The Detailers Edge

Jonathan Ramos
Ayoposi Olu-Bamisaye

January 2025

1 Introduction

The client, Emerson Jordan/The Detailer's Edge, is a self-employed business owner, seeking to improve his business's online presence and streamline his appointment scheduling process. His current online presence is an Instagram account where he handles all client interactions, including appointment scheduling. His current scheduling process involves direct messaging and phone calls, both of which can quickly become time consuming, limiting his ability to expand his business. To address his current challenges, our client has proposed a dedicated website to provide his clients an easy to understand and efficient way to schedule appointments. Automating the scheduling process through a user-friendly website will help Emerson save time, improve overall client satisfaction, and attract new clients through its professional image. Additionally, the website will serve as a portfolio for Emerson in which he can showcase the quality of his work. The website will also improve the company's credibility and trustworthiness by showing customer testimonials and reviews. Finally, the website needs to be mobile-friendly catering to his largely mobile- based clientele. Ultimately, this project will help Emerson/The Detailer's Edge focus on solely delivering high quality work, while positioning his business for success in a competitive digital market.

2 Functional Requirements User Stories and Tasks

2.1 Client Stories

S1.1: View Available Appointments (5 points)
As a client, I want to view available appointment slots in a calendar format so that I can see all scheduling options.

S1.2: Select Appointment (2 points)
As a client, I want to select an available appointment slot so that I can initiate the booking process.

S1.3: Create Appointment (3 points)
As a client, I want to create a new appointment by providing my details and service preferences so that I can secure my desired time slot.

S2: Booking Confirmation (2 points)
As a client, I want to receive confirmations so that I am assured my appointment has been set.

S3: Appointment Reminders (3 points)
As a client, I want to receive reminders so that I don't forget my appointments.

S4: Cancel Appointment (2 points)
As a client, I want to cancel my appointment so that I can accommodate schedule changes.

S5: Reschedule Appointment (3 points)
As a client, I want to reschedule my appointment so that I can accommodate schedule changes.

S6: Browse Services (3 points)
As a client, I want to browse detailed descriptions and pricing for services so that I can select the service that best fits my needs.

S7: Browse Portfolio (5 points)
As a client, I want to browse previous work through images/videos so that I can assess the quality of services before booking.

S8: Contact Form (1 point)
As a client, I want a contact form so that I can ask questions or request more information.

S9: Dynamic Testimonials (5 points)
As a client, I want to read testimonials and reviews from other clients so that I can trust the quality and professionalism of the business.
Definition of Done: Testimonials are stored in database and can be dynamically loaded and displayed.

S9.1: Submit Testimonial (3 points)
As a client, I want to submit my testimonial after receiving service so that I can share my experience.

S10: Instagram Feed Integration (3 points)
As a client, I want to view the business's Instagram feed within the website so that I can stay updated on their latest work.

S11: Client Registration (5 points)
As a client, I want to create an account so that I can manage my appointments and information.

S12: Client Login (3 points)
As a client, I want to login to my account so that I can access my appointment history and details.

2.2 Admin Stories

S13: Admin Account (3 points)
As an admin, I want secure admin login credentials so that only authorized users can make content changes on the website.

S14: Manage Portfolio CRUD (8 points)
As an admin, I want to add, edit, or delete portfolio images easily so that I can keep my portfolio up to date.

S15: Manage Services CRUD (8 points)
As an admin, I want to create, read, update, and delete service offerings so that I can keep the services available up to date.

S16: Modify Timeslots (5 points)
As an admin, I want to define which times are available for appointments so that I can maintain control over the scheduling process.

S17: View Scheduled Appointments (3 points)
As an admin, I want to view a list of all scheduled appointments for the day so that I can easily keep track of what appointments remain.

3 Non Functional Requirements

NFR Usability

- The interface shall be mobile-responsive.
- The website shall be accessible on all major browsers (Chrome, Firefox, Safari).

NFR Security

- Admin access must have secure authentication.

NFR Reliability

- The system shall have stable uptime.

NFR UI

- The system will maintain a constant monochrome color scheme.

Iteration	User Stories	Points
1	S1.1, S1.2, S1.3, S2, S13	15
2	S3, S4, S5, S11, S12	16
3	S6, S7, S8, S17	12
4	S9, S9.1, S10, S16, S15	24
5	S14	13
	Total:	**80**

Table 1: Story Planning with Point Distribution

4 Iteration Planning

5 Architecture

5.1 Cloud-Based Architecture Overview

The system implements a comprehensive cloud-based architecture utilizing Firebase as its primary platform. This serverless architecture is designed to provide robust scalability, seamless real-time updates, and enterprise-level security features. At its core, Firebase Authentication serves as the primary security layer, managing user authentication workflows for both clients and administrators. This component ensures secure login processes, handles session management, and maintains distinct access levels between client and administrative users.

The data layer is built upon Cloud Firestore, This database architecture efficiently manages several critical data collections: user profiles and credentials are stored with encrypted sensitive information; appointment data and scheduling information are structured to enable quick queries and real-time updates; service descriptions and pricing are organized to facilitate easy updates and version control; testimonials and reviews are stored with user references and timestamps; and portfolio images metadata is maintained with optimized indexing for quick retrieval.

For binary data management, Firebase Storage provides a scalable and secure solution. The storage architecture is organized into distinct buckets: portfolio images and videos are stored with optimized compression and caching strategies; user profile pictures are managed with appropriate size limitations and format standardization; and service-related media is stored with metadata linkage to corresponding service descriptions. This structured approach ensures efficient data retrieval and optimal performance.

Firebase Cloud Functions form the backbone of our operations. These functions handle several automated processes: appointment confirmation emails are triggered immediately upon booking confirmation; reminder notifications are scheduled and dispatched at configured intervals; and Instagram feed integration is managed through periodic API synchronization.

The application's delivery is handled through Firebase Hosting, which pro-

vides content delivery network (CDN) capabilities. This hosting solution ensures fast loading times through edge servers, handles SSL certification automatically, and provides seamless deployment pipelines.

5.2 Simplified Class Diagram

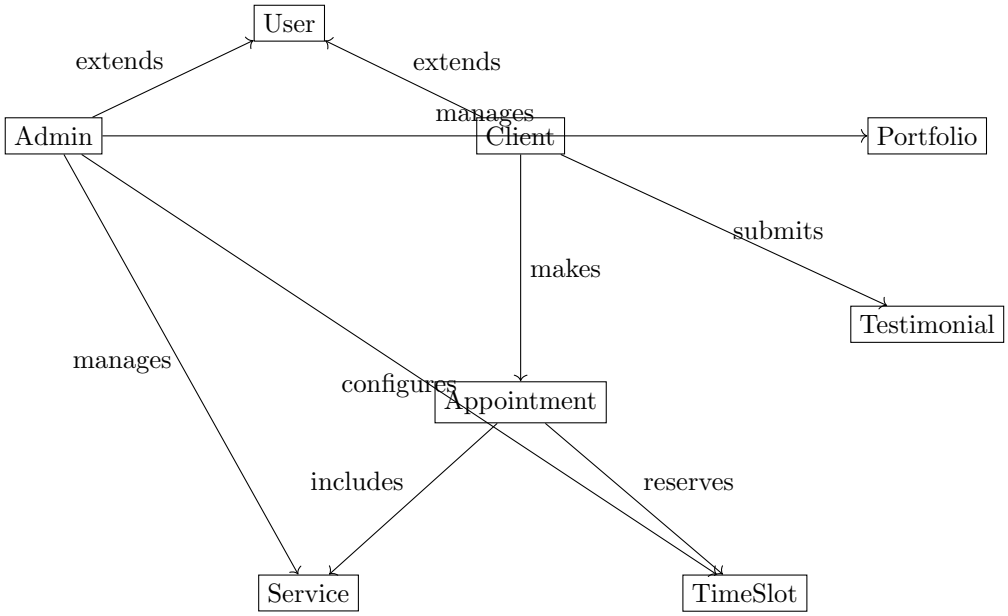


Figure 1: Simplified Class Diagram

The class diagram shows the core entities in the system and their relationships. The User class serves as the base class for both Client and Admin users. Clients can make Appointments which are associated with specific Services and TimeSlots. Clients can also submit Testimonials. Admins manage the Portfolio, Services, and TimeSlots.

6 Technology

Based on the requirements of this project, we will use HTML, CSS, and JavaScript as the foundation of the application. Additionally, we will use React to create an easy to understand user interface for all clients. By leveraging React, we can create a modern user interface that competes adequately with all modern competition. The web application will require a Database to store booking information. The Database will also serve content to the website to allow for dynamic web pages without the need for an admin to code any changes on the website. To serve the needs of our application we have selected Firebase

as it provides modern features such as user authentication while offering the flexibility of a non-relational Database.

For testing we will use Jest as the majority of the code will be written in JavaScript. Due to Firebase being chosen as our Database, we will have to mock many built-in Firebase functions to properly test the functionality of our application. Based on the experience we have had testing Firebase, successfully mocking functions may be a challenge to overcome.

7 Data

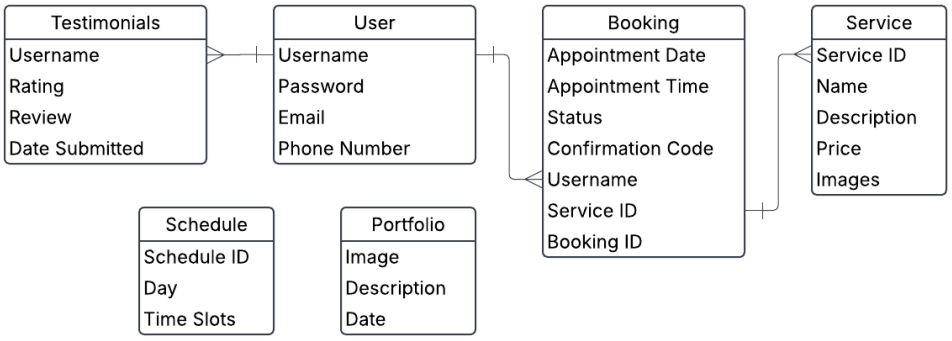


Figure 2: Diagram of our proposed Database

8 UI

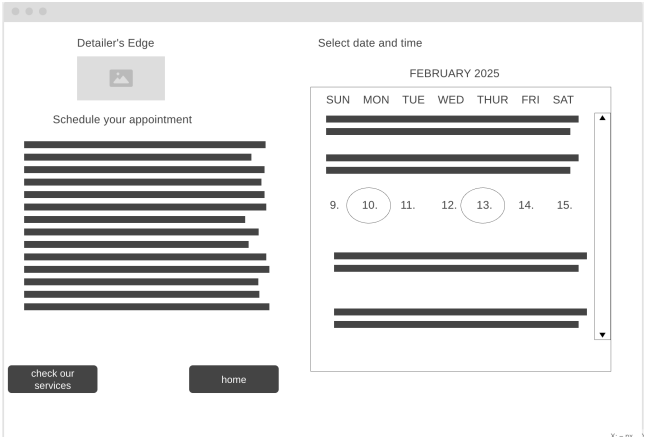


Figure 3: Scheduling Page

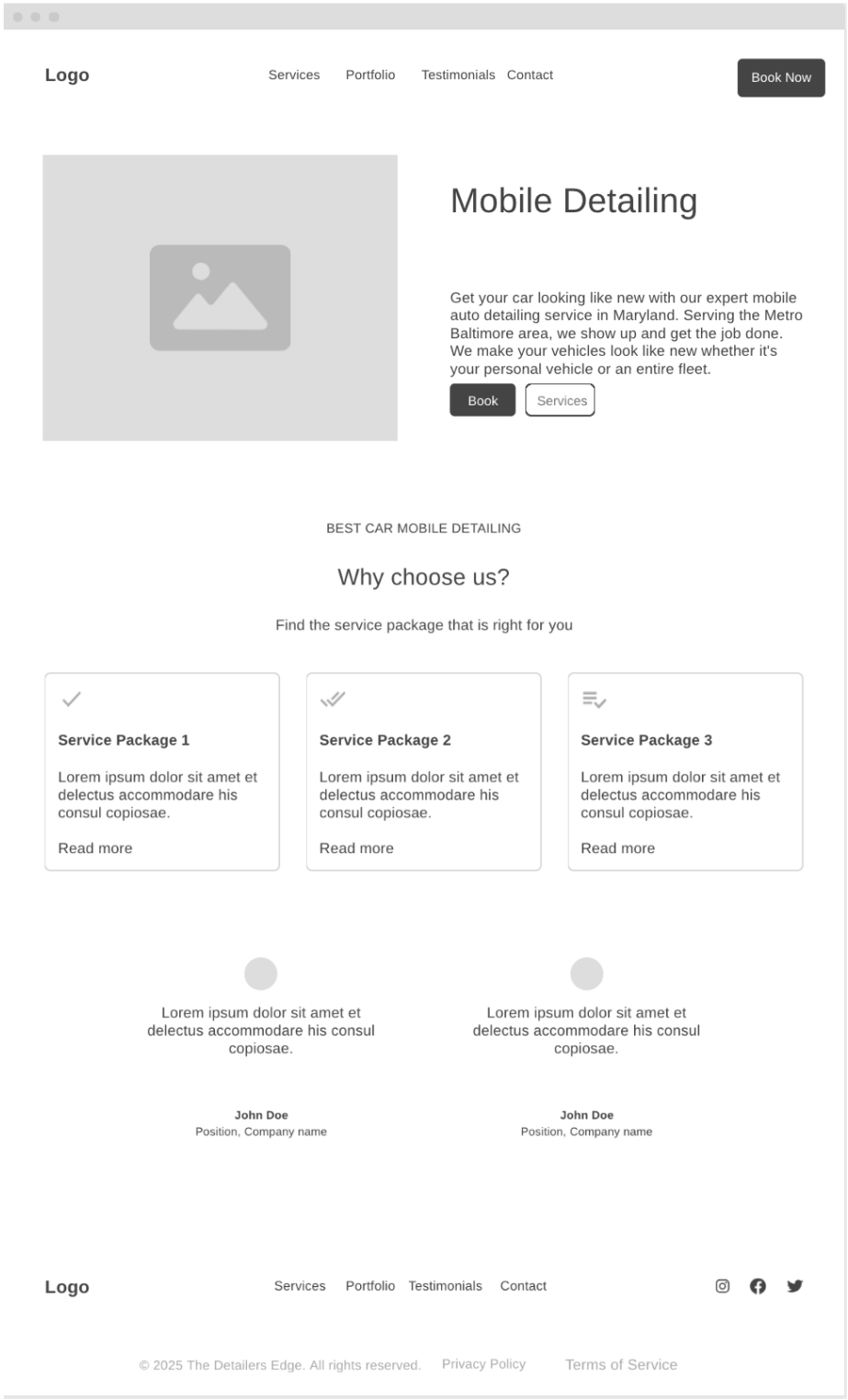


Figure 4: Home Page

9 Development Iteration 1

Name	Stories	Points	Hours
Jonathan	S1.1 S1.2	7	10
Ayoposi	S1.3 S13	6	8

Table 2: Development Iteration 1

9.1 Retrospective and Reflection

During this iteration of development, we focused on setting up our project and database properly to begin the development of the web application. Additionally, the lack of any landing page or home page needed to be addressed as the rest of the application's interface will need to consistently follow the same design and flow overall. During this iteration we also ensured the current interface is mobile friendly as exemplified through the Navbar that is used throughout all of the websites pages. To address the difference in screen size between mobile and desktop, the Navbar transforms into a side menu when the screen size is small.

The setup of firebase and the proposed collections was done during this iteration of development. Our current Firestore collections match exactly the proposed database diagram, with the exception that our user collection does not store passwords due to security concerns. To address this issue, we use Firebase authentication which allows for Google sign-in as well as Email and password sign-up. Passwords are hashed for privacy protection. To handle the appointment bookings offered, we opted to use Google Calendar. We initially underestimated the complexity behind building an entire scheduling system in the span of 1 development iteration. Since we need to leave enough time to build the rest of the application, we looked for other options that we could implement within the web application.

Currently, our testing has over 90% statement coverage for all of the pages that were worked on during this development iteration (booking.jsx, index.jsx, signup.jsx). The pages that show up as 0 coverage, have not yet been tested as they are currently placeholder pages. For testing we are using Jest, which facilitates testing of React code. Jest also allows us to mock firebase functions which is essential for testing authentication and future queries.

9.2 Testing

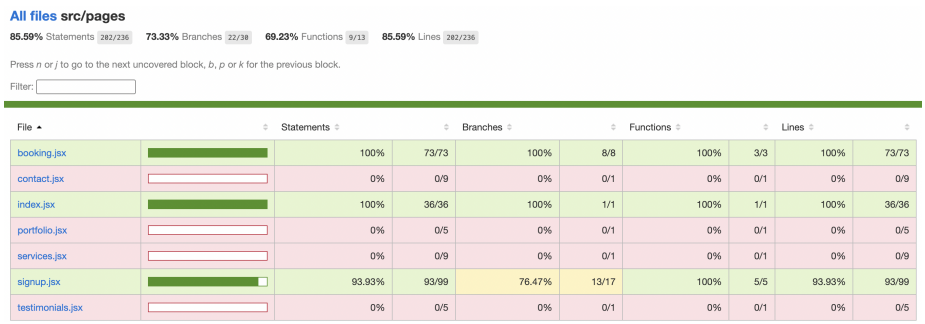


Figure 5: Testing Coverage

9.3 Planning For The Next Iteration

In the next iteration we will continue building out the main page to follow the wireframe design. Additionally, we need to work on connecting Google Calendar with Firebase through the Google Calendar API so that the database will keep track of a user's history of booked appointments. We also need to provide a sign-up for clients and a proper interface where logged-in clients can view their history and other details. Testing needs to be improved to ensure that all current pages at least render properly. We need to ensure that the web application is optimized for mobile devices on all pages so that it loads efficiently while being easy to navigate on a small screen. To ensure that git traceability is improved in the upcoming iteration, we will use development branches while each feature is being worked on.

10 Development Iteration 2

Name	Stories	Points	Hours
Jonathan	S11 S12	8	8
Ayoposi	S3 S4 S5	8	6

Table 3: Development Iteration 2

10.1 Retrospective and Reflection

This iteration of development we integrated firebase authentication into the web application. Users are now able to create new accounts using an email and password, which are saved to the authentication component in firebase. The passwords here are hashed for security, adding an extra layer of protection for

authenticated users. In addition, users can skip the manual registration step by logging in with their Google account. This offers users the flexibility to continue with a trusted account without explicitly providing us with the login credentials to their account. The persistent storage component is also updated when a user is registered, whether through email and password or Google, their basic information such as first name, last name, and email are all stored in the User collection, each user being a single document instance.

Due to limitations uncovered with using Google Calendar and its API, we have decided to use this alternative appointment scheduler called Neetocal which offers more functionality to better suit of project needs.

10.2 Testing

File		Statements		Branches		Functions		Lines	
src		32.25%	20/62	0%	0/2	0%	0/2	32.25%	20/62
src/components		100%	78/78	57.14%	4/7	33.33%	1/3	100%	78/78
src/pages		92.94%	395/425	88.67%	47/53	92.59%	25/27	92.94%	395/425
src/styles		100%	13/13	100%	0/0	100%	0/0	100%	13/13

Figure 6: Overall Testing For Iteration 2

File		Statements		Branches		Functions		Lines	
booking.jsx		100%	72/72	100%	8/8	100%	3/3	100%	72/72
contact.jsx		100%	9/9	100%	1/1	100%	1/1	100%	9/9
index.jsx		100%	36/36	100%	1/1	100%	1/1	100%	36/36
login.jsx		93.57%	102/109	92.3%	12/13	85.71%	6/7	93.57%	102/109
portfolio.jsx		100%	5/5	100%	1/1	100%	1/1	100%	5/5
profile.jsx		90.69%	39/43	77.77%	7/9	100%	3/3	90.69%	39/43
services.jsx		100%	9/9	100%	1/1	100%	1/1	100%	9/9
signup.jsx		86.13%	118/137	83.33%	15/18	88.88%	8/9	86.13%	118/137
testimonials.jsx		100%	5/5	100%	1/1	100%	1/1	100%	5/5

Figure 7: Pages Testing For Iteration 2

During this iteration of development we improved on our testing. Last iteration we only covered testing for booking, home, and sign up pages. Notably, the sign up page had not yet been implemented to support user authentication as of last iteration, therefore only tested static elements. Current testing coverage shows that all pages are now being tested. Some pages have not yet been built and therefore have only been testing to make sure they are being rendered properly into the application. The pages being tested solely for rendering are, Contact, Portfolio, Services, and Testimonials pages, as these will be developed in later iterations. The Log in, Sign up, and Profile pages include more complex

tests that mock firebase components to ensure the functionalities are working properly. In these tests we provide mock users to test the process of creating a new account and logging in. Additionally, we test for signing in with Google to ensure the pop up is working properly, allowing users to input their information.

10.3 Planning For The Next Iteration

For the next iteration we will focus on completing the functionality of the calendar appointment scheduler due to the technology switch made during this iteration of development. We will learn how the appointment scheduler will interact with the interface and provide information for our future story points. The current booking page needs to be worked on to support a more mobile-friendly interface.

Additionally, we will continue to build out the web application to support the functionality of users being able to navigate available services and the portfolio. Notably, these pages are not static, as they should be implemented in such a way that the admin can edit the contents of these pages without having to change any code. This feature improves the usability of our application for the client, as it ensures that the application can stay up to date, without the need for us to return and modify the source code.

11 Development Iteration 3

Name	Stories	Points	Hours
Jonathan	S6 S7	8	12
Ayoposi	S8 S17	4	10

Table 4: Development Iteration 3

11.1 Retrospective and Reflection

During this iteration of development, we built the back-end of the application. Currently, our back-end API provides authentication for user management through both email/password and Google Sign in. The POST request in api/auth allows the creation of a new user and their profile. A GET request is used to obtain that user's information for their profile, and a PUT request is used to update a user's information when necessary. Our API is separate from the front end, being served completely on the back end, it provides improved security as all firebase calls are done from our sever, rather than being called from each client. The change in how we handle user authentication has also allowed user's session to remain active on all pages once they have logged in, even after a refresh. We also implemented NeetoCal API to our project. In doing this we were able to successfully integrate it into the View All Appointment (S17) story. This API allows us to fetch information (Bookings, Confirmations,

etc) from the tool and utilize it inside of our codebase. This API will help us in future sprints concerning the Admin Panel. Contact form was implemented showing a clean interface and once input is received it routes to the backend.

This is a list of the methods currently provided by the authentication API:

- signup(email, password, displayName, phoneNumber)
- login(email, password)
- googleSignIn()
- logout()
- getUserProfile()
- updateUserProfile(data)

11.2 Testing

Testing needs to be worked on during the next iteration of development. The previous tests did not include the use of our back end API, directly mocking firebase functions in the front end. We need to adequately test our backend but in an attempt to make some progress toward the user stories selected for this iteration we did not have enough time to test our changes.

11.3 Planning For The Next Iteration

Due to the time it took to transition the previous features to work with the new API backend, S6 was not completed, and S7 is only partially complete due to an unforeseen obstacle with Firebase. Due to a recent update to the free tier of Firebase, we are no longer able to store images in the storage component. We are using an alternative "Cloudinary" to store these images, which will be used to serve our images in the front end of the portfolio. The admin should be able to change the images that are shown in the front end without having to access Cloudinary. To do this, we will use the Cloudinary API which allows us to upload images directly. In the following development iteration, this API will be used in our backend server to make calls to Cloudinary, allowing an admin to change the images directly from the portfolio page. The same approach will be used to develop the services page, as it also needs to provide dynamic content support for the admin. This is essential as the admin needs to keep their services available as up to date as possible.

The dynamics testimonials page needs to be worked on, as it should allow clients to submit details on their experience with the company's services. Our back end API needs to handle receiving the testimonials submitted by clients, and store them in the database for later reference. As stated previously for security concerns, all communication between our application and firebase must be done through our API as it currently does for authentication.

12 Development Iteration 4

Name	Stories	Points	Hours
Jonathan	S14 S15	16	15
Ayoposi	S9-9.1 S10	11	12

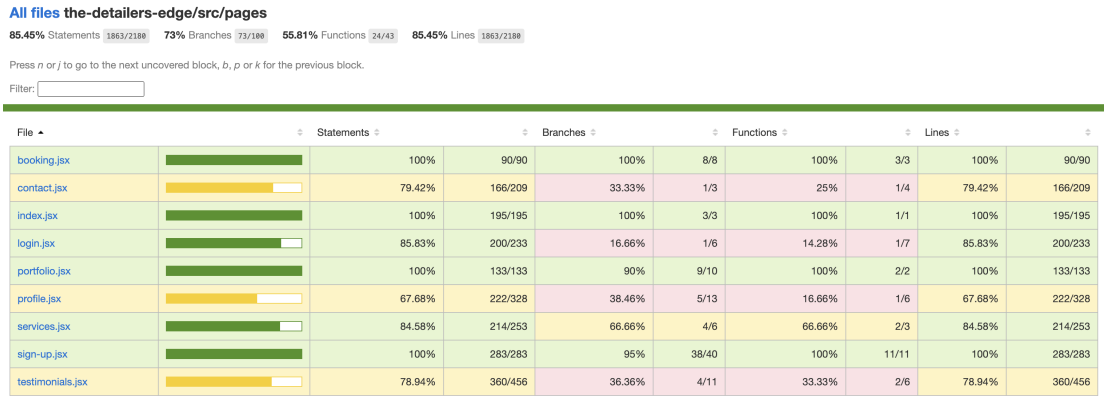
Table 5: Development Iteration 4

During this sprint, we successfully completed Dynamic Testimonials and its sub-story, Submit Testimonial, as well as Instagram Feed Integration. For the Dynamic Testimonials feature, we implemented a complete solution that connects our front-end to the backend API. Testimonials are now stored in our database and can be dynamically loaded and displayed on the website. This gives potential clients the ability to read authentic reviews from other customers. The Submit Testimonial functionality allows clients to share their experiences after receiving services. We built a clean, user-friendly form that collects the testimonial data and sends it to our backend. Once submitted, the testimonial is immediately stored in the database and becomes available for display in the testimonials section. For the Instagram Feed Integration, we successfully incorporated the business's Instagram feed directly into the main home page. This integration allows clients to stay updated and the feed displays recent posts in an attractive, responsive grid layout that matches our overall design aesthetic.

This sprint expanded admin functionality in S14 Manage Portfolio CRUD and S15 Manage Services CRUD. This allows a trusted admin user to manage the content that is displayed for all users directly from the front end. We use the Cloudinary API directly from our backend to communicate with the Cloudinary storage service which then sends the response data to the front end with appropriate error checking before sending a response. Similarly, we communicate with firebase from our backend when updating the available services. This ensures that the front end performs safer operations through our backend. These admin pages are only available to be used by an authenticated admin user, in the event an unauthorized user attempts to accesses these pages they are unable to perform any operations. Additionally, to provide the admin access to this functionality we implemented a basic dashboard that an admin can use to navigate through the application.

12.1 Testing

For testing, we maintained an 85% branch coverage and a 73% statement coverage on our page components. Backend testing was a roadblock this sprint and we will work on fixing it the next sprint.



All files the-detailers-edge/src/pages

85.45% Statements 1863/2180 73% Branches 73/100 55.81% Functions 24/43 85.45% Lines 1863/2180

Press n or j to go to the next uncovered block, b, p or k for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
booking.jsx		100%	90/90	100%	8/8	100%	3/3	100%	90/90
contact.jsx		79.42%	166/209	33.33%	1/3	25%	1/4	79.42%	166/209
index.jsx		100%	195/195	100%	3/3	100%	1/1	100%	195/195
login.jsx		85.83%	200/233	16.66%	1/6	14.28%	1/7	85.83%	200/233
portfolio.jsx		100%	133/133	90%	9/10	100%	2/2	100%	133/133
profile.jsx		67.68%	222/328	38.46%	5/13	16.66%	1/6	67.68%	222/328
services.jsx		84.58%	214/253	66.66%	4/6	66.66%	2/3	84.58%	214/253
sign-up.jsx		100%	283/283	95%	38/40	100%	11/11	100%	283/283
testimonials.jsx		78.94%	360/456	36.36%	4/11	33.33%	2/6	78.94%	360/456

12.2 Planning For The Next Iteration

The next iteration will involve significant improvement on the testing of our application. Additionally, we will expand the admin dashboard to support more admin functionalities such as the ability to modify the available time slots.