UE22CS352B - Object Oriented Analysis & Design

Mini Project Report

Title: Local Service Booking Platform

Submitted by:

Dheeraj C L : PES1UG22CS181
Dhanush M : PES1UG22CS179
C H Nikhil   : PES1UG22CS151
Chandan H S : PES1UG22CS152

6th Semester   Section: C

Mr.Vinay Joshi
Assistant Professor

January - May 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka,
India

Problem Statement:

Finding and booking reliable local services, such as plumbers, electricians, or cleaners, often presents significant challenges for customers. Information is frequently fragmented across various websites or offline directories, making it difficult to compare providers, ascertain real-time availability, understand pricing structures, and gauge service quality based on verified reviews. This typically leads to a time-consuming process involving multiple phone calls and potential scheduling conflicts. Simultaneously, many local service businesses, particularly small to medium-sized enterprises, lack accessible and affordable digital tools to effectively manage their online presence, showcase their offerings, handle appointment scheduling efficiently, and attract new clientele. Relying on manual booking methods (phone calls, emails, paper calendars) can lead to administrative overhead, scheduling errors, and missed business opportunities in an increasingly digital marketplace.

This project addresses this disconnect by developing a centralized web platform that streamlines the process of discovering, booking, and managing local services, benefiting both customers seeking convenience and reliability, and businesses aiming for better visibility and operational efficiency.

Key Features:

The platform provides distinct functionalities tailored for Customers and Business Administrators:

I. Customer Features:

- Advanced Service Search & Filtering: Enables users to search for businesses based on service category, location (city), specific business name, desired date/time availability, price range, and minimum customer rating.
- Detailed Business Profiles: Displays comprehensive information about businesses, including address, contact details, service descriptions, pricing indicators, advance payment requirements, and aggregated customer ratings.
- Real-time Slot Booking: Allows customers to select available time slots based on their search criteria and initiate the booking process directly through the platform.
- Booking Management: Provides registered customers with a dashboard to view their upcoming and past bookings, view specific booking details, and cancel upcoming bookings within the defined cancellation policy window.
- Secure Advance Payment: Integrates a mechanism (simulated or actual) for customers to securely pay the required advance amount to confirm their booking. (Potential for final payment as well).
- Rating & Review System: Allows customers to submit star ratings and text

comments for completed services, contributing to the business's overall rating visible to other users.
- User Account Management: Secure customer registration and login functionality using email and password.

II. Business Administrator Features:

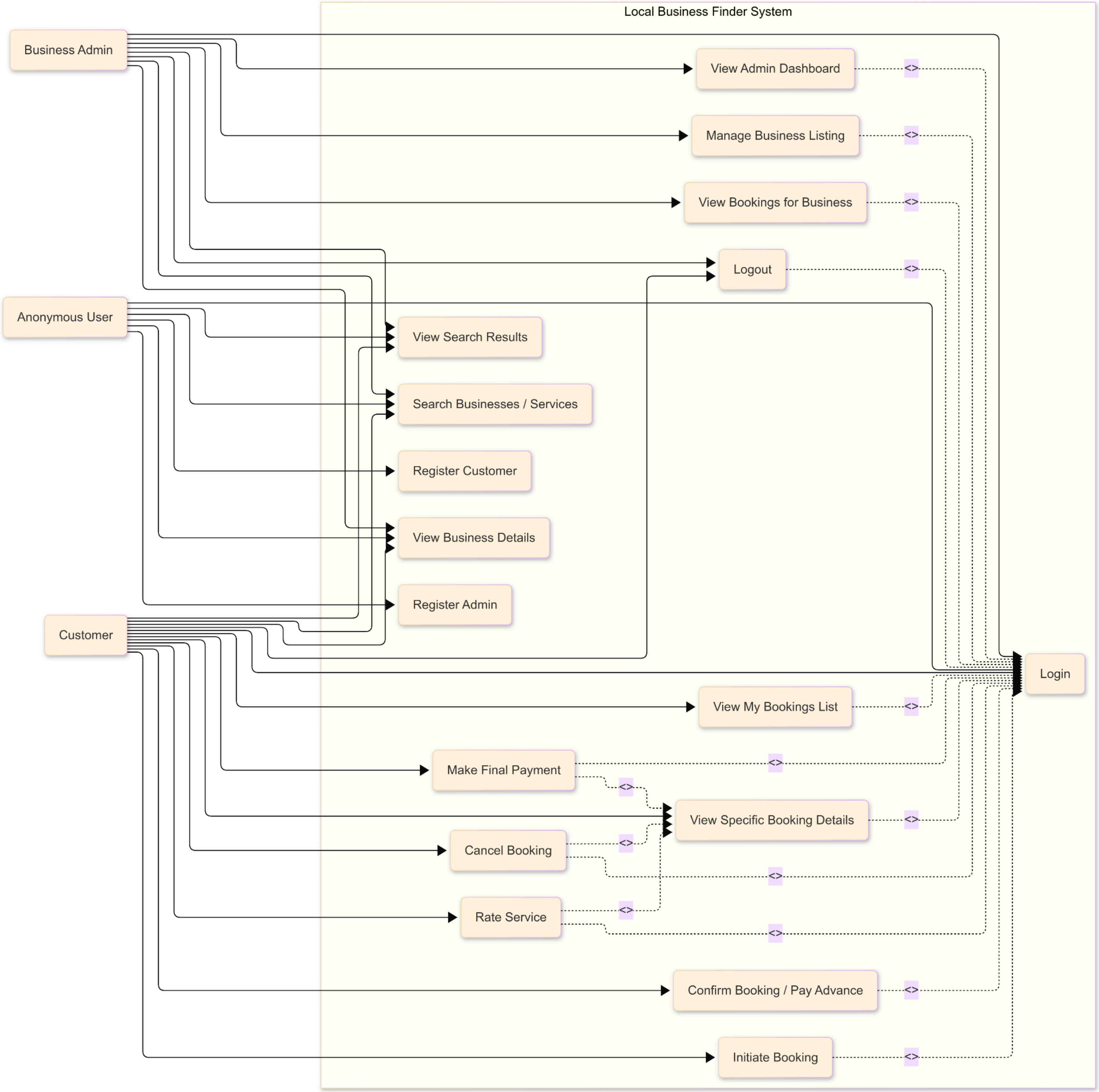
- Business Profile Management: Allows registered admins to create, view, and update their detailed business listing (name, category, address, description, pricing, advance amount, etc.).
- Booking Dashboard: Provides admins with a view of bookings made for their business, including customer details, requested date/time, and booking status. (Implicit availability management via booked slots).
- Admin Account Management: Secure business administrator registration and login functionality.

III. Platform Features:

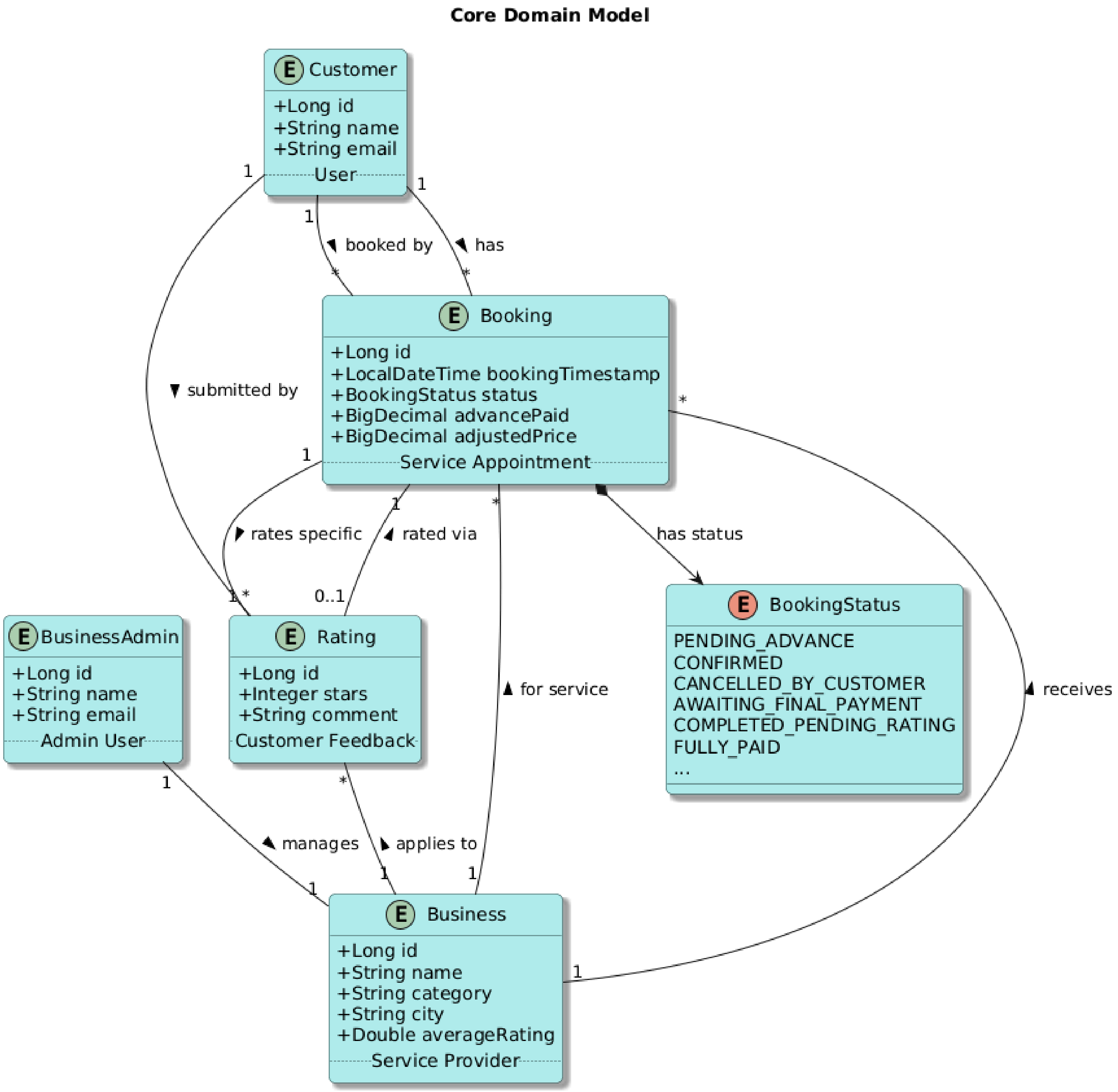
- Role-Based Access Control: Differentiates functionality and access based on user roles (Anonymous, Customer, Business Admin) using Spring Security.
- MVC Architecture: Built using the Model-View-Controller pattern for maintainability and separation of concerns.
- Web-Based Interface: Accessible through a standard web browser using dynamic HTML rendering via Thymeleaf.

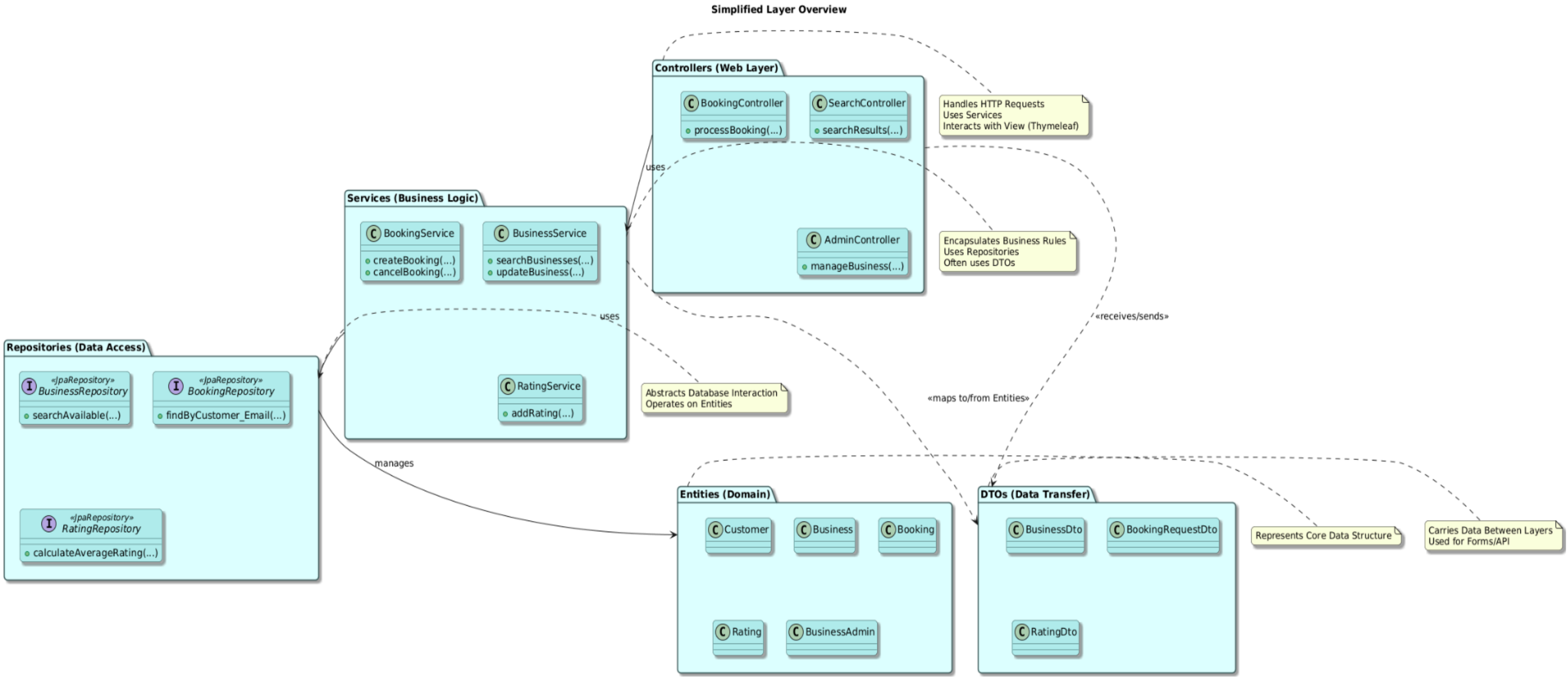
Models:

Use Case Diagram:



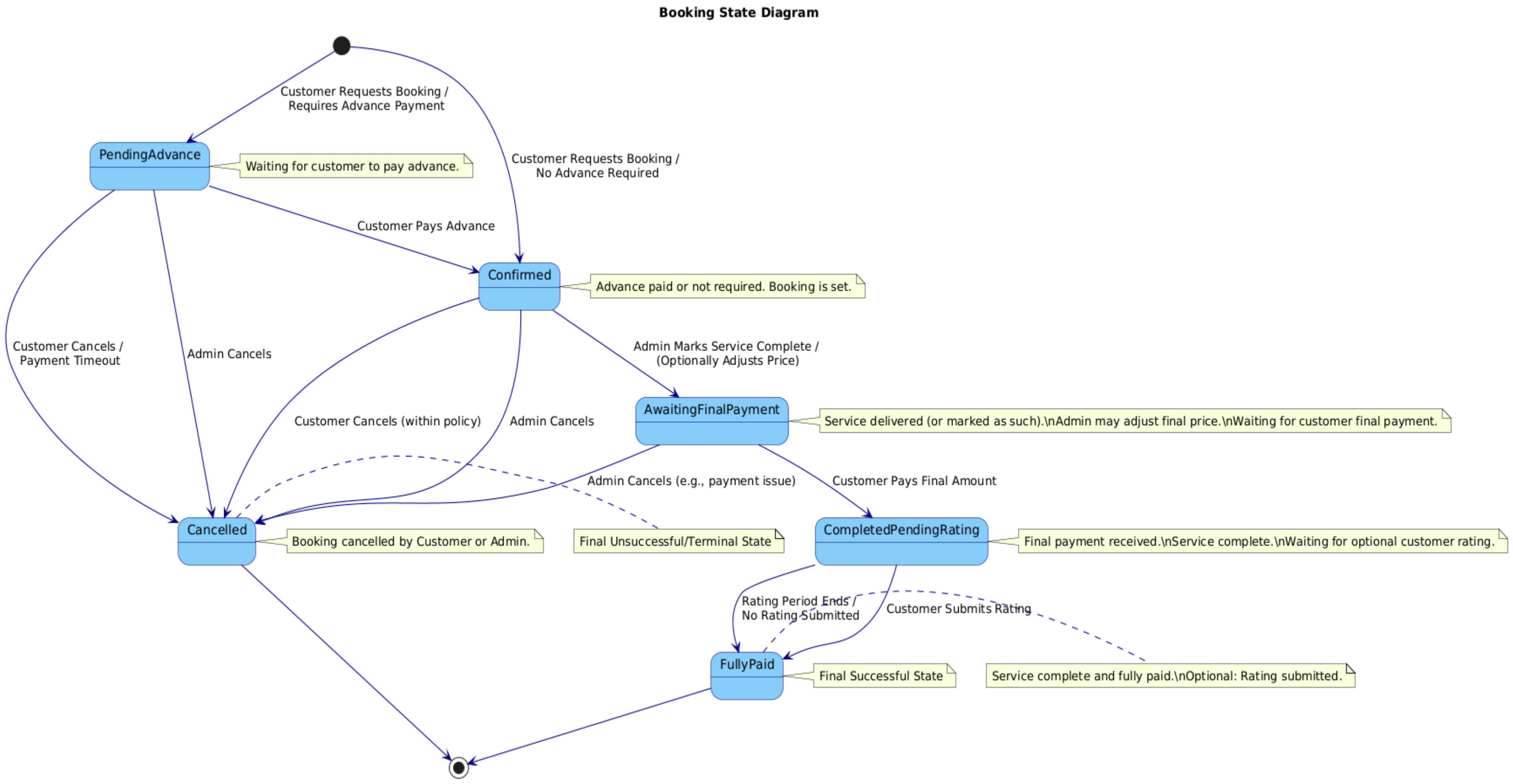
Class Diagram:





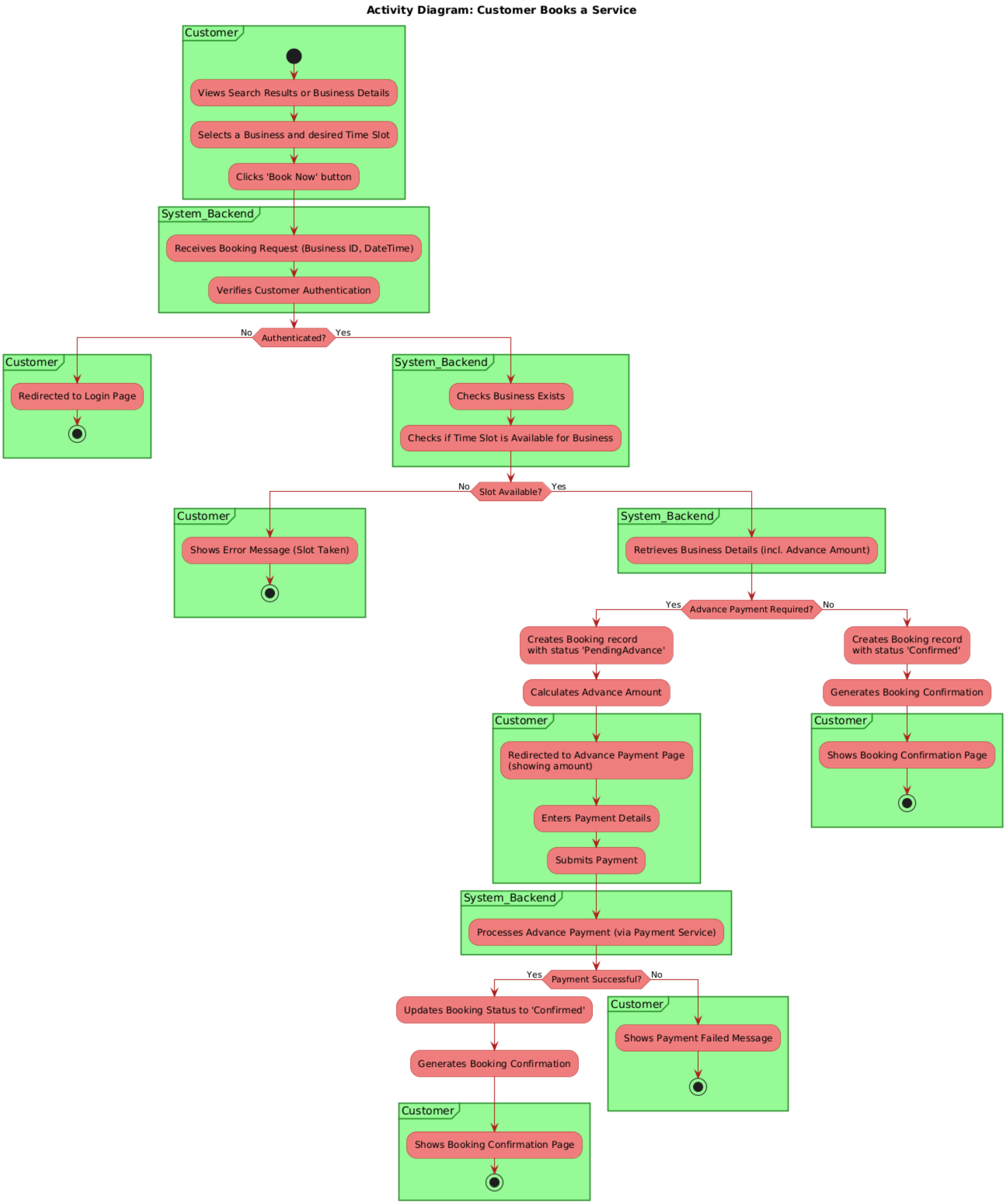
(Simplified Layer Overview)

State Diagram:


Booking State Diagram
Customer Requests Booking /
Requires Advance Payment
Customer Requests Booking /
No Advance Required
PendingAdvance
Waiting for customer to pay advance.
Customer Pays Advance
Confirmed
Advance paid or not required. Booking is set.
Customer Cancels /
Payment Timeout
Admin Cancels
Admin Marks Service Complete /
(Optionally Adjusts Price)
Customer Cancels (within policy)
Admin Cancels
AwaitingFinalPayment
Service delivered (or marked as such).\nAdmin may adjust final price.\nWaiting for customer final payment.
Admin Cancels (e.g., payment issue)
Customer Pays Final Amount
Cancelled
Booking cancelled by Customer or Admin.
Final Unsuccessful/Terminal State
CompletedPendingRating
Final payment received.\nService complete.\nWaiting for optional customer rating.
Rating Period Ends /
No Rating Submitted
Customer Submits Rating
FullyPaid
Final Successful State
Service complete and fully paid.\nOptional: Rating submitted.

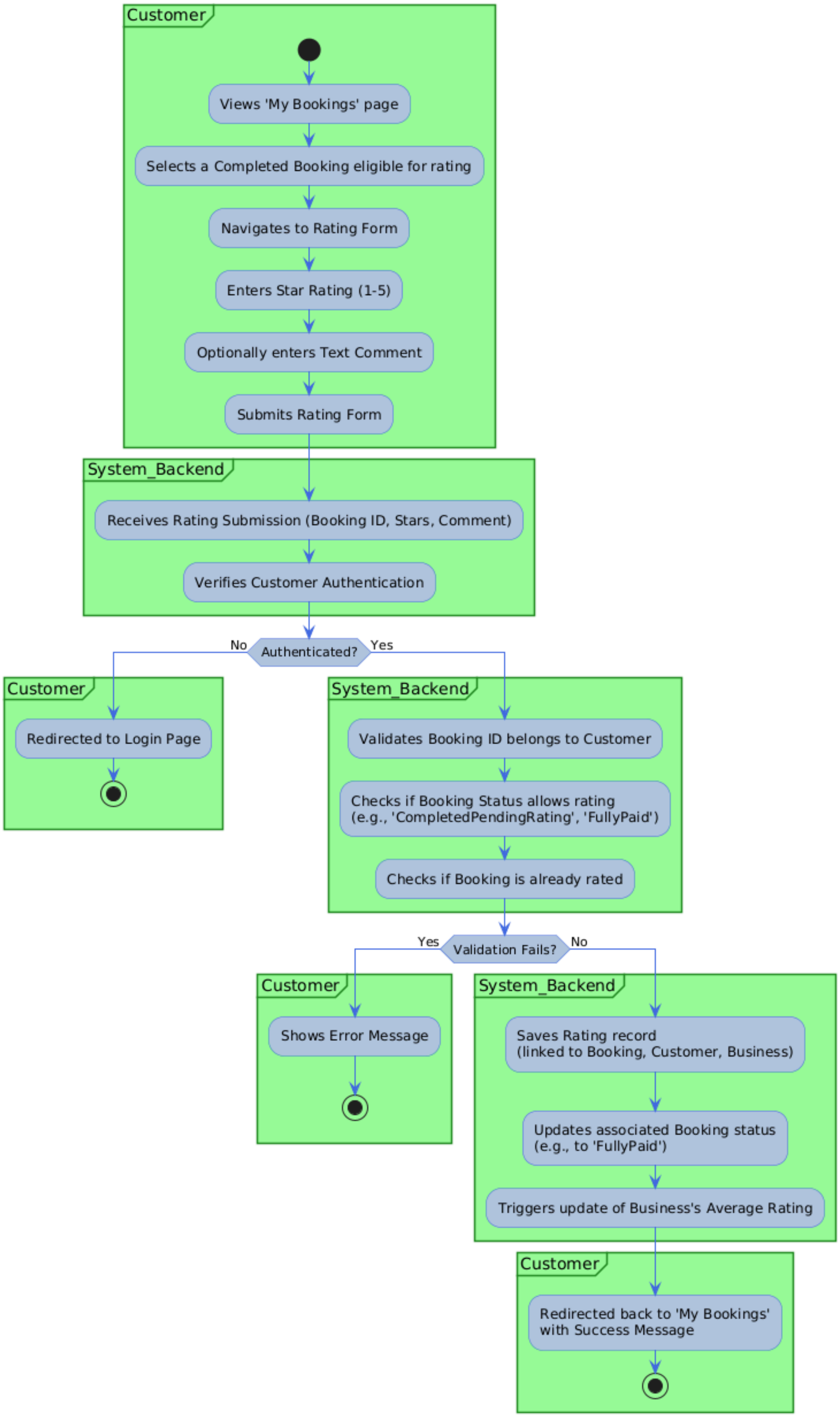
Activity Diagrams:

1. Major Usecase


Activity Diagram: Customer Books a Service
Customer
Views Search Results or Business Details
Selects a Business and desired Time Slot
Clicks 'Book Now' button
System_Backend
Receives Booking Request (Business ID, DateTime)
Verifies Customer Authentication
No
Authenticated?
Yes
Customer
Redirected to Login Page
System_Backend
Checks Business Exists
Checks if Time Slot is Available for Business
No
Slot Available?
Yes
Customer
Shows Error Message (Slot Taken)
System_Backend
Retrieves Business Details (incl. Advance Amount)
Yes
Advance Payment Required?
No
Creates Booking record with status 'PendingAdvance'
Calculates Advance Amount
Creates Booking record with status 'Confirmed'
Generates Booking Confirmation
Customer
Redirected to Advance Payment Page (showing amount)
Enters Payment Details
Submits Payment
Customer
Shows Booking Confirmation Page
System_Backend
Processes Advance Payment (via Payment Service)
Yes
Payment Successful?
No
Updates Booking Status to 'Confirmed'
Generates Booking Confirmation
Customer
Shows Payment Failed Message
Customer
Shows Booking Confirmation Page

2. Minor Use case

Activity Diagram: Customer Submits a Rating


Customer
Views 'My Bookings' page
Selects a Completed Booking eligible for rating
Navigates to Rating Form
Enters Star Rating (1-5)
Optionally enters Text Comment
Submits Rating Form
System_Backend
Receives Rating Submission (Booking ID, Stars, Comment)
Verifies Customer Authentication
No
Authenticated?
Yes
Customer
Redirected to Login Page
System_Backend
Validates Booking ID belongs to Customer
Checks if Booking Status allows rating
(e.g., 'CompletedPendingRating', 'FullyPaid')
Checks if Booking is already rated
Yes
Validation Fails?
No
Customer
Shows Error Message
System_Backend
Saves Rating record
(linked to Booking, Customer, Business)
Updates associated Booking status
(e.g., to 'FullyPaid')
Triggers update of Business's Average Rating
Customer
Redirected back to 'My Bookings'
with Success Message

Architecture Patterns, Design Principles, and Design Patterns:

Architecture Patterns

Model – View – Controller Pattern (MVC)

Model:

- Domain Objects (Entities): Classes like Customer, Business, Booking, Rating, BusinessAdmin annotated with @Entity define the data structure and relationships, managed by Spring Data JPA.
- Repositories (Data Access Layer): Interfaces like CustomerRepository, BusinessRepository extending JpaRepository abstract database interactions.
- Services (Business Logic Layer): Classes like CustomerService, BusinessService, BookingService annotated with @Service encapsulate the core application logic, data validation, and coordination between repositories.

View:

- Thymeleaf Templates: HTML files (.html) located in src/main/resources/templates (e.g., search.html, booking-form.html, admin/dashboard.html) render the user interface dynamically based on data passed from the Controller. They use Thymeleaf's attributes (th:*) to interact with the model data. Reusable parts like headers, footers, and navigation bars are managed using Thymeleaf fragments (fragments.html).

Controller:

- Spring MVC Controllers: Classes like SearchController, BookingController, AdminController annotated with @Controller handle HTTP requests mapped via annotations (@GetMapping, @PostMapping), process input (often bound to DTOs), call service methods, populate the Model object, and return the logical view name for Thymeleaf to resolve.

Design Principles :

Single Responsibility Principle (SRP):

- Description: A class should have only one reason to change, meaning it should encompass a single, well-defined responsibility.

- Application: This principle is evident in the layered architecture.

    1. BookingController is solely responsible for handling web requests related to bookings (displaying forms, processing submissions).
    2. BookingService is solely responsible for the business logic of creating, finding, cancelling, and managing the state of bookings.
    3. BookingRepository is solely responsible for persisting and retrieving Booking entities from the database.
    4. Booking entity is solely responsible for representing the data structure of a booking.
    5. This separation makes the code easier to understand, test, and modify without impacting unrelated components.

Open/Closed Principle (OCP):

- Description: Software entities (classes, modules, functions) should be open for extension but closed for modification. New functionality should ideally be added by adding new code, not changing existing, tested code.

- Application: While direct modification might occur, Spring's structure facilitates this principle.

    1. Service Layer: If a new type of search filter (e.g., "Has Weekend Availability") were needed, modifications would primarily be within the BusinessService and potentially BusinessRepository (perhaps adding a new query method), extending the search capability without necessarily altering the core searchBusinesses method signature drastically or breaking existing controller interactions.
    2. Dependency Injection: Adding new features often involves creating new service classes (e.g., a hypothetical NotificationService) and injecting them where needed, extending functionality without modifying the core classes consuming the new service.

Dependency Inversion Principle (DIP):

- Description: High-level modules (e.g., Controllers) should not depend directly on low-level modules (e.g., specific database access code). Both should depend on abstractions (e.g., interfaces). Abstractions should not depend on details; details should depend on abstractions.

- Application: This is a cornerstone of the Spring framework, heavily utilized in the project via Dependency Injection (DI)

  1. Controllers (BookingController) depend on Service abstractions (BookingService), not directly on Repositories or database logic.
  2. Services (BookingService) depend on Repository interfaces (BookingRepository), which are abstractions over the actual database implementation provided by Spring Data JPA.
  3. Spring's IoC container manages the instantiation and injection (@Autowired) of concrete implementations (like the proxy classes generated by Spring Data JPA for repositories), decoupling the components and making the system flexible and testable.

Don't Repeat Yourself (DRY):

- Description: Avoid duplication of code or logic. Every piece of information or logic should have a single, authoritative representation within the system.

- Application:

  1. Thymeleaf Fragments: Common UI elements (navbar, footer, CSS/JS includes) are defined once in fragments.html and reused across multiple pages (th:replace="~{fragments :: footer}"), preventing duplication in the view layer.
  2. Service Layer Methods: Common operations like checking user existence (customerService.existsByEmail(...)), validating booking availability (bookingService.isSlotAvailable(...) - hypothetical example), or calculating average ratings (ratingService.calculateAverageRatingForBusiness(...)) are encapsulated in service methods, ensuring the logic is defined once and called wherever needed.

Design Patterns

Repository Pattern:

- Description: Decouples the business logic layer from the data access layer by providing collection-like interfaces for accessing domain objects, hiding the underlying persistence mechanism.

- Application:

  Directly implemented using Spring Data JPA. Interfaces like CustomerRepository, BusinessRepository, BookingRepository annotated with @Repository extend JpaRepository. This provides standard CRUD methods (save, findById, findAll, delete) and allows defining custom queries (e.g., findByEmail, findByCustomer_EmailOrderByBookingTimestampDesc) often without writing SQL, fulfilling the pattern's goal of abstracting data access. The Service layer interacts purely with these repository interfaces.

Service Layer Pattern:

- Description: Defines an application's boundary and available operations through a layer of services, which encapsulate the business logic and coordinate responses. It acts as a facade over the domain logic and data access layers.

- Application:

  Implemented using Spring's @Service annotation on classes like BookingService, BusinessService, CustomerService. These classes contain the core business rules (e.g., validating if a booking can be cancelled based on BookingStatus and time, ensuring a business belongs to the correct admin before update, checking for duplicate emails). They orchestrate calls to one or more repositories and provide transactional boundaries (@Transactional - often implicit or explicit). Controllers interact with these services, keeping the business logic out of the web layer.

Data Transfer Object (DTO) Pattern:

- Description: Used to pass data between layers or processes, often aggregating data from multiple sources or providing a specific structure needed by the presentation layer, thus reducing method calls and decoupling layers.

- Application: Widely used throughout the application. Examples:
    1. CustomerDto: Used for handling customer registration form data, separating view concerns (like confirmPassword) from the Customer entity.
    2. BookingRequestDto: Carries data from the booking form (business ID, date/time, notes) to the BookingController and BookingService.
    3. BusinessDto: Used for creating/updating business listings via admin forms.
    4. These DTOs prevent exposing internal entity structures directly to the view/web layer and facilitate input validation (@Valid in controllers). The Service layer is responsible for mapping between DTOs and Entities.

Singleton Pattern (Managed by Framework):

- Description: Ensures that a class has only one instance and provides a global point of access to it.

- Application:

    Spring's core Inversion of Control (IoC) container manages beans (classes annotated with @Component, @Service, @Repository, @Controller, @Configuration) as singletons by default. This means only one instance of CustomerService, BookingController, BusinessRepository, etc., exists within the application context. Spring handles their instantiation and injects them (@Autowired) wherever required. This promotes efficiency and consistent behaviour, leveraging the pattern without requiring manual implementation by the developer.

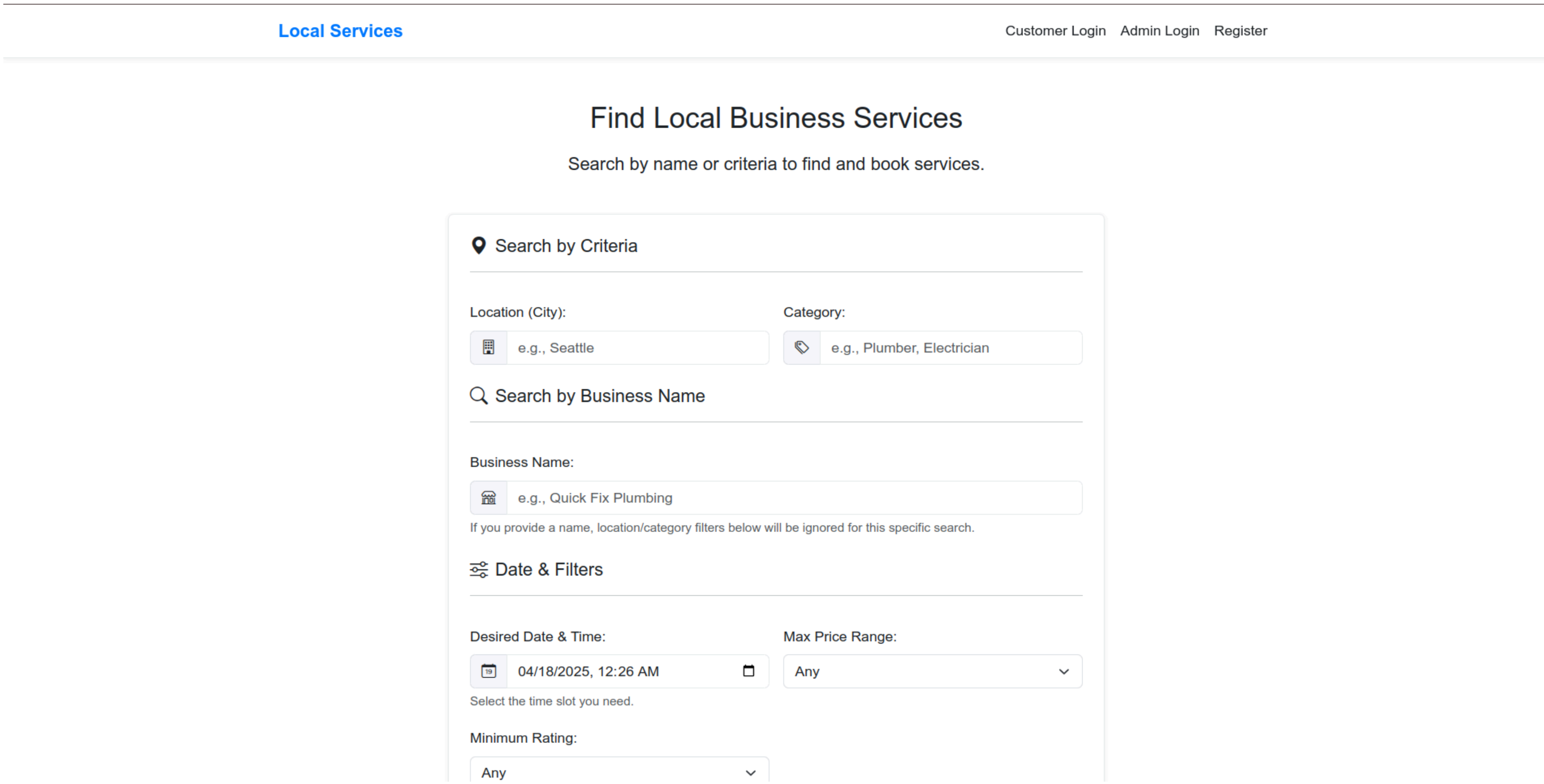
Github link to the Codebase:

https://github.com/dheerajcl/localbusinessfinder_OOAD

Screenshots

UI:

1) Home page:



Find Local Business Services

Search by name or criteria to find and book services.

Search by Criteria

Location (City): e.g., Seattle

Category: e.g., Plumber, Electrician

Search by Business Name

Business Name: e.g., Quick Fix Plumbing

If you provide a name, location/category filters below will be ignored for this specific search.

Date & Filters

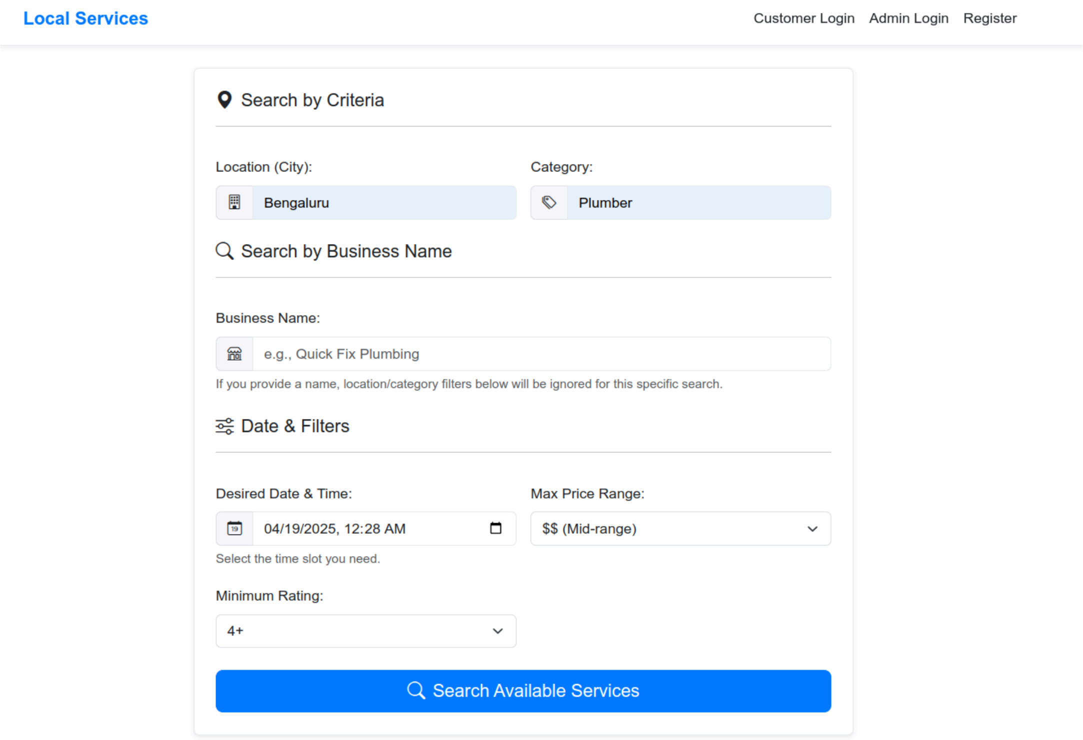
Desired Date & Time: 04/18/2025, 12:26 AM

Select the time slot you need.

Max Price Range: Any

Minimum Rating: Any

2) Advanced Search functionality with multiple filters:



Search by Criteria

Location (City): Bengaluru

Category: Plumber

Search by Business Name

Business Name: e.g., Quick Fix Plumbing

If you provide a name, location/category filters below will be ignored for this specific search.

Date & Filters

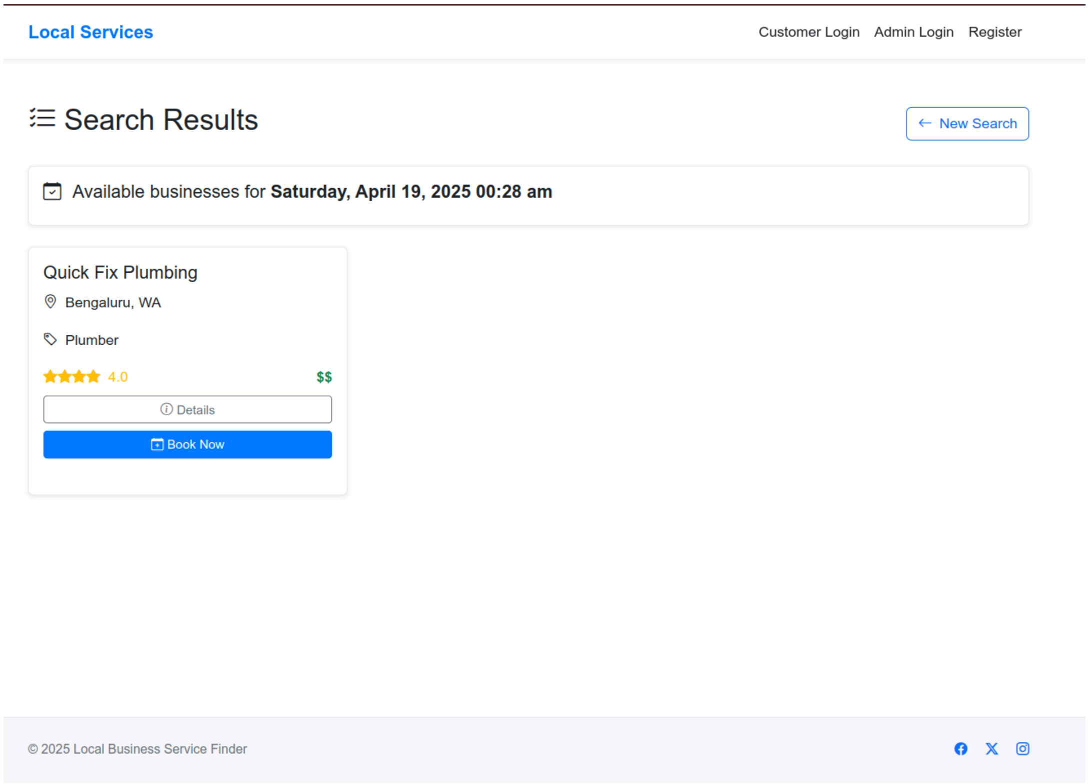
Desired Date & Time: 04/19/2025, 12:28 AM

Select the time slot you need.

Max Price Range: $$ (Mid-range)

Minimum Rating: 4+

3) Search Results:



Search Results

Available businesses for **Saturday, April 19, 2025 00:28 am**

Quick Fix Plumbing

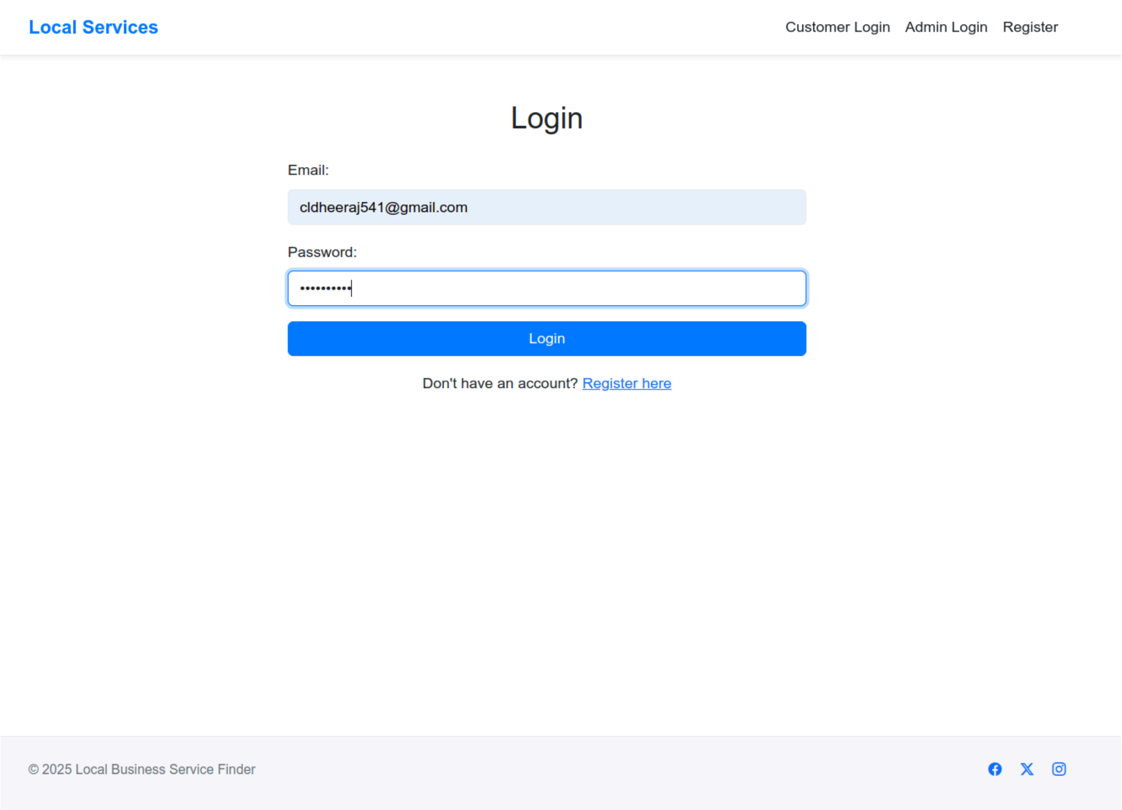
Bengaluru, WA

Plumber

★★★★ 4.0 $$

© 2025 Local Business Service Finder

4) Now, to book the service I need to Login:



Login

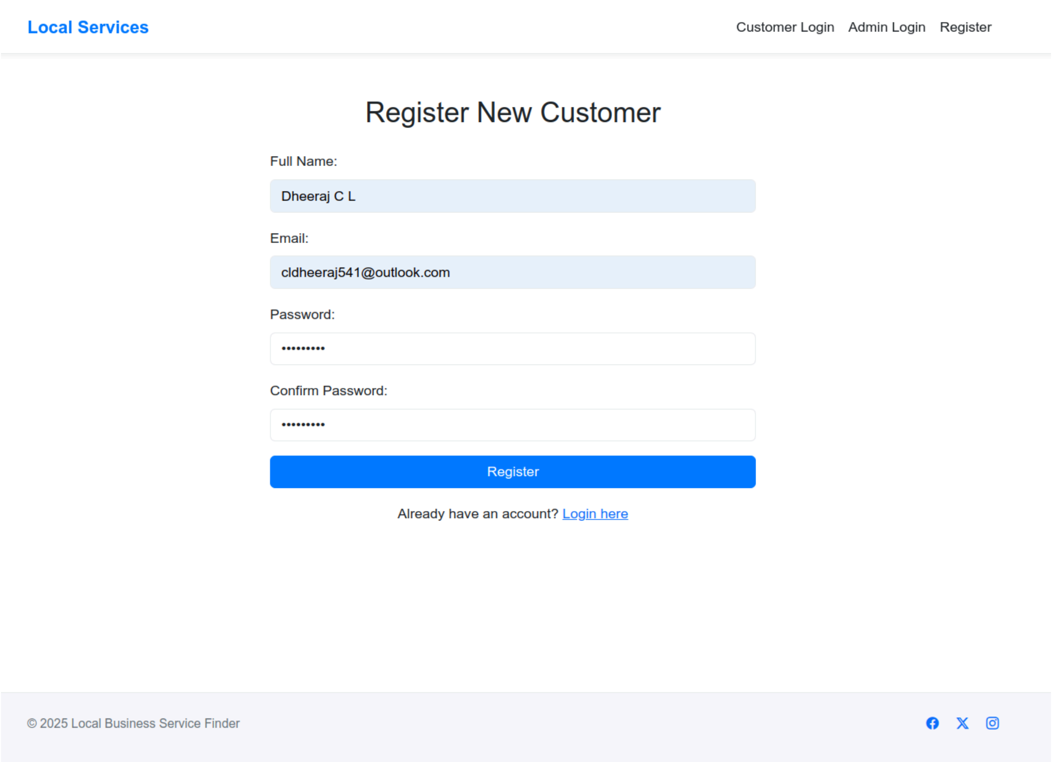
Email: cldheeraj541@gmail.com

Password: ••••••••••

Don't have an account? Register here

© 2025 Local Business Service Finder

5) If new Customer, should register first:



Register New Customer

Full Name: Dheeraj C L

Email: cldheeraj541@outlook.com

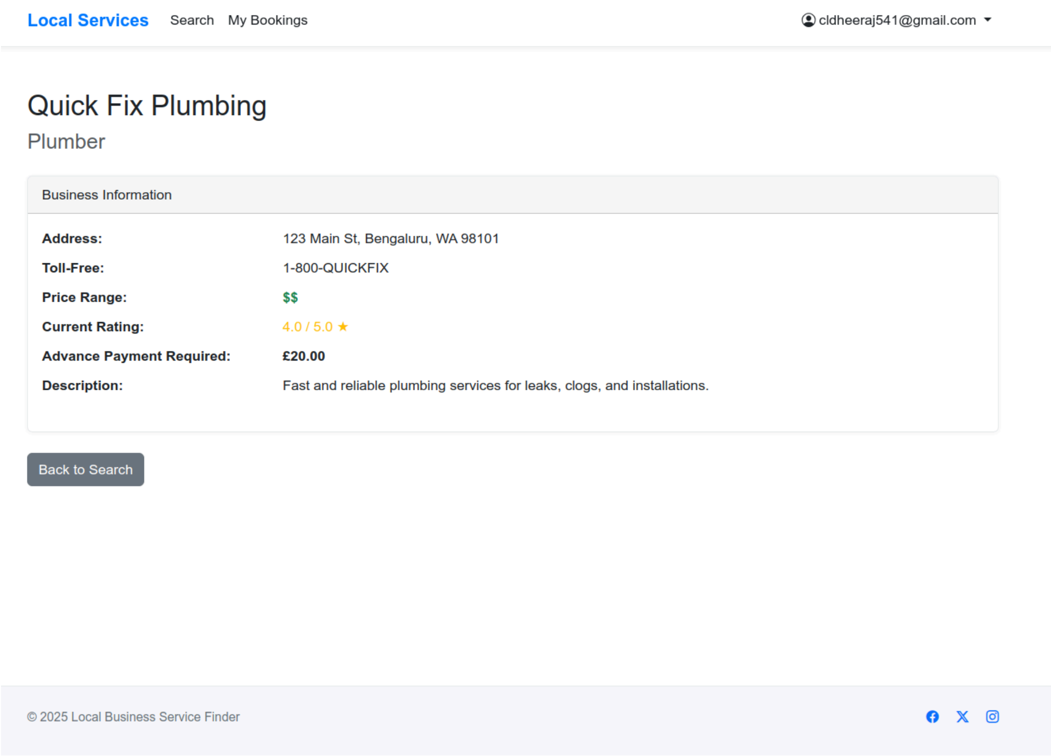
Password: •••••••••

Confirm Password: •••••••••

Already have an account? Login here

© 2025 Local Business Service Finder

6) Service Details:



Quick Fix Plumbing

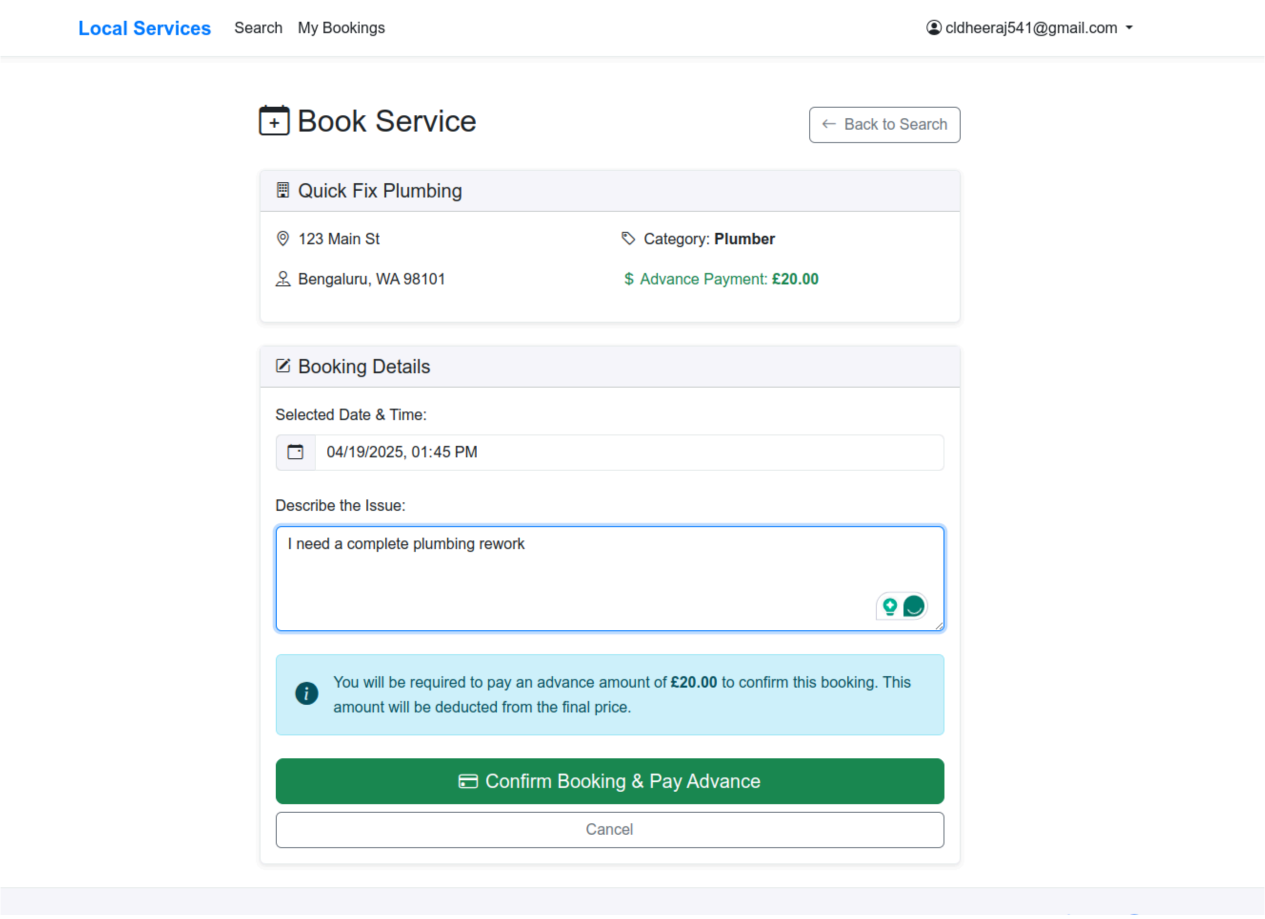
Plumber

Business Information

Address:	123 Main St, Bengaluru, WA 98101
Toll-Free:	1-800-QUICKFIX
Price Range:	$$
Current Rating:	4.0 / 5.0 ★
Advance Payment Required:	**£20.00**
Description:	Fast and reliable plumbing services for leaks, clogs, and installations.

© 2025 Local Business Service Finder

7) Booking page:



Book Service

Quick Fix Plumbing

123 Main St

Bengaluru, WA 98101

Category: **Plumber**

$ Advance Payment: **£20.00**

Booking Details

Selected Date & Time:

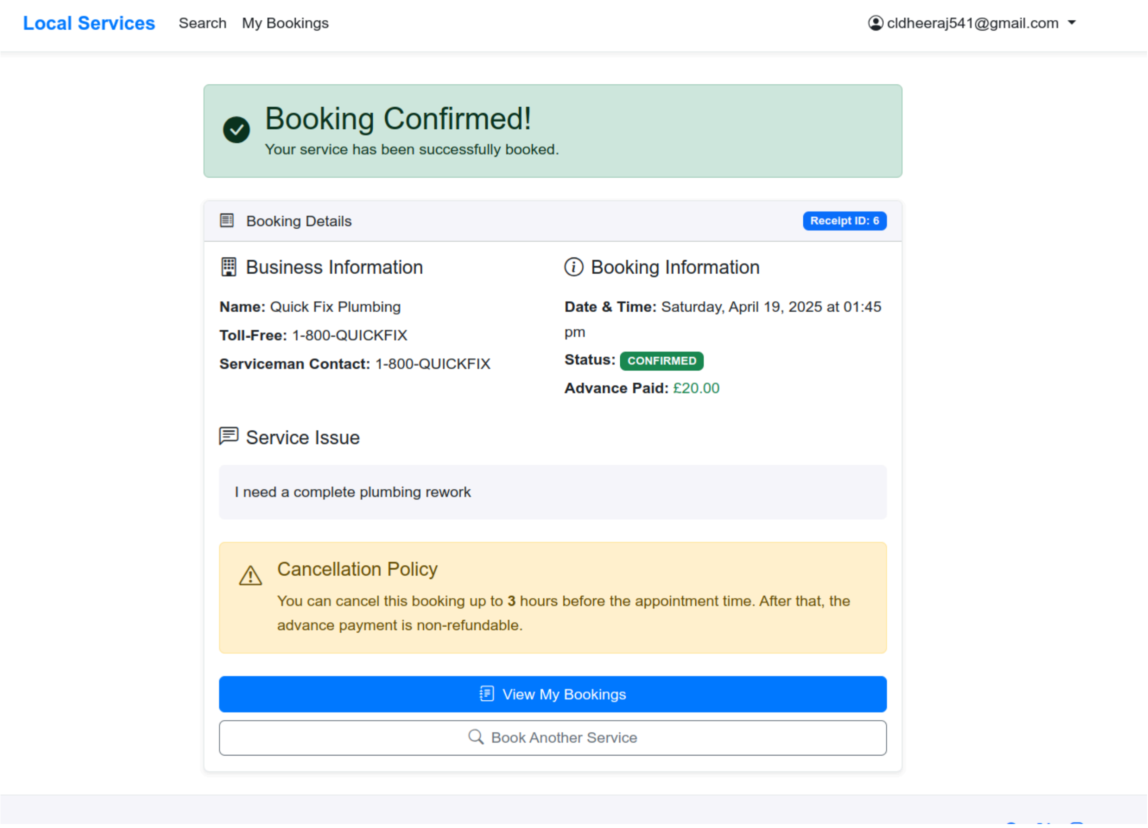
04/19/2025, 01:45 PM

Describe the Issue:

I need a complete plumbing rework

You will be required to pay an advance amount of **£20.00** to confirm this booking. This amount will be deducted from the final price.

8) Booking Confirmation:



Booking Confirmed!

Your service has been successfully booked.

Booking Details — Receipt ID: 6

Business Information

Name: Quick Fix Plumbing

Toll-Free: 1-800-QUICKFIX

Serviceman Contact: 1-800-QUICKFIX

Booking Information

Date & Time: Saturday, April 19, 2025 at 01:45 pm

Status: CONFIRMED

Advance Paid: £20.00

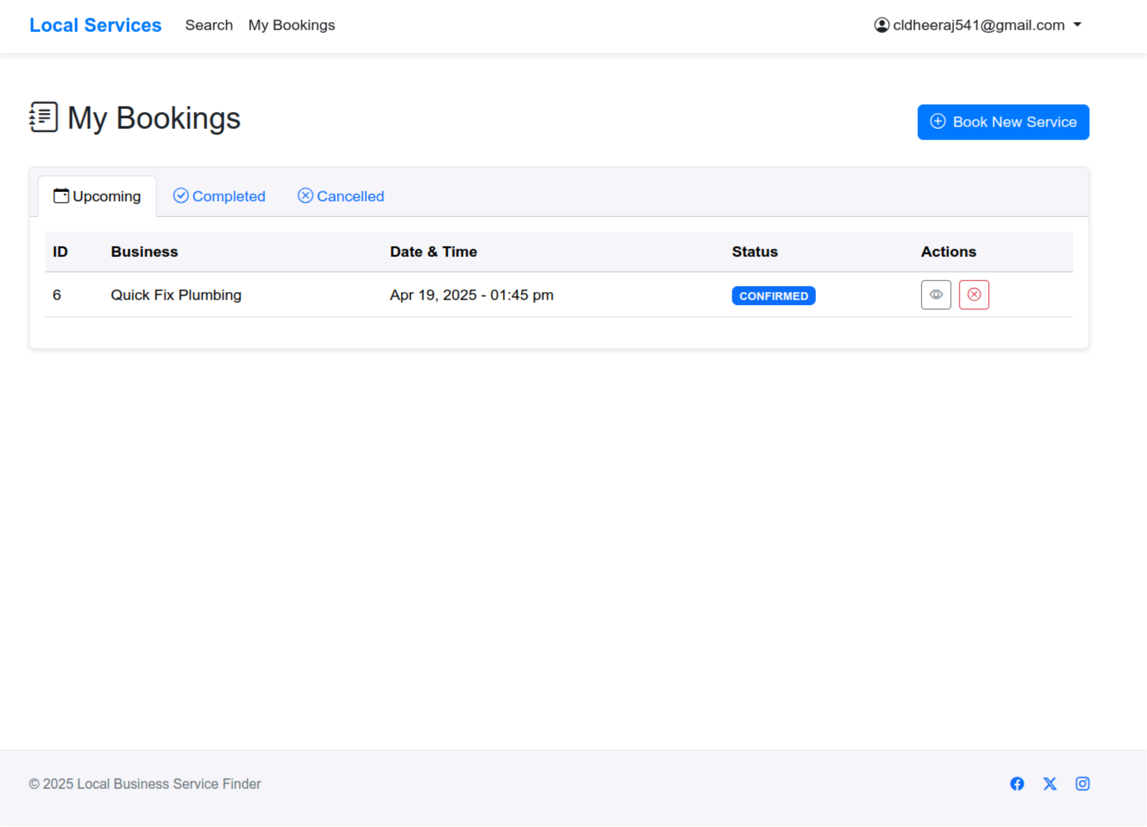
Service Issue

I need a complete plumbing rework

Cancellation Policy

You can cancel this booking up to **3** hours before the appointment time. After that, the advance payment is non-refundable.

9) Customer Dashboard (Customer can view current, completed and cancelled bookings here) :

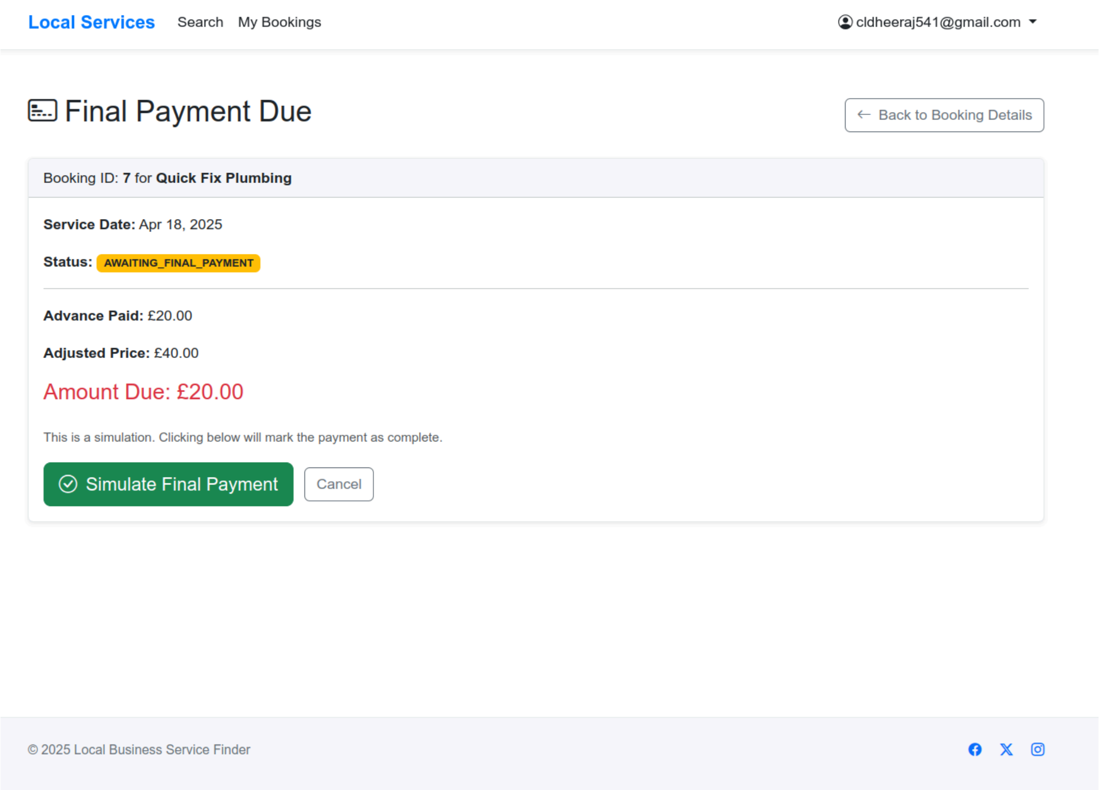


My Bookings

ID	Business	Date & Time	Status	Actions
6	Quick Fix Plumbing	Apr 19, 2025 - 01:45 pm	CONFIRMED	👁 ⊗

© 2025 Local Business Service Finder

10) After the completion of the service, the Customer can pay the Final Price from his Dashboard:



Final Payment Due

Booking ID: **7** for **Quick Fix Plumbing**

Service Date: Apr 18, 2025

Status: AWAITING_FINAL_PAYMENT

Advance Paid: £20.00

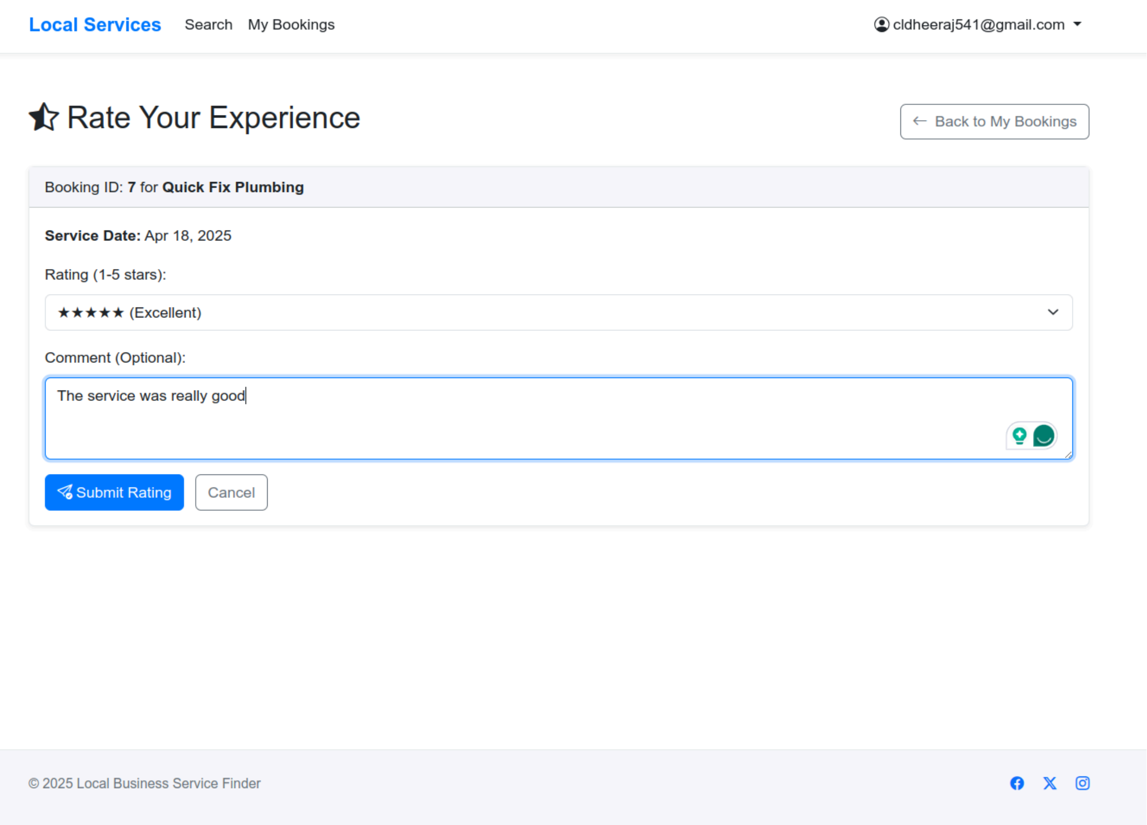
Adjusted Price: £40.00

Amount Due: £20.00

This is a simulation. Clicking below will mark the payment as complete.

© 2025 Local Business Service Finder

11) Ratings for the Service:



Rate Your Experience

Booking ID: **7** for **Quick Fix Plumbing**

Service Date: Apr 18, 2025

Rating (1-5 stars):

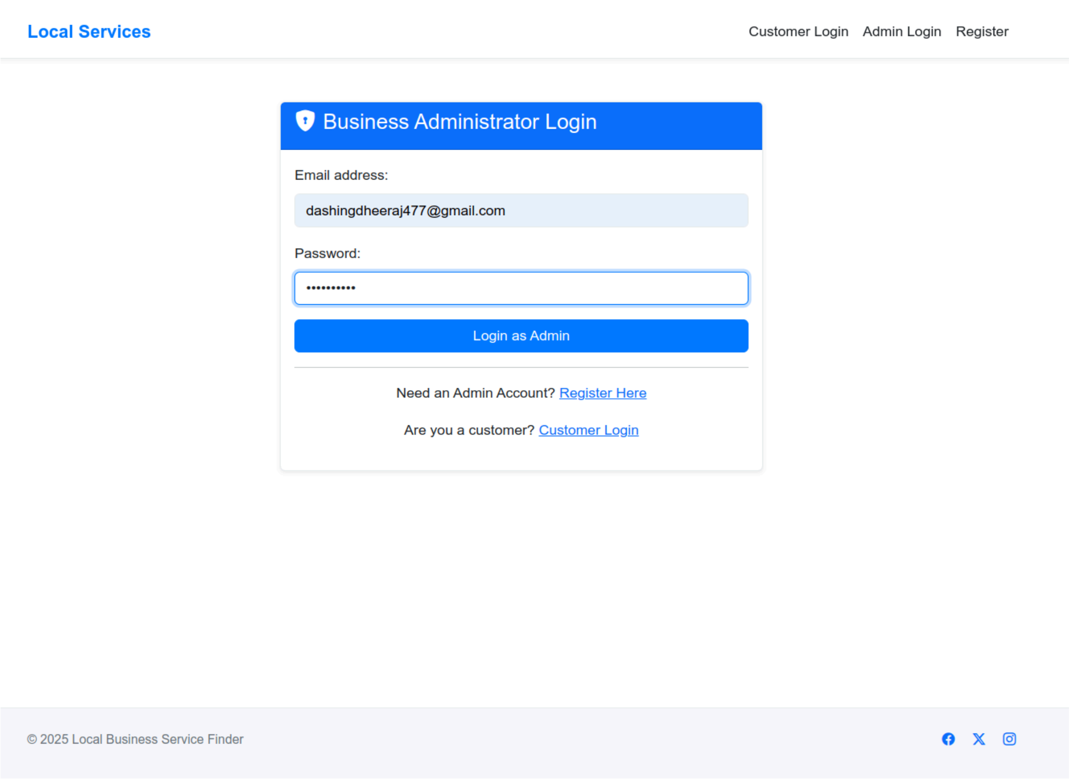
★★★★★ (Excellent)

Comment (Optional):

The service was really good

© 2025 Local Business Service Finder

12) Business admins can register and login from a separate URL endpoint:



Local Services

Customer Login Admin Login Register

Business Administrator Login

Email address:

dashingdheeraj477@gmail.com

Password:

••••••••••

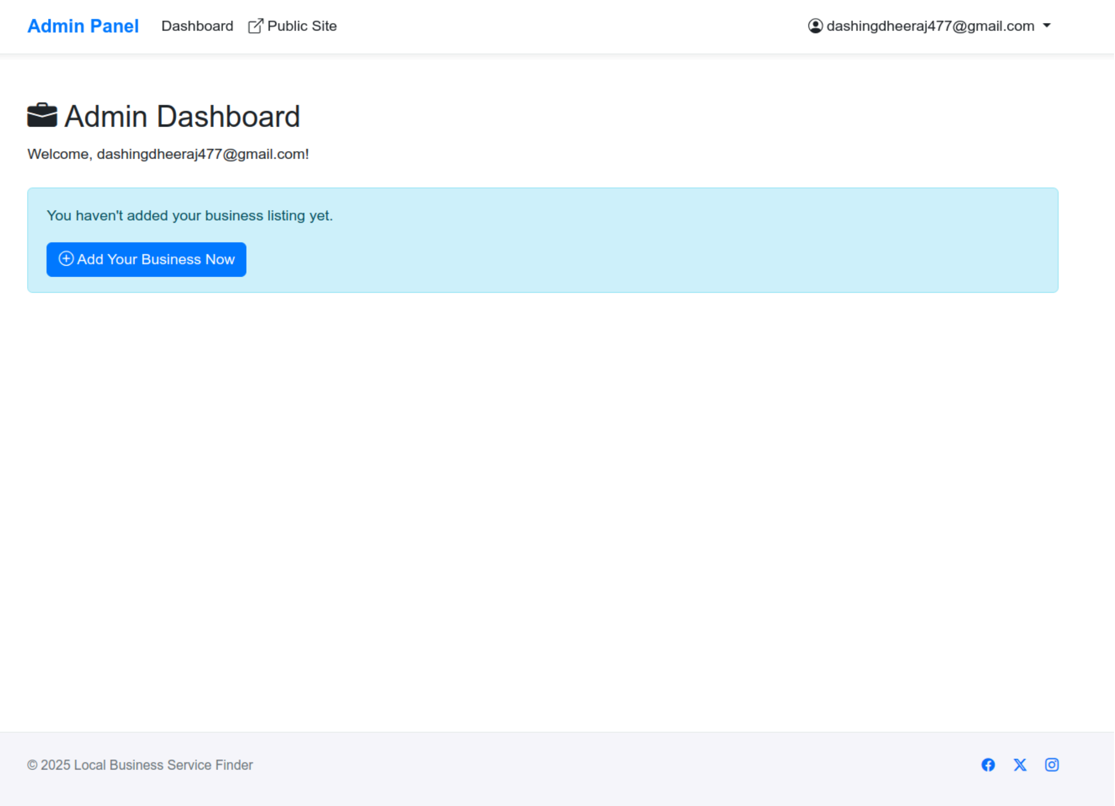
Login as Admin

Need an Admin Account? Register Here

Are you a customer? Customer Login

© 2025 Local Business Service Finder

13) Admin Dashboard:



Admin Panel Dashboard Public Site dashingdheeraj477@gmail.com

Admin Dashboard

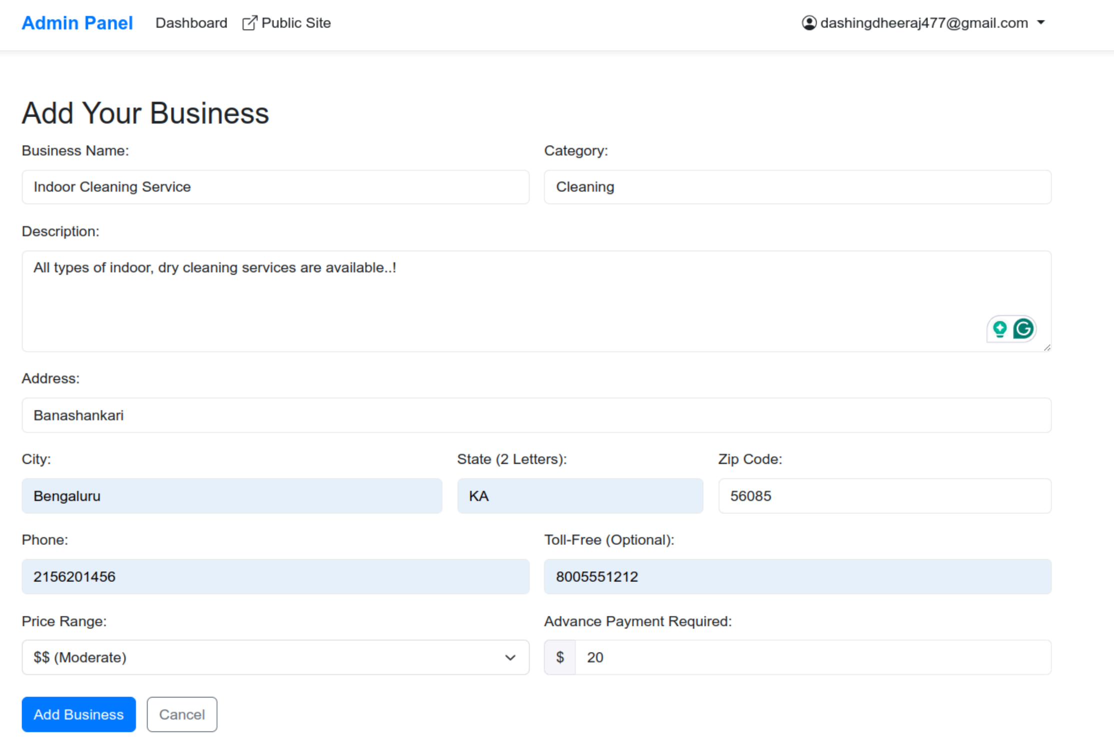
Welcome, dashingdheeraj477@gmail.com!

You haven't added your business listing yet.

Add Your Business Now

© 2025 Local Business Service Finder

14) Business admin adding his service:



Add Your Business

Business Name: Indoor Cleaning Service

Category: Cleaning

Description: All types of indoor, dry cleaning services are available..!

Address: Banashankari

City: Bengaluru

State (2 Letters): KA

Zip Code: 56085

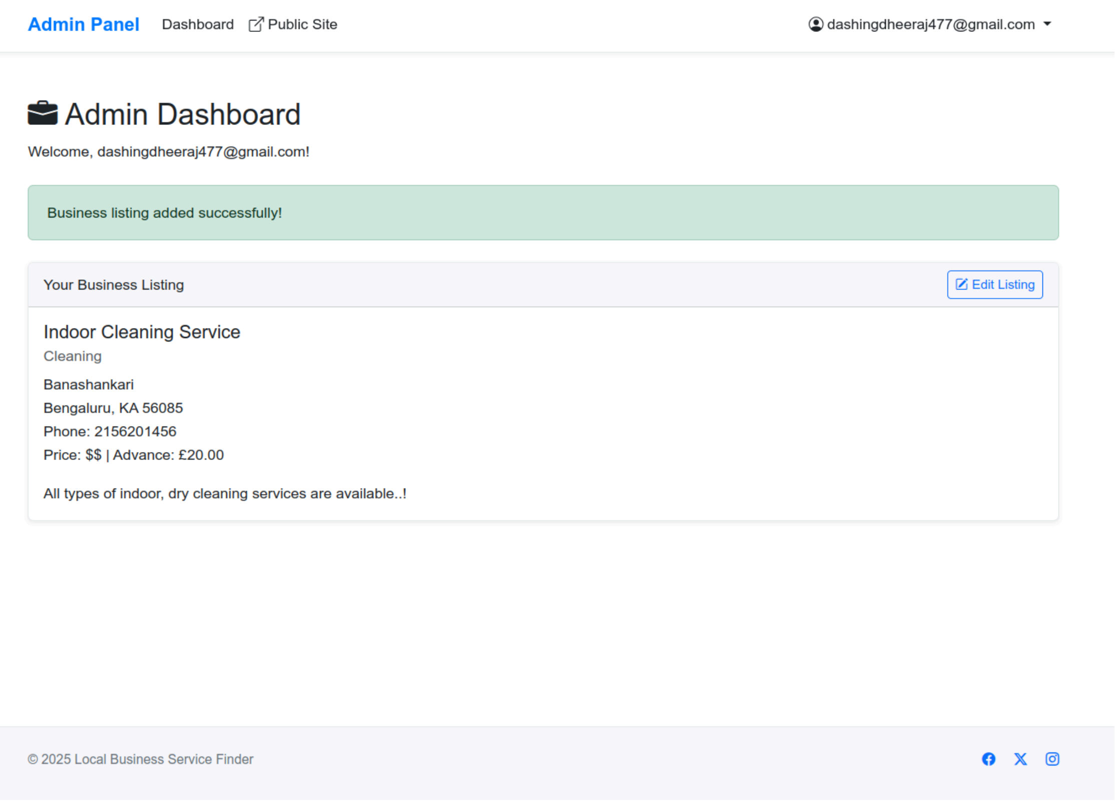
Phone: 2156201456

Toll-Free (Optional): 8005551212

Price Range: $$ (Moderate)

Advance Payment Required: $ 20

15) Business added successfully (can edit too):



Admin Dashboard

Welcome, dashingdheeraj477@gmail.com!

Business listing added successfully!

Your Business Listing

Indoor Cleaning Service

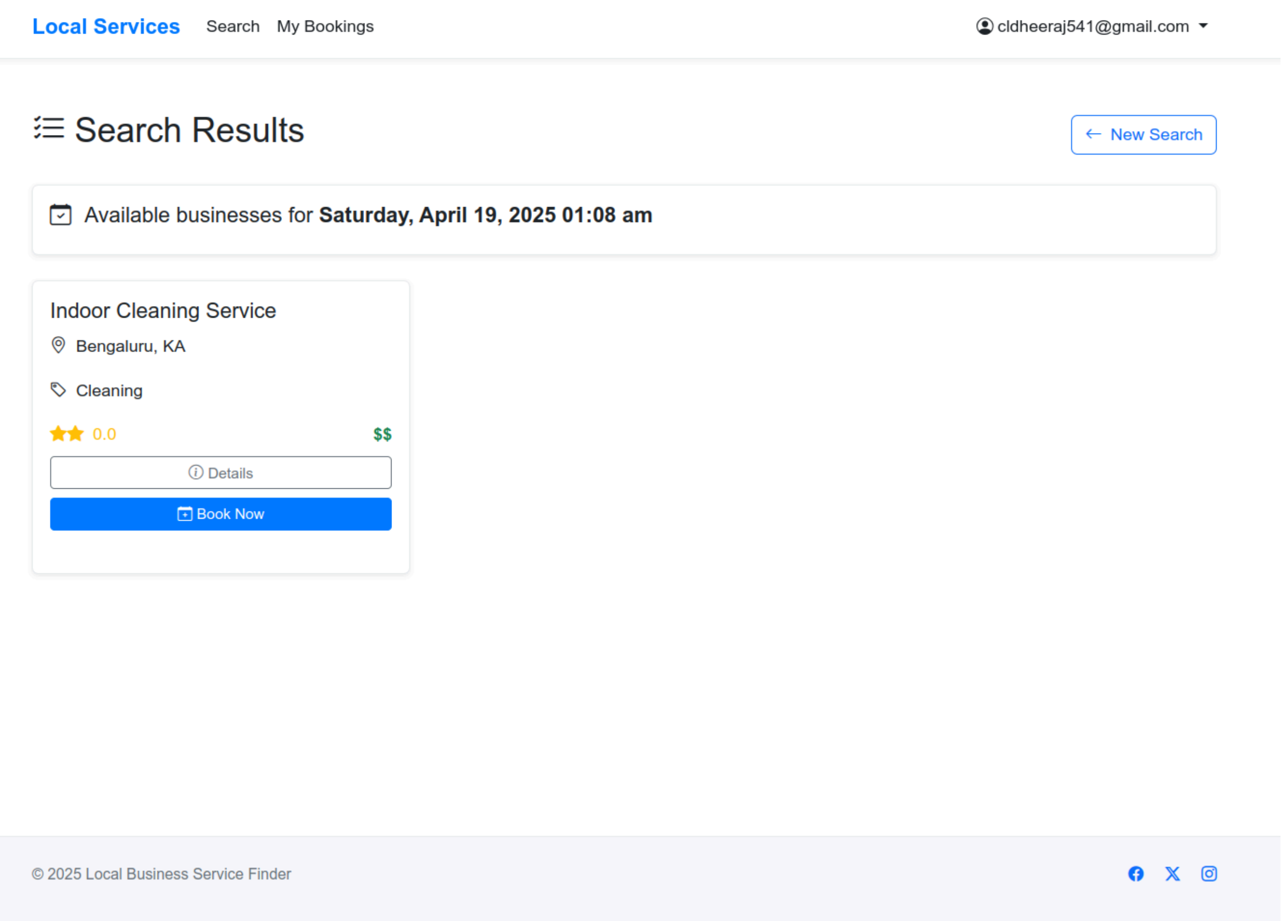
Cleaning

Banashankari
Bengaluru, KA 56085
Phone: 2156201456
Price: $$ | Advance: £20.00

All types of indoor, dry cleaning services are available..!

© 2025 Local Business Service Finder

16) The newly added business is now visible to users



Search Results

Available businesses for **Saturday, April 19, 2025 01:08 am**

Indoor Cleaning Service
Bengaluru, KA
Cleaning
★★ 0.0 $$

© 2025 Local Business Service Finder

Individual contributions of the team members:

Name	**Module worked on**
C H Nikhil	Authentication & User Management:
Dhanush M	Service Search & Business Display:
Dheeraj C L	Booking Workflow & Customer Management:
Chandan H S	Admin Business Management & Ratings: