Technical Design Document (TDD) – *Go Local*

1 INTRODUCTION:

The Go Local is a community-driven web application aimed at simplifying the process of finding and connecting with trusted local service providers. It addresses the common challenge of locating reliable professionals—such as electricians, plumbers, tutors, and more—by allowing users to search based on location, view verified profiles, read community reviews, and make service bookings. The platform promotes transparency, trust, and convenience through features like user ratings, service recommendations, and real-time notifications, while also providing service providers with tools to manage their offerings and engage with potential customers.

2 TECHNOLOGY STACK:

Layer	Technology Used
Frontend	React, Bootstrap
Backend	Java (Spring Boot)
Database	MySQL
API Format	RESTful (JSON)
Version Control	GitHub

3 SYSTEM ARCHITECTURE:

3.1 High Level Architechture:

The Go Local platform is a web-based community service provider system designed to connect local service providers with customers. It features multiple user roles — Customers, Service Providers (Helpers), and Administrators — each with tailored functionality. The architecture ensures smooth interaction, data management, and security throughout the platform.

3.2 Components:

3.2.1 Client Side (Browser / User Interface)

- Technology: Responsive web UI (React, Bootstrap , Spring Boot).
- Responsibilities:
  - Display landing page with search, top performers, and customer testimonials.
  - Provide forms for Service Provider and Customer sign-ups, and Login/Registration functionality.
  - Render dashboards for logged-in users with personalized information (profile, booking history, etc.).

  - Facilitate navigation to services listing, detailed service profiles, and booking workflows.
  - Display dynamic content such as top services filtered by city and profession.
  - Show ratings and contact information clearly.
  - Maintain branding consistency via logos, images, and UI themes.

3.2.2 Web Server (Spring Boot Application)

- Technology: Spring Boot framework (Java) for backend logic.
- Responsibilities:
  - User Management: Handle registration, authentication, password recovery.
  - Role-Based Access: Differentiate behaviour for Customers, Service Providers, and Admins.
  - Business Logic: Manage service listings, provider profiles, booking requests and ratings.
  - Search & Filtering: Process user queries based on location, profession, and other filters.
  - Data Flow Control: Validate inputs from the UI, transform data for persistence, and retrieve data for display.
  - Admin Dashboard: Oversee user activity, manage reports, and monitor content credibility.
  - File Handling: Support image uploads for profile pictures and service banners

3.2.3 Database (MySQL)

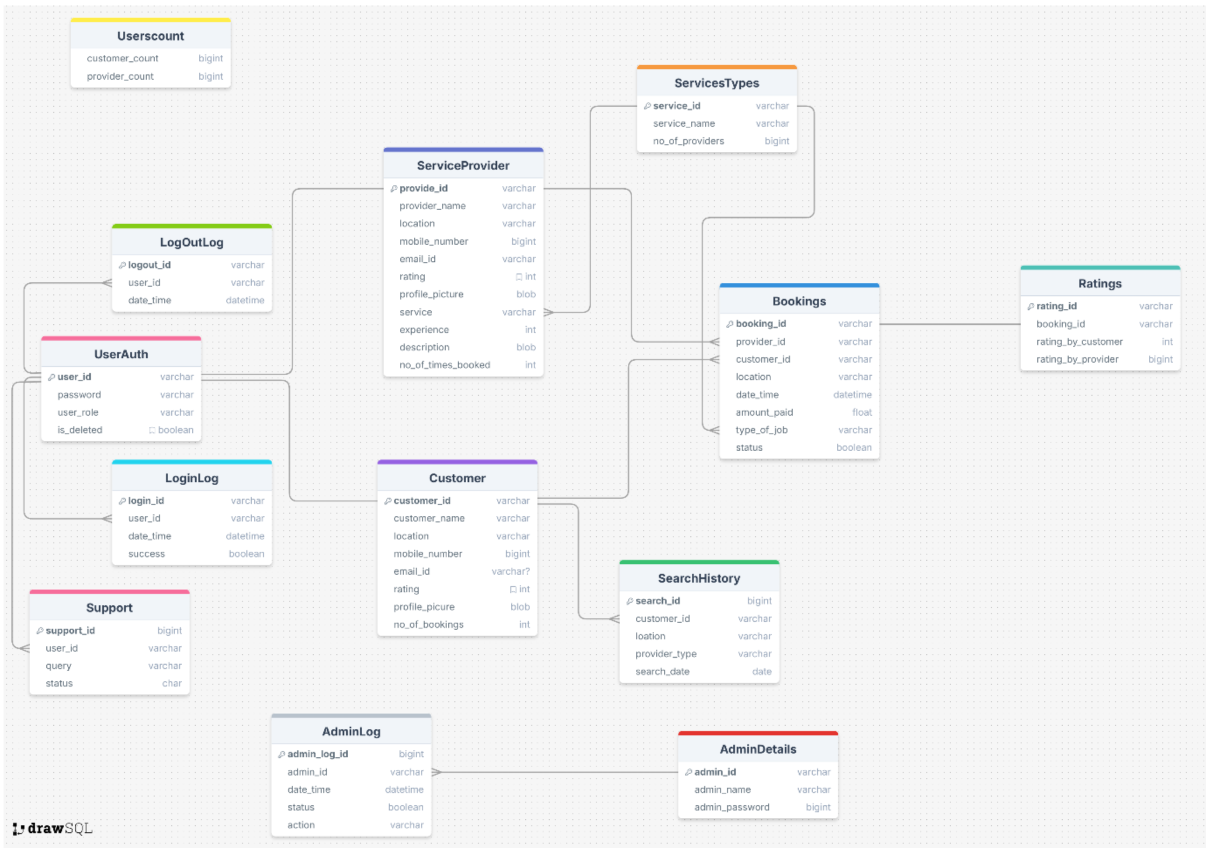
- Technology: Relational Database Management System (MySQL).
- Responsibilities:
  - Store user profiles including service providers and customers with fields such as name, contact info, location, and photographs.
  - Maintain service listings with service details, categories, and provider associations.
  - Record user ratings linked to service providers.
  - Log booking history and status updates for customers and helpers.
  - Manage authentication data including usernames, hashed passwords, and roles.
  - Store admin data and activity logs.

3.3 System Workflow Summary

1. Landing Page Interaction: Users arrive, browse top performers, read reviews, or search services.
2. Registration: Users choose to sign up as Customers or Service Providers via separate sign-up forms.
3. Authentication: Login via username/password with appropriate authentication checks.
4. User Dashboard: Post-login, users access a personalized dashboard displaying services, search options, profile info, and booking history.

5. Service Discovery: Customers search for services by location and profession; filtered results display service provider listings.
6. Service Provider Profile: Detailed view of selected service providers with ratings, completed work, and contact info.
7. Admin Monitoring: Admins monitor platform activity, user reports, and ensure data integrity.

4. DATABASE IMPLEMENTATION:



Key entities:

4.1. Usercount

The Usercount table is a utility table that maintains a record of the total number of customer and provider logins. It serves as an analytical or statistical record for overall system usage by type of user.

Columns:

1. customer_count (bigint) – Represents the total number of customers who have created account into the system.
2. provider_count (bigint) – Represents the total number of service providers who has account.

This table is standalone and not directly connected to any other table. It may be updated via backend logic during login operations.

4. 2. LogOutLog

This table tracks logout activities by users. It helps monitor user session durations and supports auditing and security measures.

Columns:

1. logout_id (PK) (varchar) – Unique identifier for the logout session.
2. user_id (FK)(varchar) – ID of the user who logged out.
3. date_time (datetime) – Timestamp indicating when the logout occurred.

Connected to UserAuth through user_id (foreign key relation implied but not drawn).

4. 3. UserAuth

UserAuth handles the authentication credentials of users in the system. This table manages login credentials and basic roles.

Columns:

1. user_id (PK)(varchar) – Unique ID for the user (could be customer, provider).
2. password (varchar) – Encrypted user password.
3. user_role (varchar) – Specifies the type of user: customer, or provider.
4. is_deleted (boolean) – Indicates whether the user account has been deleted or deactivated.

Referenced by LoginLog, LogOutLog, and Support. Acts as a central table for user-based operations.

4. 4. LoginLog

Tracks all login attempts by users, both successful and unsuccessful. Useful for security monitoring and analytics

Columns:

1. login_id (PK)(varchar) – Unique ID for each login attempt.
2. user_id (FK)(varchar) – The ID of the user attempting to log in.
3. date_time (datetime) – Timestamp of the login attempt.
4. success (boolean) – Whether the login was successful.

Connected to UserAuth via user_id.

4. 5. Support

The Support table stores user queries related to technical or general help. It is used by support staff to resolve user-reported issues.

Columns:

1. support_id(PK) (bigint) – Unique identifier for each support query.

2. user_id(FK) (varchar) – ID of the user raising the issue.

3. query (varchar) – Description of the query or issue.

4. status (varchar) – Status of the query (e.g., pending, resolved).

Connected to UserAuth via user_id.

4. 6. ServiceProvider

This table stores detailed information about the service providers registered on the platform.

Columns:

1. provider_id (PK)(varchar) – Unique identifier for each provider.

2. provider_name (varchar) – Name of the service provider.

3. gender (varchar) – Gender of the provider.

4. dob (date) – Date of birth.

5. mobile_number (varchar) – Contact number.

6. email (varchar) – Email address.

7. rating (float) – Average rating received from customers.

8. no_of_bookings (bigint) – Total number of bookings fulfilled.

9. location (varchar) – Operational location.

10. experience (float) – Years of experience.

11. type_of_service (varchar) – Type of services provided.

12. status (varchar) – Current availability status.

13. profile_image (blog) – Profile picture.

Connected to Bookings via provider_id.

Connected to ServicesTypes via type_of_service.

Connected to Ratings indirectly through Bookings.

4. 7. Customer

Stores information about customers who use the platform to avail services.

Columns:

1. customer_id (PK)(varchar) – Unique identifier for each customer.

2. customer_name (varchar) – Name of the customer.

3. gender (varchar) – Gender.

4. dob (date) – Date of birth.

5. mobile_number (varchar) – Contact number.

6. email (varchar) – Email address.

7. location (varchar) – Address or area of operation.

8. rating (float) – Customer's own average rating.

9. no_of_bookings (bigint) – Total bookings made.

10. profile_picture (blob) – Profile image.

Connected to Bookings via customer_id.

Connected to Ratings indirectly through Bookings.

4. 8. Search History

Records search queries performed by customers, helping to provide personalized recommendations and analytical insights.

Columns:

1. search_id (PK)(varchar) – Unique ID for the search query.

2. Customer_id (FK)(varchar) – Customer performing the search.

3. location (varchar) – Location of search.

4. provider_type (varchar) – Type of provider.

5. search_date (datetime) – Timestamp of search.

Connected to Customer or UserAuth via user_id.

4. 9. AdminLog

Tracks admin activities such as updates, deletions, or system management actions for audit purposes.

Columns:

1. admin_log_id(PK) (varchar) – Unique log ID.

2. admin_id (varchar) – ID of the admin performing the action.

3. date_time (datetime) – When the action occurred.

4. status (varchar) – Status or result of the action.

5. action (varchar) – Description of the action.

Linked to AdminDetails via admin_id.

4. 10. AdminDetails

Stores admin profile information.

Columns:

1. admin_id (PK)(varchar) – Unique identifier.

2. admin_name (varchar) – Admin's name.

3. status (varchar) – Current account status.

Connected to AdminLog.

4. 11. Bookings

Core transactional table that logs all service bookings made by customers.

Columns:

1. booking_id (PK)(varchar) – Unique booking identifier.

2. provider_id(FK) (varchar) – ID of the service provider assigned.

3. customer_id (FK)(varchar) – ID of the customer making the booking.

4. location (varchar) – Location where service is needed.

5. date_time (datetime) – Date and time of the booking.

6. amount_paid (float) – Total price paid for the service.

7. type_of_service (varchar) – Type/category of the service.

8. status (varchar) – Booking status (e.g., completed, pending, cancelled).

Connected to Customer, ServiceProvider, ServicesTypes.

Linked to Ratings through booking_id.

4. 12. ServicesTypes

Holds the list of available service categories on the platform.

Columns:

1. service_id (PK)(varchar) – Unique ID for the service type.

2. service_name (varchar) – Human-readable name of the service.

3. no_of_providers (bigint) – Number of providers offering this service.

Connected to ServiceProvider and Bookings via type_of_service.

4. 13. Ratings

Stores customer and provider ratings post-service completion.

Columns:

1. rating_id(PK) (varchar) – Unique identifier for the rating.

2. booking_id (FK)(varchar) – Associated booking record.

3. rating_by_customer (float) – Customer's rating to provider.

4. rating_by_provider (float) – Provider's rating to customer.

Linked to Bookings via booking_id.

Indirectly connected to both Customer and ServiceProvider.

5.USER INTERFACE IMPLEMENTATION:

5.1. Login/Sign-Up Page

Purpose: The Login/Sign-Up page serves as the primary authentication gateway. It enables existing users to access their accounts and directs new users to the appropriate registration flow.

Layout and Components:

- **Brand Logo (Top Right):**
  - **Component:** Image.
  - **Description:** Displays the "GO LOCAL" logo, reinforcing brand identity. May also function as a link to the homepage.
- **Page Header (Top Left):**
  - **Component:** Text Heading.
  - **Content:** "Login/Sign-Up".
  - **Description:** Clearly identifies the page's dual function.
- **New User Call-to-Action (Below Header):**
  - **Component:** Text Link.
  - **Content:** "Not a member? Create one."
  - **Description:** Navigational link directing prospective users to the registration page/flow.
- **Authentication Form Area (Central, overlaying a background image):**
  - **Background Image:**
    - **Component:** Image.
    - **Description:** Provides visual context; content may vary.
  - **Form Mode Tabs:**
    - **Component:** Tab Navigation (e.g., buttons or styled links).
    - **Options:** "Sign-up", "Sign-in".
    - **Description:** Allows users to switch between the sign-up and sign-in forms. The "Sign-in" tab is depicted as active by default.
  - **Username Input Field:**
    - **Component:** Text Input.
    - **Label:** "Username".
    - **Data Type:** String.
    - **Validation:** Required; specific format TBD (e.g., alphanumeric, email).
  - **Password Input Field:**
    - **Component:** Password Input.
    - **Label:** "Password".
    - **Data Type:** String (masked).
    - **Validation:** Required; minimum length, complexity rules TBD.
  - **Login Button:**

        - ♣ **Component:** Button (Submit).
        - ♣ **Label:** "Login".
        - ♣ **Action:** Submits the sign-in form credentials for authentication.
    - **Forgot Password Link:**
        - ♣ **Component:** Text Link.
        - ♣ **Content:** "Forgot Password?".
        - ♣ **Action:** Navigates the user to the password recovery/reset flow.

5. 2. Sign-Up Page - Register as a Helper

Purpose: This page facilitates the registration process for individuals intending to offer services ("Helpers") on the "Go Local" platform. It captures all necessary data for creating a Helper profile.

Layout and Components:

- **Brand Logo (Top Right):**
    - **Component:** Image.
    - **Description:** Displays the "GO LOCAL" logo.
- **Page Header (Top Left):**
    - **Component:** Text Heading.
    - **Content:** "Sign-Up".
- **Form Context Sub-Header:**
    - **Component:** Text Sub-heading.
    - **Content:** "Register as a Helper".
- **User Type Toggle:**
    - **Component:** Switch/Toggle Control.
    - **Options:** "Helper", "Customer".
    - **Default:** "Helper" selected.
    - **Description:** Allows users to switch registration type if they landed on the wrong form.
- **Illustrative Graphic (Left Side):**
    - **Component:** Image/Illustration.
    - **Description:** A visual element relevant to the "Helper" role (e.g., a handyman).
- **Registration Form Fields (Right Side, two-column layout):**
    - **Name Input:** Text Input, Label: "Name".
    - **Mobile No. Input:** Telephone Input, Label: "Mobile No.". Validation for phone number format.
    - **E-mail Id Input:** Email Input, Label: "E-mail Id". Validation for email format; uniqueness check required.
    - **Occupation Dropdown:** Select Input, Label: "Occupation". Default: "---Select---". Options to be populated from a predefined list.

    - **Location Input:** Text Input, Label: "Location". May incorporate geolocation or autocomplete features.
    - **Photograph Upload:** File Input, Label: "Photograph", Button Text: "Upload". For profile picture. Validation for file type and size.
    - **Username Input:** Text Input, Label: "Username". Validation for format, length, and uniqueness.
    - **Describe You Textarea:** Textarea Input, Label: "Describe you". For a brief helper bio/description. Character limit may apply.
    - **Password Input:** Password Input, Label: "Password". Validation for strength (length, complexity).
    - **Confirm Password Input:** Password Input, Label: "Confirm Password". Must match the "Password" field. Includes a visibility toggle icon.
- **Footer Section (Bottom):**
    - **Description:** Consistent footer element as seen on other pages.

5.3. Sign-Up Page - Register as a Customer

Purpose: This page enables users to register as "Customers" on the "Go Local" platform, i.e., those seeking services. It collects the data required for customer account creation.

Layout and Components:

- **Brand Logo (Top Right):**
    - **Component:** Image.
    - **Description:** Displays the "GO LOCAL" logo.
- **Page Header (Top Left):**
    - **Component:** Text Heading.
    - **Content:** "Sign-Up".
- **Form Context Sub-Header:**
    - **Component:** Text Sub-heading.
    - **Content:** "Register as a Customer".
- **User Type Toggle:**
    - **Component:** Switch/Toggle Control.
    - **Options:** "Helper", "Customer".
    - **Default:** "Customer" selected.
- **Registration Form Fields (Centered, overlaying a background image):**
    - **Background Image:**
        - **Component:** Image.
        - **Description:** Provides visual context; content may vary (depicts people).
    - **Name Input:** Text Input, Label: "Name".
    - **Mobile No. Input:** Telephone Input, Label: "Mobile No.". Validation for phone number format.

    - **E-mail Id Input:** Email Input, Label: "E-mail Id". Validation for email format; uniqueness check required.
    - **Location Input:** Text Input, Label: "Location".
    - **Photograph Upload:** File Input, Label: "Photograph", Button Text: "Upload". For profile picture (consider if mandatory for customers). Validation for file type and size.
    - **Username Input:** Text Input, Label: "Username". Validation for format, length, and uniqueness.
    - **Password Input:** Password Input, Label: "Password". Validation for strength. Includes a visibility toggle icon.
    - **Confirm Password Input:** Password Input, Label: "Confirm Password". Must match the "Password" field. Includes a visibility toggle icon.
- **Register Button:**
    - **Component:** Button (Submit).
    - **Label:** "Register".
    - **Action:** Submits the customer registration form data.
- **Footer Section (Bottom):**
    - **Description:** Consistent footer element.

5. 4. Login Success Page

Purpose: The Login Success page provides immediate positive feedback to the user upon successful authentication. It confirms the login action and facilitates navigation back to the application's main interface.

Layout and Components:

- **Brand Logo (Top Right):**
    - **Component:** Image.
    - **Description:** Displays the "GO LOCAL" logo.
- **Success Message Area (Centered):**
    - **Primary Headline:** Text Heading, Content: "Congratulation User". (Note: Consider dynamic user name if available).
    - **Secondary Message:** Text Sub-heading, Content: "You Have Successfully Logged In !!!".
- **Success Graphic (Below Message):**
    - **Component:** Image/Icon.
    - **Description:** A visual cue reinforcing successful login (e.g., a password field with a checkmark).
- **Redirection Prompt:**
    - **Component:** Text.
    - **Content:** "Redirect Back To The Home Page".
- **Home Page Button:**
    - **Component:** Button.

    - **Label:** "Home Page".
    - **Action:** Manually navigates the user to the application's home page or dashboard.
    - **Note:** Consider implementing automatic redirection after a short delay (e.g., 3-5 seconds) for improved user experience.
- **Footer Section (Bottom):**
    - **Description:** Consistent footer element.

5. 5. Helper Profile Page

Purpose: The Helper Profile page displays detailed information about a specific service provider ("Helper"). This page is intended for customers to review a helper's credentials, services, ratings, and contact information.

Layout and Components:

- **Top Navigation Bar:**
    - **Platform Name (Left):** Text, Content: "Go Local".
    - **Home Button (Right):** Button/Link, Label: "HOME". Action: Navigates to the main user dashboard or homepage.
    - **Brand Logo (Far Right):** Image, Description: Displays the "GO LOCAL" logo.
- **Main Profile Section (Two-pane layout):**
    - **Left Pane (Helper Identity & Summary):**
        - **Profile Picture:** Image, Displays the helper's uploaded photo.
        - **Helper Brand/Alias:** Text Heading, Content: "JK_Music" (example). sUser-defined display name or brand.
        - **Experience:** Text, Content: "Experience: 3yrs" (example). Key metric from the helper's profile.
        - **Work/Activity Indicator:** Visual Element (e.g., progress bar), Label: "work". Purpose needs clarification (e.g., profile completeness, recent activity).
    - **Right Pane (Detailed Information & Actions):**
        - **Helper Full Name:** Text Heading, Content: "John Kate" (example).
        - **Profession/Title:** Text Sub-heading, Content: "Personal Music Tutor" (example). Helper's declared specialization.
        - **Location:** Text, Content: "Chennai" (example). Helper's primary service area or base location.
        - **Average Ratings:**
            - **Component:** Rating Display (e.g., stars).
            - **Description:** Visually represents the aggregated user ratings for the helper.
        - **Profile Information Tabs:**
            - **Component:** Tab Navigation.

        - ♣ **Options:** "About", "CONTACT INFORMATION" (example).
        - ♣ **Description:** Allows users to switch between different sections of the helper's profile. "CONTACT INFORMATION" is depicted as the active tab.
    - ♣ **Contact Information Section (Visible when "CONTACT INFORMATION" tab is active):**
        - ♣ **Phone:** Text Label: "Phone:", Value: Helper's contact number (e.g., "+918934567827").
        - ♣ **Address:** Text Label: "Address:", Value: Helper's physical or service address (e.g., "Siruseri, Chennai-630032").
        - ♣ **E-mail:** Text Label: "E-mail:", Value: Helper's contact email address (e.g., "johnkate@gmail.com").
    - ♣ **About Section (Content visible when "About" tab is active - not shown in current state):**
        - ♣ **Description:** This section would contain a more detailed textual description provided by the helper about their services, skills, and background.

5. 6.Top Services Listing Page (Profession: Painter - Chennai)

This page is a service listing platform interface, showing top painting services available in Chennai, under the "Painter" profession. It features a repeated service card design for one provider and a footer for navigation.

Layout Overview:

1. **Header Section**
    - o Title: “Here’s What Top Services We Found For…”
    - o Subtext: The chosen City and profession, e.g., City Chennai and Profession Painter
2. **Service Cards Section**
    - o Service Name: e.g., Asray Painting Services
    - o Short Description: About the service provider, e.g., "Hey I am Asray. A painter who can make Painting you can’t belie…" (truncated)
    - o Button: Know More
3. **How It Works Section**
    - o User selects city and profession: (currently: Chennai, Painter).
    - o Matching services are fetched and displayed.
    - o User reviews listings: (each card provides an overview and a "Know More" button).
    - o User clicks “Know More” for more details: (navigates to detailed profile or service page).

4. **Fields and Elements Description**
   - City & Profession: Indicates user's selected city (Chennai) and profession (Painter).
   - Service Name: Business name ("Asray Painting Services").
   - Service Description: Brief intro of the service provider.
   - Know More Button: Likely redirects to a detailed service page/profile.
   - Image Placeholder: Gray square image (currently blank). Intended for service logo/image.
5. **Footer Section**
   - Site name
   - Social Media Icons: Placeholder icons (4 present)
   - Links

5. 7.About Us Page – Go Local

This is an “About Us” page for the “Go Local” service platform. It provides details about the team, project background, and includes a contact form for user inquiries, along with branding and footer navigation.

Layout Overview:

1. **Header Section**
   - Title: “About Us”
   - Subheading: Team Info
   - Paragraph: Team information.
2. **Project Description Section**
   - Heading: Project Description
   - A description of what our platform does and what services we actually provide.
3. **Contact Us Section**
   - Fields:
     - ♣ First name
     - ♣ Last name
     - ♣ Email address
     - ♣ Message box
     - ♣ Submit Button

4. **Footer Section**
   - Site name
   - Social Media Icons: 4 icons visible (LinkedIn, Facebook, Twitter, Instagram-style placeholders)
   - Navigation Columns (4 topics): Each with repeated "Page" links

How It Works Section

- Visitors read about the team/project: under the Team Info and Project Description sections.
- Users can contact the team via a simple form submission.
- Navigation and external links are available in the footer.

Fields and Elements Description

- **About Us Title:** Main heading introducing the page.
- **Team Info Text:** Represents description of the team.
- **Project Description:** C for project overview.
- **Contact Form Fields:**
    - **First Name:** Text input for user's first name.
    - **Last Name:** Text input for user's last name.
    - **Email Address:** Input for contacting user.
    - **Your Message:** Text area for user queries or feedback.
    - **Submit Button:** Sends form input (functionality presumed, not shown).
- **Footer Site Name:** website branding.
- **Social Media Icons:** Links to platforms (design placeholders).
- **Navigation Links**

5. 8.Details about the service Provider – Go Local

This page introduces selected Service provider under the “Go Local” platform. It includes a brief personal bio, a summary of completed work, rating info, and contact information.

Layout Overview:

1. **Header Section**
    - **Title:** “Service Title” (bold, top left)
    - **Go Local Logo:** Positioned top right

2. **Profile Image**
    - **Banner-style image of the painter**
3. **About Myself Section**
    - **Heading:** About Myself
    - **Text:**
        - ♣ Self-description
        - ♣ No. of Services provided till date
        - ♣ My Ratings: Meant to include stars
4. **Contact Me Section**
    - **Phone Number:** Phone number of the service provider
5. **Footer Section**

- Site name
- **Social Media Icons**
- **Navigation Columns**

How It Works Section

- Visitors read about the painter: under the About Myself section.
- Visual context is provided with a profile-style image.
- Users can contact the painter via phone number provided.
- Navigation and external links are available in the footer.

Fields and Elements Description

- **Title:** Displays the service provider's name and role.
- **Profile Image:** Painter's photo adds trust and personal branding.
- **About Myself:** Introduces the painter and their services.
- **Completed Work:** Shows professional credibility.
- **Ratings:** To show user/customer feedback.
- **Phone Number:** Direct contact method.
- **Footer Site Name:** platform branding.
- **Social Media Icons:** Represents painter/platform presence on social media.
- **Navigation Links.**

5. 9.User Profile Page – Go Local

This is a user profile page for someone registered on the Go Local platform. It displays the user's basic details, contact information, and booking history for various services like electrician, painter, and babysitter.

Layout Overview:

1. **Header Section**
   - **Title:** "Go Local" (centred)
   - **Go Local Logo:** Top-right corner
   - **Home Button**
2. **User Profile Information**
   - **Profile Picture of the user**
   - **Username of the user**
   - **Location of the user**
   - **Display Name:** Username
3. **Contact Information**
   - **Phone:** Phone number of the user
   - **Address: Full address of the user**
   - **E-mail:** Email of the user

4. **Bookings History**
    - **Booking Cards:** Listed vertically with icons and “View” buttons

How It Works Section

- Visitors read about the user: under the User Profile Information section.
- Visual context is provided with a profile-style image.
- Users can contact the user via phone number or email provided.
- Navigation and external links are available in the footer.

Fields and Elements Description

- **Profile Picture:** Displays the user's photo for identification.
- **User Info:** Includes username, city, and display name.
- **Contact Details:** Phone, address, and email.
- **Bookings:** Shows services the user has booked with dates and status.
- **“View” Buttons:** Presumably open booking details or history.
- **Footer Site Name:** Placeholder for website or platform branding.
- **Social Media Icons:** Represents user/platform presence on social media.
- **Navigation links**

5. 10. Home Page – Go Local Platform

This is the homepage of the "Go Local" service platform, designed to offer users access to local service providers like electricians, babysitters, tutors, and more. The layout showcases featured professionals, testimonials, and suggested services for further exploration.

Layout Overview:

1. **Header Section**
    - Platform Name: “Go Local” (top-centre)
    - Navigation Menu: Includes links like “Home,” “Login,” “About,” or “Services”
2. **Hero Image Section**
    - Large image of professionals at work: (technician and assistant)
    - Short description or promotional line: seen below the image (represents the platform's mission)
3. **Our Top Performers Section**
    - Each includes an image, title, and brief text: representing verified and top-rated service providers
4. **Hear From Our Customers Section**
    - Testimonial section with customer feedback

- Includes a star rating below a short review
- Accompanied by an image of customers interacting with service staff

5. **Explore More Section**
   - Additional service cards:
   - Each card includes a relevant image, description, and possibly a “Book Now” or “Learn More” button
6. **Footer Section**
   - Contains “Contact Us” and a simple site map/grid of links
   - Multiple Navigation links

How It Works Section

- Visitors read about the platform's mission: under the Hero Image Section.
- Visual context is provided with images of professionals at work.
- Users can explore top-rated service providers and customer testimonials.
- Navigation and external links are available in the footer.

Fields and Elements Description

- Hero Banner: Promotes the platform visually with a thematic image.
- Top Performers: Highlights trusted service providers.
- Customer Testimonials: Adds trust with social proof (5-star feedback).
- Explore More: Suggests additional services like coaching, babysitting.
- Footer: Offers quick access to city/state-wise service lists.

5. 11.Home Page (Updated Layout after logging in) – Go Local

This is an enhanced version of the Go Local homepage, with access to user’s own profile. It retains core functionality — showcasing service providers, customer reviews, and additional service offerings — in a user-friendly manner.

Additional feature:

- **User Profile Icon:** The addition of the user profile icon in the top-left corner of the header section. Clicking this icon navigates the user to the User Profile Page.

  Reference to Previous Functionalities:

- The functionalities described in the original Home Page layout remain unchanged. For detailed descriptions of these functionalities, please refer to the original Home Page section.

6.BACKEND IMPLEMENTATION:

6.1.MainController:

```
@RestController

Public class MainController{

}
```

6. 2.UserAuthController:

```
@RestController

@RequestMapping("/api/userauth")

Public class UserAuthController{

}
```

6. 3. ServiceProvider Controller:

```
@RestController

@RequestMapping("/api/providers")

Public class ServiceProvider{

}
```

6. 4.Customer Controller:

```
@RestController

@RequestMapping("/api/customer")

Public class Customer{

}
```

6. 5.Bookings Controller:

```
@RestController

@RequestMapping("/api/bookings")
```

Public class Bookings{}

6. 6.Ratings Controller:

@RestController

@RequestMapping("/api/ratings")

Public class Ratings{

}

6. 7.Search History Controller:

@RestController

@RequestMapping("/api/search")

Public class SearchHistory{

}

6. 8.Support Controller:

@RestController

@RequestMapping("/api/support")

Public class Support{

}

6. 9.Admin Controller:

@RestController

@RequestMapping("/api/admin")

Public class Admin{

}

6. 10.Login and Logout Log Controller:

@RestController

@RequestMapping("/api/log")

Public class Log{

}

6. 11.Service Types:

@RestController

@RequestMapping("/api/servicetypes")

Public class ServiceTypes{

}

7.VALIDATION:

7.1. Landing Page (Before Login)

- Public access, no inputs.
- Ensure:
  - o Static data like banners, descriptions, or team photos are from safe sources
  - o No dynamic parameters are reflected without encoding

7. 2. Register as a Helper Page

Frontend (React):

- Name:
  - o Required
  - o Only letters and spaces
  - o Minimum 2 characters
- Mobile Number:
  - o Required
  - o Exactly 10 digits
- Email:
  - o Required
  - o Must be a valid email format
- Occupation:
  - o Required
  - o Selected from dropdown
- Location:
  - o Required
  - o Alphanumeric, spaces/hyphens allowed
- Photograph:
  - o Required
  - o Formats allowed: JPG, JPEG, PNG

    - Max size: 2MB
- Description:
    - Optional
    - If entered: 10 to 300 characters
- Username:
    - Required
    - Minimum 3 characters
    - Alphanumeric and underscores only
- Password:
    - Required
    - Minimum 8 characters
    - Must include:
    - 1 uppercase, 1 lowercase, 1 digit, 1 special char
- Confirm Password:
    - Required
    - Must match Password
- Validation Approach:
    - Reactive Forms with React validators
    - Real-time validation and error messages
    - Image preview and validation before upload

Backend (Spring Boot):

- Field Validations:
    - @NotBlank, @Pattern, @Size, @Email used
    - Regex for mobile and password complexity
- Photograph:
    - Validated for type and size in controller
- Confirm Password:
    - Checked manually to match password

8. API IMPLEMENTATION:

Module	**Endpoint**	**Method**	**Purpose**
User Authentication	/api/userauth/register	POST	Register a new user
User Authentication	/api/userauth/login	POST	User login
User Authentication	/api/userauth/{id}	GET	Get user details by ID
User Authentication	/api/userauth/{id}	DELETE	Delete user
Service Providers	/api/providers/	GET	Get all service providers
Service Providers	/api/providers/	POST	Add new provider
Service Providers	/api/providers/{id}	GET	Get provider by ID
Service Providers	/api/providers/{id}	PUT	Update provider
Customers	/api/customers/	GET	Get all customers
Customers	/api/customers/	POST	Add new customer

Customers	/api/customers/{id}	GET	Get customer by ID
Customers	/api/customers/{id}	PUT	Update customer
Bookings	/api/bookings/	GET	Get all bookings
Bookings	/api/bookings/	POST	Create new booking
Bookings	/api/bookings/{id}	GET	Get booking by ID
Bookings	/api/bookings/{id}	PUT	Update booking
Ratings	/api/ratings/	POST	Submit a rating
Ratings	/api/ratings/booking/{bookingId}	GET	Get rating for a booking
Service Types	/api/services/	GET	Get all service types
Service Types	/api/services/	POST	Add a new service type
Search History	/api/search/customer/{customerId}	GET	Get search history for a customer
Search History	/api/search/location	GET	Get all the service provider in the location
Search History	/api/search/location/{customerId}	GET	Get the specific service provider in the location and customer if exists
Support	/api/support/	POST	Raise a new support query
Support	/api/support/user/{userId}	GET	View queries raised by a user
Admin	/api/admin/login	POST	Admin login
Admin	/api/admin/logs	GET	Get admin logs
Admin	/api/admin/logs	POST	Add a new log entry
Log	/api/loginlog/	POST	Record login event
Log	/api/loginlog/user/{userId}	GET	Get login history of a user
Logout Logs	/api/logoutlog/	POST	Record logout event

9. Services

This section outlines the core service layer classes responsible for the business logic and data manipulation within the "Go Local" platform.

Service Layer Classes:

- **AuthService**:
  - **Responsibilities:** Handles user authentication processes including login (credential verification) and registration (new user account creation). Manages password hashing for secure storage and password reset functionalities. Handles the generation, validation, and management of JSON Web Tokens (JWT) or similar session management tokens for authenticated users.
  - **Interacts With:** UserService, Database (User Credentials).

- **Exposed To:** Authentication-related API Endpoints (e.g., /login, /register, /forgot-password).

- **UserService**:
    - **Responsibilities:** Manages user profiles for both "Helpers" and "Customers". This includes creating, retrieving, updating, and deleting user profile information (e.g., name, contact details, location, profile picture, description/bio for Helpers, booking history for Customers). Handles user account settings and preferences.
    - **Interacts With:** AuthService, BookingService, ReviewService, Database (User Profiles, Helper Profiles, Customer Profiles).
    - **Exposed To:** User profile management API Endpoints (e.g., /users/{userId}, /helpers/{helperId}).
- **HelperService (formerly VenueService/PlayerService combined for this context):**
    - **Responsibilities:** Manages the listing, searching, and detailed profiles of "Helpers" (service providers). This includes handling helper-specific information such as occupation/profession, experience, skills, "About Me" sections, and potentially service areas. Facilitates searching and filtering of helpers based on criteria like profession, location, and ratings.
    - **Interacts With:** UserService, ReviewService, BookingService, Database (Helper Profiles, Occupations/Professions).
    - **Exposed To:** Helper listing and search API Endpoints (e.g., /helpers, /search/helpers).
- **BookingService**:
    - **Responsibilities:** Manages the service booking process. This includes checking helper availability (if applicable, though not explicitly shown for all services), creating new bookings, retrieving booking history for customers and potentially helpers. May handle booking status updates (e.g., confirmed, completed, cancelled). Pricing logic, if dynamic, would reside here. Implements logic to prevent double-booking if scheduling is involved.
    - **Interacts With:** UserService, HelperService, NotificationService, Database (Bookings).
    - **Exposed To:** Booking management API Endpoints (e.g., /bookings, /users/{userId}/bookings).
- **ReviewService**:
    - **Responsibilities:** Manages user-submitted reviews and ratings for "Helpers" based on completed bookings. Ensures that a user can submit a review for a service they have utilized. Calculates and updates average ratings for helpers.
    - **Interacts With:** UserService, HelperService, BookingService, Database (Reviews, Ratings).
    - **Exposed To:** Review submission and retrieval API Endpoints (e.g., /helpers/{helperId}/reviews, /bookings/{bookingId}/review).
- **NotificationService**:

  - **Responsibilities:** Handles the sending of notifications to users. This could include email or SMS notifications for events such as successful registration, booking confirmations, booking reminders, review submissions, or responses to contact form inquiries. (Initially, this might be a stub or basic email implementation).
  - **Interacts With:** AuthService, BookingService, ReviewService, ContactService, External Email/SMS gateways.
  - **Exposed To:** Internal service calls triggered by events.
- **ContactService (New, based on About Us page):**
  - **Responsibilities:** Manages inquiries submitted through the "Contact Us" form on the "About Us" page. Stores contact messages and potentially routes them to platform administrators.
  - **Interacts With:** NotificationService (for notifying admins of new messages), Database (Contact Submissions).
  - **Exposed To:** Contact form submission API Endpoint (e.g., /contact-us).
- **ValidationService**:
  - **Responsibilities:** Provides reusable custom validation logic that can be utilized by other services or at the controller/API endpoint level. This includes validating input data formats (e.g., email, phone number), data integrity, and adherence to business rules not covered by basic model validation.
  - **Interacts With:** All other services requiring complex or shared validation logic.
  - **Exposed To:** Internal service calls.
- **ContentManagementService (Conceptual, for static/semi-static content):**
  - **Responsibilities:** Manages content for pages like "About Us" (Team Info, Project Description), "Homepage" (Top Performers, Customer Testimonials). This might involve retrieving content from a database or a simpler content store.
  - **Interacts With:** Database or file system (for content storage).
  - **Exposed To:** API endpoints serving website content.