registration is hashed using the Bcrypt algorithm before it is stored in the database. On each subsequent login attempt, the provided password is checked against the hashed version in the database. HttpOnly authentication tokens, which are created on a successful login, are employed to keep users logged in. These tokens are useful as they can only be read by the server, which helps mitigate some attack vectors. Each issued auth token is stored in a database table which allows the server to retrieve users associated with valid tokens (Fig. 1c).

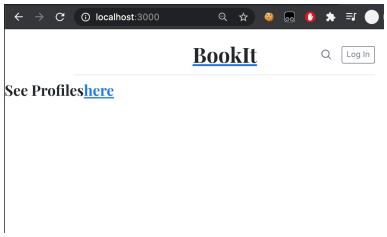
BookIt also defines private routes which are SPA pages accessible by logged in users only, like the messenger and calendar page. Private routes were implemented in ReactJs via context variables, which are global variables accessible by subcomponents of the application. Server authentication middleware in ExpressJs is also employed to prevent access to unauthorized data (Fig. 1).

(a)

(b)



(c)



(d)

Figure 1: (a) Setting up private routes with context variable (b) using context variable to display or hide “Messages” link (c) the server middleware used for verifying a user’s credential (d) “Messages” link hidden from unauthorized user

Since BookIt is a SPA, the server does not need to concern itself with rendering HTML. Rather, the server is focused primarily on performing database queries based on HTTP requests. API endpoints had been created to process such requests (Fig 2).

(a)

(b)

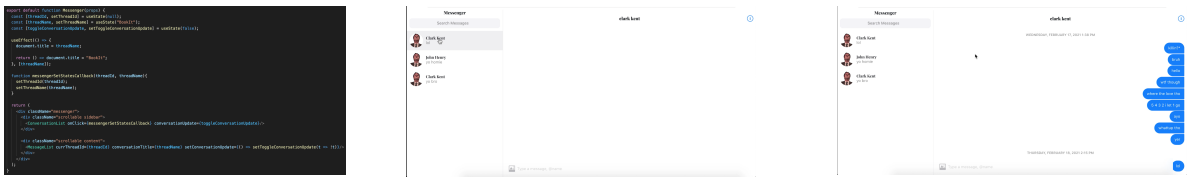
Figure 2: (a) The endpoint function which processes a “send message” request (b) the set of messaging endpoints where the validateUser middleware verifies users before executing any queries

4.3 Detailed Design

4.3.1 SPA Model and Results

Each page was developed as a React component, which themselves could be made up of smaller components. An example would be the Messenger page component, which is composed of the Conversation List and Message List components (Fig. 3a). Each component usually has state variables associated with it which trigger re-renders of components via the useState hooks. Further, useEffect hooks are used to trigger processing post re-renders. For instance, when a user clicks on a Conversation component from the Conversation List, it calls the useState hook to change the threadID and threadName state variables. The Message List component then uses the

threadID and threadName states in useEffect hooks to fetch the conversation from the back end and re-render it (Fig. 3b-c).

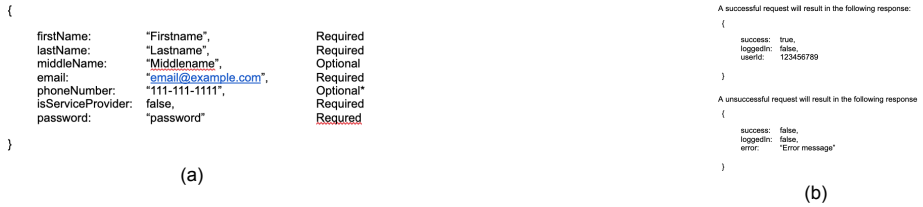


(a) (b) (c)

Figure 3: (a): The Messenger component is made up of the Conversation List and Message List Components
(b) the Messenger component before the Message List useEffect is called (c) the Messenger component after the useEffect is called

4.3.2 Server Model and Results

Most API endpoints will be responsible for a database query. Each endpoint requires a specific input format. To facilitate development, a specification document describing the input and output API data formats was written (Fig. 5).

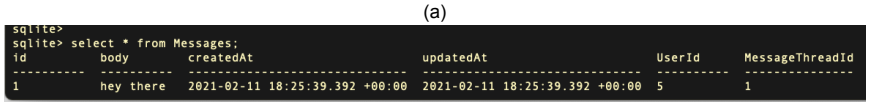


(a) (b)

Figure 5: The specification for the register user API endpoint (a) the data format required by the endpoint (b) the data format of the endpoint response

Each table in the database is modeled as an extension of the Sequelize Model class (Fig. 6).

(a)



(b)

Figure 6: (a) definition of the Message model and relationships (b) the resultant table, where the UserId and MessageThreadId columns are foreign keys from the User table and Message Thread table.

4.3.3 Final Results

Using the concepts and techniques described in the previous two sections, the following features were developed: profile creation and management, schedule management, and service provider querying and scheduling (Fig. 7).

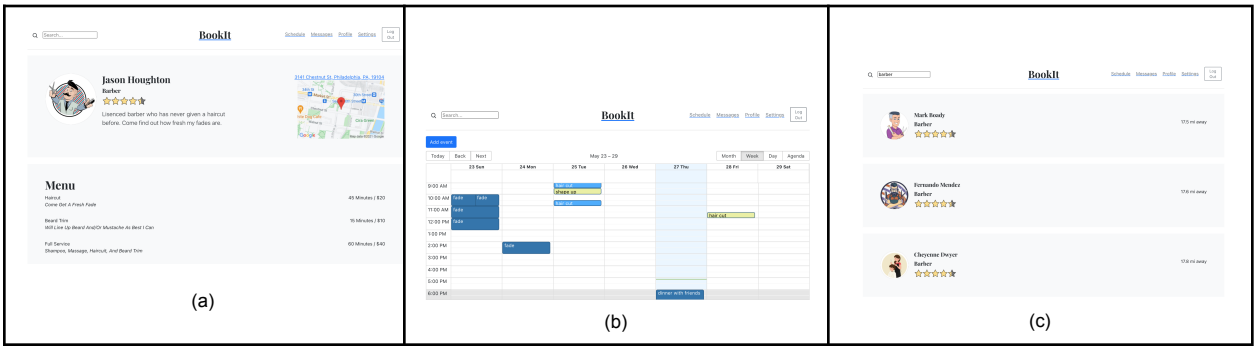


Figure 7: (a) Example service provider profile (b) the calendar page which shows events that are color coded (c) search results for service provider

5. Discussion

One of the biggest lessons learned is that a lot of time can be expended in thinking about how to implement a feature if one dives straight into the code without any meaningful forethought. For example, the design of the messaging database tables would have been developed much quicker had sequence diagrams been used first. Having learned this early on, developing sequence diagrams like the one shown in Fig. 8 for registration has become integral when developing new features.

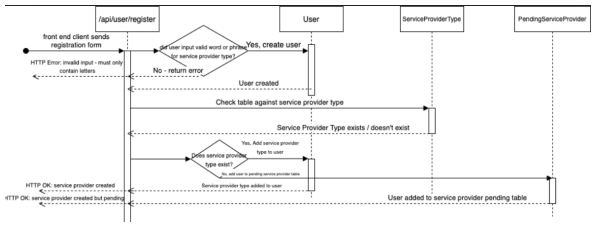


Figure 8: Sequence diagram for registering a new service provider

Further, the API specification document had proved valuable when developing new frontend features. With API endpoints properly documented, less time was wasted waiting for a developer to respond to questions regarding the backend.

Lastly, modularity was important in reducing redundant code and improving code readability. Writing long pieces of code resulted in there being more bugs, and it

inevitably became more difficult to maintain. By taking the time to abstract out common functionality and appropriately delegate responsibility among code components, code maintenance was greatly improved and bugs became easier to find.

6. Budget

As this project was implemented with open source software and is not yet at a state where it is ready for production, there are no tangible costs associated with development.

As this project begins to roll out, there will be some new costs to think about. These costs would include things like credit card transaction fees, improving our servers for larger scale hosting, and advertisements.

6.1 Projecting Future Costs

6.1.1 Transactions

Paypal allows for free transaction support for small to medium businesses, and includes a small transaction fee on each purchase. Scaling up, for a high volume of transactions, Paypal offers complete support for $30 a month.

6.1.2 Data Storage

To move our data to the cloud, it would currently fall within the free-tier of Amazon Web Services. However, when we would scale up to say, 100GB per month of data using Amazon Aurora database for onDemand data, it would be $70.91 per month.

6.1.3 Advertising

To advertise with Google Ads, the cost-per-click changes depending on many different factors including the type of website or business, however, the average

cost-per-click for an ecommerce website is about $1 per click. Assuming a large scale of 100 clicks per day, we would have to pay around $3,000 per month.

6.1.4 Overall

The pricing as the website and business expand can vary widely depending on size and how much data and advertising it needs. On the whole, advertising costs the most, as there are many low-cost options for storing large amounts of data these days.

6.1.5 Future Cost Estimate

Transactions	30.00 + transaction fees
Data Storage	70.91
Advertising	3000.00
Total	3100.91 / per month

7. Project Management

7.1. Team Organization

The team organization remains largely the same:

Team Member	Responsibility
Fernando	Chief developer for service provider front end Assist development of server side code
Jason	Chief developer for client front end Assist development of server side code
Cheyenne	Chief developer of server side code Assist development of front end

Sources

[1] DataUSA. (2020). *Barbers, Hair Dressers, Hair Stylists, Cosmetologists, and Make-Up Artists Median Ages*. Retrieved from https://datausa.io/

[2] U.S. Bureau of Labor Statistics. (2017). *Barbers, Hair Dressers, Hair Stylists, Cosmetologists, and Make-Up Artists Employment Stats*. Retrieved from https://datausa.io/

[3] U.S. Census. (2019). *U.S. ACS Demographic and Housing Estimates*. Retrieved from https://data.census.gov/