Core Features

1. User Management

- **User Registration**: Allow users (customers and artisans) to register on the platform.

Description

The user registration feature enables new users, including customers and artisans, to create an account on the platform through email/password or third-party authentication (Google or Facebook). It collects essential user information, validates the input, and securely stores the details in the database.

Actors

1. **Customer (Client):** Registers to search for and book services.
2. **Artisan (Handyman):** Registers to offer and manage their services.
3. **System:** Facilitates the registration process, validates inputs, and stores user data in the database.
4. **Third-Party Providers (Google, Facebook):** Authenticate users and provide profile data for registration.

Preconditions

1. The user must have access to the registration page via a compatible browser.
2. The system should be online and capable of handling user requests.
3. Required fields (e.g., name, email, password) must be clearly labeled.
4. The system must be configured with Google and Facebook API credentials.

Postconditions

1. User account information is successfully stored in the database.
2. The user receives a confirmation of successful registration.
3. The system creates user roles (customer or artisan) based on the registration type.

Steps

- The user navigates to the registration page.
- The user selects their role (Customer or Artisan).

The user chooses a registration method:

- **Manual Registration:**
  - Fills in required fields (Name, Email Address, Password, Phone Number, etc.).
  - Submits the registration form.
  - The system validates and processes the data (input validation, password encryption).
- **Third-Party Registration:**
  - The user clicks "Sign Up with Google" or "Sign Up with Facebook."
  - The system redirects the user to the selected provider's OAuth login page.
  - After successful login, the provider returns an access token and profile data (e.g., name, email).
  - The system checks if the email already exists:
    - If not, it creates a new user account using the retrieved profile data.
    - If the email exists, it links the third-party account to the existing user profile.

If registration is successful:

- The system stores the user details in the database (for manual or third-party methods).
- Sends a confirmation email (manual registration only).

The user is redirected to the login page.

Third-Party Integration Notes

1. **Google OAuth Setup:**
    - Obtain API credentials (Client ID and Secret) from the Google Cloud Console.
    - Configure redirect URIs for the application.
    - Use a library like `passport-google-oauth20` (Node.js) or `google-auth-library` (Python).
2. **Facebook OAuth Setup:**
    - Create a Facebook App in the Facebook Developer Console to obtain App ID and Secret.
    - Configure redirect URIs.
    - Use a library like `passport-facebook` (Node.js) or `facebook-sdk` (Python).
3. **Security Considerations:**
    - Use HTTPS to secure OAuth communication.
    - Validate tokens received from third-party providers.
    - Store tokens securely in the database or session.

- **User Login**: Provide authentication for users with secure login (JWT or session-based).

Description

The user login feature allows registered users (customers and artisans) to securely access the platform through email/password or third-party authentication (Google or Facebook). The system ensures authentication, creates JWTs, and redirects users to their respective dashboards.

Actors

1. **Customer (Client):** Logs in to book and manage services.
2. **Artisan (Handyman):** Logs in to manage their service listings and bookings.
3. **System:** Handles authentication, and JWT generation, and grants access to the user dashboard.
4. **Third-Party Providers (Google, Facebook):** Authenticate users and provide profile data for login.

Preconditions

1. The user must have a registered account.
2. The login page must be accessible via a compatible browser.
3. The system must be connected to the user database and third-party APIs.

Postconditions

1. The user is authenticated and redirected to their dashboard (Customer or Artisan).
2. A JWT is created and securely stored.
3. The system logs the user's activity for tracking purposes.

Steps:

1. The user navigates to the login page.
2. The user selects a login method:
    - **Manual Login:**
        - Enters their email and password.
        - Submits the login form.
        - The system retrieves the user data and validates the credentials:
            - Checks if the email exists and verifies the password.
    - **Third-Party Login:**
        - The user clicks "Login with Google" or "Login with Facebook."
        - The system redirects the user to the selected provider's OAuth login page.
        - Upon successful login, the provider returns an access token and user profile data (e.g., email).
        - The system checks if the email exists in the database:
            - If it exists, the system logs in the user.
            - If it doesn't, the system prompts the user to complete registration.
3. If login is successful:
    - For JWT-based login:
        - The system generates a JWT containing the user's details.

    - The system logs the login time and user ID for tracking.
4. The user is redirected to their dashboard.

- **Password Recovery**: Implement password reset functionality using email recovery links.

Description

The password recovery feature allows users to reset their forgotten passwords securely. By submitting their registered email address, users receive a recovery link that redirects them to a password reset page. This ensures users can regain access to their accounts without compromising security.

Actors

1. **Customer (Client):** Requests a password reset to regain access to their account.
2. Artisan (Handyman): Uses the feature to reset a forgotten password.
3. System**:** Sends a recovery email, validates the recovery token, and facilitates the password reset process.

Preconditions

1. The user must have an active account with a registered email address.
2. The system's email service must be configured and operational.
3. The user should have access to the email address associated with their account.

Postconditions

1. The user successfully resets their password.
2. The system updates the user's password in the database (hashed for security).
3. The recovery token is invalidated after use.

Steps

1. The user navigates to the "Forgot Password" page.
2. The user enters their registered email address into the recovery form and submits it.
3. The system verifies if the email exists in the database:
    - If the email does not exist, the system displays an error message (e.g., "No account is associated with this email").
    - If the email exists, the system generates a unique, time-limited recovery token.
4. The system sends a recovery email to the user with a secure link containing the token.
    - The link format: `https://platform.com/reset-password/<token>`
5. The user receives the email and clicks the recovery link.
6. The system validates the recovery token:
    - Checks if the token is valid and has not expired.
    - If invalid or expired, the system displays an error message (e.g., "This link is no longer valid").
7. If the token is valid, the system redirects the user to the password reset page.
8. The user enters a new password and confirms it.
9. The system validates the new password:

    - Ensures the password meets security criteria (e.g., minimum length, special characters).
    - Ensures the "new password" and "confirm password" fields match.
10. If validation succeeds, the system:
    - Hashes the new password.
    - Updates the user's password in the database.
    - Invalidates the recovery token to prevent reuse.
11. The system displays a success message and redirects the user to the login page.
12. The user logs in using their new password.

- **Profile Management**: Enable users to manage their profiles, including updating personal details and adding a profile picture.

Description

The profile management feature allows users (customers and artisans) to view and update their personal information, including their name, email, phone number, and other details. Users can also upload or update their profile picture. This ensures that user profiles remain accurate and up to date.

Actors

1. **Customer (Client):** Updates their profile details to manage contact information for service bookings.

2. Artisan (Handyman)**:** Updates their profile to present accurate service-related information to potential clients.
3. System: Handles updates to the database and validates inputs, ensuring data integrity.

Preconditions

1. The user must be logged into their account.
2. The system must have access to the database to retrieve and update user information.
3. The profile update page or section must be accessible through the dashboard.

Postconditions

1. The updated user details are stored securely in the database.
2. The profile picture (if uploaded) is stored and displayed on the user's profile.
3. The user receives confirmation of successful updates.

Steps

1. The user navigates to the "Profile" section from their dashboard.
2. The system retrieves the current profile details and displays them in editable fields.
3. The user updates their desired information, which may include:
    - Name
    - Phone Number
    - Email Address
    - Address (if applicable)

4. If the user wants to update their profile picture:
    - The user clicks the "Upload Picture" button.
    - The system opens a file upload dialog.
    - The user selects an image file (e.g., JPG, PNG).
    - The system validates the image file type and size.
5. The user submits the updated profile details.
6. The system validates the input:
    - Ensures all required fields are completed.
    - Validates email format and phone number.
7. If validation fails:
    - The system displays appropriate error messages (e.g., "Invalid email format").
8. If validation succeeds:
    - The system updates the user information in the database.
    - If a new profile picture was uploaded, the system stores the image in the designated storage location and updates the database with the image path.
9. The system displays a success message confirming the profile update.
10. The user’s updated profile is displayed.

---

2. Artisan Management

-
- **Service Listing**: Artisans can list their services, including pricing, availability, and description.

Description

The service listing feature allows artisans to create and manage detailed listings for the services they provide. Each listing includes a service name, description, pricing, and availability. This ensures that clients have clear and accessible information to book the required services.

Actors

1. **Artisan (Handyman):** Creates, updates, and manages their service listings.
2. **Customer (Client):** Views the service listings to book services.
3. **System:** Validates, stores, and displays service listings to potential clients.

Preconditions

1. The artisan must be logged into their account.
2. The system must provide an accessible interface for managing service listings.
3. The database must be available to store the details of service listings.

Postconditions

1. The artisan’s service listing is saved in the database.
2. The listing becomes visible to clients in the service search and booking sections.
3. The artisan receives confirmation of the successful creation or update of their listing.

Steps

1. **Creating a New Service Listing:**
    - The artisan navigates to the "Service Listings" section of their dashboard.
    - The artisan clicks the "Add New Service" button.
    - The system displays a form with the following fields:
        - **Service Name:** A short, clear name (e.g., "AC Repair").
        - **Description:** Detailed information about the service (e.g., "Repairing and maintaining air conditioners").
        - **Pricing:** The cost of the service (e.g., fixed price or hourly rate).
        - **Availability:** Days and times the service is offered (e.g., "Mon-Fri, 9 AM - 6 PM").
    - The artisan fills out the form and submits it.
    - The system validates the input:
        - Ensures all required fields are completed.
        - Validates the pricing format (e.g., numeric values).
    - If validation fails, the system displays error messages (e.g., "Pricing must be a valid number").
    - If validation succeeds:
        - The system saves the service details in the database.
        - The artisan receives a confirmation message (e.g., "Your service has been listed successfully").
2. **Updating an Existing Service Listing:**
    - The artisan navigates to their existing listings in the "Service Listings" section.
    - The artisan selects a service to update and clicks "Edit."
    - The system displays the current details in an editable form.
    - The artisan modifies the desired fields and submits the changes.
    - The system validates and updates the information in the database.

    - The artisan receives a confirmation message.
3. **Deleting a Service Listing:**
    - The artisan navigates to the "Service Listings" section.
    - The artisan selects a service to delete and clicks "Delete."
    - The system displays a confirmation prompt (e.g., "Are you sure you want to delete this service?").
    - Upon confirmation, the system removes the service from the database and updates the client-facing service listings.
    - The artisan receives a success message (e.g., "Service has been deleted").

- **Ratings and Reviews**: Customers can rate artisans and leave reviews after service completion.

Description

The ratings and reviews feature allows customers to provide feedback on the services they have received from artisans. After the service is completed, customers can rate the service (typically with a numeric scale, such as 1 to 5 stars) and leave a written review. This system helps maintain service quality, fosters trust between clients and artisans, and provides valuable insights for future clients.

Actors

1. **Customer (Client):** Rates and leaves reviews for the artisan after service completion.
2. **Artisan (Handyman):** Receives and views ratings and reviews provided by customers.

3. **System:** Manages and stores ratings and reviews, ensuring they are displayed correctly and securely.

Preconditions

1. The service must be completed (i.e., the booking should be marked as "Completed").
2. The customer must be logged into their account.
3. The artisan must have completed the service and marked it as completed in the system.
4. The system must ensure that only one review per service per customer is allowed.

Postconditions

1. The rating and review are stored in the database and associated with the relevant artisan and service.
2. The customer's review is displayed on the artisan’s profile page and/or the service page.
3. The artisan receives a notification of a new rating or review.

Steps

1. **Providing a Rating and Review:**
    - After the service is marked as completed, the customer navigates to the "Completed Services" section of the platform.
    - The customer selects the completed service and clicks on the "Rate Service" button.
    - The system displays a rating interface with:

        - **Star Rating:** A 1 to 5 star scale.
        - **Written Review:** A text box for the customer to provide comments.
    - The customer selects a rating and optionally enters a review.
    - The system validates the rating:
        - Ensures that a rating (1-5 stars) is selected.
    - If the customer submits the review:
        - The system stores the rating and review in the database, linking it to the artisan's profile and the specific service.
        - The artisan is notified that a new review has been posted.
        - The system displays a confirmation message to the customer (e.g., "Thank you for your review!").
2. **Viewing Ratings and Reviews:**
    - The artisan can view the feedback on their profile page or on the specific service listing page.
    - The system displays the average rating along with a list of all customer reviews.
    - Reviews are displayed with the customer's name, rating, and review text (ensure that user identities are protected, e.g., only showing the first name or username).

- **Portfolio Uploads**: Artisans can showcase their previous work via uploaded images or descriptions.

Description

The portfolio upload feature allows artisans to showcase their previous work by uploading images or adding descriptions of completed projects. This helps build trust with potential customers by providing visual proof of the artisan's skills and

experience. Artisans can manage and update their portfolios to highlight their best work.

Actors

1. **Artisan (Handyman)**: Uploads and manages images and descriptions of previous work.
2. **Customer (Client):** Views the artisan's portfolio to evaluate their experience and quality of work.
3. **System:** Manages and stores portfolio images and descriptions, ensuring they are displayed correctly and securely.

Preconditions

1. The artisan must be logged into their account.
2. The artisan must have completed at least one project that they want to showcase.
3. The system must provide a section for uploading and managing the portfolio.
4. The system must have storage capability for images and descriptions.

Postconditions

1. The uploaded images and descriptions are saved in the database or storage system.
2. The portfolio is displayed on the artisan's profile page for customers to view.
3. The artisan receives confirmation that the portfolio has been successfully uploaded or updated.

Steps

1. **Navigating to Portfolio Management:**
    - The artisan navigates to the "Portfolio" section of their dashboard.
    - The system displays options to either add a new project or update an existing project.
2. **Uploading Images and Descriptions:**
    - The artisan clicks on the "Add New Project" or "Upload Portfolio" button.
    - The system displays a form with the following fields:
        - **Project Title:** A brief title for the project (e.g., "Bathroom Renovation").
        - **Project Description:** A text box for the artisan to describe the project, including details such as what work was done and any special techniques used.
        - **Upload Images:** An option to upload images showcasing the completed project. The system may allow multiple image uploads (e.g., before-and-after pictures).
    - The artisan uploads one or more images, choosing files from their device.
        - The system may limit the image file size (e.g., 5 MB) and file types (e.g., JPG, PNG).
    - The artisan submits the project details.
3. **Validating the Upload:**
    - The system validates the input:
        - Ensures all required fields are completed.
        - Validates the image file type and size.
    - If validation fails:
        - The system displays an error message (e.g., "File type not supported," "Description is too short").
    - If validation succeeds:
        - The system stores the project details and images in the database or a file storage system.

        - The portfolio is updated and ready to be viewed by customers.
        - The artisan receives a success message (e.g., "Your portfolio has been updated successfully").
4. **Viewing the Portfolio:**
    - The artisan’s portfolio is displayed on their profile page.
    - The portfolio includes the project titles, descriptions, and uploaded images in a clean and organized layout.
    - Customers can browse the portfolio to evaluate the artisan’s skills and the quality of their previous work.
5. **Updating or Deleting Portfolio Entries:**
    - The artisan can return to the portfolio section to update or delete existing portfolio entries.
    - To update:
        - The artisan selects an existing project, clicks "Edit," and modifies the project description or images.
        - The system validates the updated data and stores it.
    - To delete:
        - The artisan selects a project and clicks "Delete."
        - The system confirms the deletion and removes the project from the portfolio.

---

rana

3. Customer Management

Service Requests

Description

The service request feature allows customers to request specific

services by filling out a form detailing their needs, enabling a tailored experience. The system validates the input, securely stores the request, and notifies relevant service providers about the customer's requirements.

Actors

- **Customer (Client):** Submits a request for a specific service.
- **Artisan (Service Provider):** Receives and reviews service requests to decide on offering the service.
- **System:** Facilitates the request submission, validation, storage, and notification process.

Preconditions

- The customer must be logged into their account.
- The system must have service providers registered with available services.
- The service request form must clearly label all required fields (e.g., type of service, preferred date and time).

Postconditions

- The service request is successfully stored in the system.
- Relevant service providers are notified of the request.
- The customer receives confirmation of the submitted request.

Steps

1. The customer logs in and navigates to the service request form.
2. The customer selects the type of service and fills in the required details:

    - Type of service (e.g., plumbing, electrical, cleaning).
    - Preferred date and time.
    - Additional requirements (e.g., specific tools or materials needed).
3. The customer submits the request.
4. The system performs the following actions:
    - Validates the request (e.g., checks mandatory fields).
    - Stores the request securely in the database.
    - Sends a confirmation notification to the customer (email or app notification).
5. The system forwards the request to relevant service providers based on service type and location.

Booking System

Description

The booking system enables customers to schedule appointments with artisans based on their availability. The system manages the scheduling process, validates conflicts, and ensures both parties are notified upon confirmation.

Actors

- **Customer (Client):** Views artisan profiles and schedules appointments.
- **Artisan (Service Provider):** Updates availability and manages bookings.
- **System:** Facilitates the booking process, checks schedule conflicts, and sends notifications.

Preconditions

- The customer must select a specific service provider.
- The service provider must have their availability updated in the system.
- The system must ensure accurate synchronization of availability calendars.

Postconditions

- A booking is successfully created and stored in the system.
- Both the customer and the service provider receive confirmation notifications.
- The booking details are accessible in the customer's order history and the service provider's schedule.

Steps

1. The customer logs in and navigates to the service provider's profile.
2. The customer views the service provider's availability calendar.
3. The customer selects a preferred date and time slot for the booking.
4. The system performs the following actions:
    - Checks the service provider's schedule for conflicts.
    - Validates the selected time slot.
    - If the slot is unavailable, the system prompts the customer to choose a different time.
    - If the slot is available, the system confirms the booking.
5. Notifications are sent to both parties:
    - **Customer:** Receives a booking confirmation with details (date, time, and service provider information).

    - **Service Provider:** Receives a notification of the new booking.
6. The booking details are securely stored in the database:
    - Added to the customer's order history.
    - Updated in the service provider's schedule.

Favorites

Description

The favorites feature allows customers to save their preferred artisans for quick access and future bookings. This functionality enhances user convenience by providing a personalized list of artisans directly accessible from the customer's dashboard.

Actors

- **Customer (Client):** Saves favorite artisans for future use.
- **System:** Facilitates adding artisans to the favorites list, storing the data, and displaying it when needed.

Preconditions

- The customer must be logged into their account.
- The customer must have interacted with a service provider, such as viewing their profile or booking a service.

Postconditions

- The selected artisan is successfully added to the customer's favorites list.

- The updated favorites list is accessible from the customer's dashboard.

Steps

1. The customer logs in and navigates to the service provider's profile.
2. The customer clicks the **"Add to Favorites"** button on the profile page.
3. The system performs the following actions:
    - Confirms the action with the customer (e.g., via a confirmation popup or message).
    - Updates the customer's favorites list in the database.
    - Ensures the favorite artisan is linked to the specific customer account.
4. The system provides confirmation to the customer (e.g., "Artisan added to your favorites").
5. The customer can view their favorite artisans in a dedicated **Favorites** section on their dashboard for future reference.

Order History

Description

The order history feature allows customers to view past service

requests and transactions. This functionality provides a structured record of completed bookings, enabling customers to track service history and view transaction details, including feedback and costs.

Actors

- **Customer (Client):** Views past service requests and transaction details.
- **System:** Retrieves, displays, and organizes past bookings and transactions.

Preconditions

- The customer must have completed at least one service booking.
- The system must store and organize past bookings and transaction data.

Postconditions

- Past bookings and associated transactions are successfully displayed in a structured format.
- Customers can filter and access detailed information about individual orders.

Steps

1. The customer logs in and navigates to the **Order History** section from their dashboard.
2. The system retrieves all completed bookings and associated transaction details from the database.
3. The system displays the order history in a structured format (e.g., list or table), including:

    - Service type
    - Artisan details
    - Service date and time
    - Cost of service
    - Feedback (if applicable)
4. The customer can filter the history by:
    - Date range
    - Service type (e.g., plumbing, electrical, etc.)
    - Artisan name or service provider
5. The customer clicks on a specific order to view detailed information, including:
    - Service date and time
    - Cost breakdown
    - Feedback or rating left by the customer (if applicable)
6. The customer can also leave additional feedback or request support if issues arise with past services.

4. Search and Filtering

Service Search

Description
The service search feature enables customers to find artisans based on specific criteria such as skills, location, or service type. This functionality enhances user experience by helping customers efficiently locate suitable artisans for their needs.

Actors

- **Customer (Client):** Searches for artisans based on skills, location, or service type.
- **System:** Processes the search query, retrieves relevant results, and displays them to the customer.

Preconditions

- The system must have a database of registered artisans, their skills, and available services.
- The system must be capable of processing search queries and filtering results.

Postconditions

- Relevant search results, including artisan profiles, are displayed to the customer based on their query.
- The customer can view detailed profiles or proceed to book services.

Steps

1. The customer logs in and navigates to the **Service Search** feature.
2. The customer enters a search term in the search bar (e.g., “plumber,” “electrician,” “kitchen renovation”).

3. The system processes the query by matching it with artisan profiles, considering skills, location, or service type.
4. The system retrieves relevant artisans from the database and sorts the results by relevance, taking into account factors like:
    - Skills or specialties
    - Location (proximity to the customer)
    - Service type or category
5. The results are displayed in a list or grid format, including basic information such as:
    - Artisan name
    - Service types offered
    - Location or service area
    - Average rating or reviews
6. The customer can filter results further by:
    - Service type (e.g., electrical, plumbing, carpentry)
    - Location (e.g., within a specific radius)
    - Ratings or reviews (e.g., top-rated artisans)
7. The customer can view individual artisan profiles for more details, including:
    - Detailed skills and expertise
    - Portfolio or past work
    - Availability and contact information
8. If the customer finds a suitable artisan, they can proceed to book a service.

Filters

Description
The filters feature allows customers to refine their search results using various criteria such as price range, ratings, and distance (if location-based). This functionality enables a more tailored and efficient search experience, helping customers find the most suitable artisans based on their specific preferences.

Actors

- **Customer (Client):** Refines search results using filters such as price range, ratings, and distance.
- **System:** Applies the selected filters to search results and displays the updated results in real-time.

Preconditions

- The customer must have initiated a search or be browsing a category (e.g., service type or artisan list).
- The system must have available data on pricing, ratings, and distance (for location-based searches).

Postconditions

- The search results are filtered according to the customer's selected criteria, and only matching results are displayed.
- The customer can adjust filters or proceed to book a service.

Steps

1. The customer logs in and navigates to the **Service Search** or browsing section.

2. The customer initiates a search or selects a category to browse (e.g., plumbing services, electricians).
3. The system displays the search results based on the customer's query.
4. The customer selects one or more filters from the available options:
    - **Price range**: Select a minimum and maximum price for the service.
    - **Ratings**: Choose a minimum rating threshold (e.g., 4 stars and above).
    - **Distance** (if location-based): Choose a proximity radius for nearby artisans (e.g., within 10 miles).
5. The system applies the selected filters to the current search or browsing session.
6. The system updates and displays the filtered results in real-time, showing artisans that match the selected criteria.
7. The customer can view the updated results and proceed to:
    - Adjust the filters (if necessary) to further refine the search.
    - View artisan profiles for more information.
    - Book a service if they find a suitable artisan.

Category Browsing

Description

The category browsing feature groups services into distinct categories (e.g., plumbing, carpentry, electrical work) to make it

easier for customers to navigate the platform and find relevant artisans. This functionality improves the user experience by organizing services in a logical manner.

Actors

- **Customer (Client):** Browses available services by category and views artisans offering those services.
- **System:** Displays services grouped by category and provides the ability to filter or view details of artisans within each category.

Preconditions

- The system must have a well-organized database of services and artisans, with each service correctly assigned to a relevant category.
- The customer must be logged into their account to interact with the platform (optional but recommended for a personalized experience).

Postconditions

- The customer is presented with a list or grid of services within the selected category.
- The customer can filter the results or directly proceed to view profiles or book a service.

Steps

1. The customer logs in and navigates to the **Category Browsing** section (e.g., from the homepage or the main menu).

2. The customer selects a category (e.g., plumbing, carpentry, electrical work) from the available list.
3. The system retrieves all registered services and artisans offering services within the selected category.
4. The system displays the services in a structured format (e.g., list or grid view).
5. The customer can:
    - View the artisans' profiles and service details.
    - Apply filters (e.g., price range, ratings, location) to refine the results.
    - View service provider availability and request a service.
6. The customer can click on any specific artisan or service to view detailed information or proceed with booking the service.

1.

khaled

5. Communication

- **Notifications**: Provide email notifications for updates like booking confirmations, service completion, or feedback reminders.

  Description: The system will send notifications to users (clients and artisans) for key events such as booking confirmations, service completion, and reminders for feedback submission. These notifications can be delivered via email.

  Actors:

- **Regular Users (Clients):** Receive notifications for booking updates, service status, and feedback reminders.
- **Service Providers (Artisans):** Receive notifications for service requests, booking confirmations, and service completion reminders.
- **System:** Manages and sends the notifications to the appropriate users.

Preconditions:

- The user must be registered and logged into the platform.
- The notification service email must be integrated and operational.
- The user must have enabled notifications for the specific event email.
- The system must track events such as bookings, completions, and feedback submissions.

Postconditions:

- The user successfully receives the notification via email .
- Notifications are logged for future reference and can be accessed by users if needed.

Steps:

1. **Booking Confirmation Notification:**
    - When a client books a service, the system will confirm the booking and send a notification.

    - The notification can be sent through email
    - The system records the event of booking confirmation and sends it to the client and artisan.
2. **Service Completion Notification:**
    - Once a service is completed by the artisan, the system will send a service completion notification to both the client and the artisan.
    - This notification may include a prompt to rate or provide feedback on the service.
3. **Feedback Reminder Notification:**
    - After service completion, the system sends a reminder notification to the client asking for feedback on the artisan's performance.
    - This could be sent via email .
4. **User Interaction with Notifications:**
    - The user receives a notification, either via email
    - email, the user can follow the link or action items (e.g., rate service, view booking details).
5. **Logging Notifications:**
    - Each notification is logged in the system, enabling the user to view past notifications and actions taken.
    - The system stores the notifications for future reference, allowing users to check notifications at any time.

6. Payments

- **Payment System**: Allow cash-on-delivery

Description: The system enables clients to pay artisans in cash upon completing a service. This ensures direct payment between the client and the artisan while allowing the platform to track payment statuses for administrative and record-keeping purposes.

Actors:

- **Regular Users (Clients):** Responsible for paying artisans directly in cash after the service is completed.
- **Service Providers (Artisans):** Receive cash payments and confirm the payment status in the system.
- **System:** Tracks payment status, generates receipts, and records transactions for both parties.

Preconditions:

1. The client must have a confirmed booking with the artisan.
2. The artisan must complete the service and mark it as "Completed" in the system.
3. The system must have functionality to track payment status (e.g., "Pending," "Paid").

Postconditions:

1. The payment status is updated to "Paid" in the system.
2. A digital receipt is generated and recorded in the client's and artisan's transaction histories.
3. The system updates the booking record to reflect successful payment.

Steps:

- **Service Completion:**
  - The artisan completes the requested service.
  - The artisan updates the system to mark the booking as "Completed."
- **Cash Payment:**
  - The client pays the artisan directly in cash after verifying that the service has been completed.
- **Payment Confirmation:**
  - The artisan logs into the system and marks the payment status as "Paid."
  - The system verifies the booking and updates the payment status in the database.
- **Receipt Generation:**
  - Once the payment is confirmed, the system generates a digital receipt containing the transaction details (e.g., client name, artisan name, service type, and payment amount).
- **Notification:**
  - The system sends a notification to the client confirming the successful payment and provides a link to download the receipt.
- **Record-Keeping:**
  - The system logs the transaction details for both the client and artisan, making them available in their respective transaction histories.

- **Pricing Transparency**: Display the estimated cost of a service before booking.

Description:

The system displays the estimated cost of a service to clients before they confirm a booking. This ensures pricing transparency and allows clients to make informed decisions about their service requests.

Actors:

- **Regular Users (Clients):** View the estimated cost of a service before proceeding with booking.
- **System:** Calculates and displays the estimated cost based on predefined service prices and details provided by artisans.

Preconditions:

1. The artisan must have provided base pricing for their services in the system.
2. The system must be able to retrieve the artisan's pricing information from the database.
3. The client must be logged in to browse and book services.

Postconditions:

1. The estimated cost is displayed to the client during the booking process.
2. The client can decide to proceed with the booking based on the displayed price.
3. The system logs the estimated cost as part of the booking record.

Steps:

- **Service Selection:**
  - The client browses available services and selects a specific service (e.g., plumbing or carpentry).
- **Retrieve Pricing Data:**
  - The system retrieves the artisan's pricing information from the database, including any additional details like minimum service cost or per-hour rates.
- **Display Estimated Cost:**
  - The system displays the estimated cost to the client before they confirm the booking. This information is shown on the service request page or during the booking process.
- **Client Confirmation:**
  - The client reviews the estimated cost and confirms the booking if they agree to the price.
- **Record Pricing Details:**
  - The system records the estimated cost as part of the booking record for future reference by both the client and the artisan.

7. Quality Assurance

- **Service Rating System**: A detailed rating system for artisans based on punctuality, quality, and professionalism.

Description:

The system allows clients to rate artisans after completing a service. The rating system evaluates artisans based on specific criteria, such as punctuality, quality of work, and professionalism. This helps maintain service quality and allows future clients to make informed decisions.

Actors:

- **Regular Users (Clients):** Provide ratings and feedback after receiving a service.
- **Service Providers (Artisans):** View aggregated ratings and feedback to improve their services.
- **System:** Stores ratings, calculates average scores, and displays them on artisan profiles.

Preconditions:

1. The service must be marked as "completed" in the system.
2. The client must have received the service to be eligible to leave a rating.
3. The artisan’s profile must be active and associated with the completed service.

Postconditions:

1. The rating and feedback are stored in the system.
2. The artisan's profile is updated with the new rating and aggregated average score.
3. Future clients can view the updated ratings and feedback on the artisan's profile.

Steps:

1. **Service Completion:**
    - The client confirms that the service has been completed in the system.
2. **Prompt for Rating:**
    - The system prompts the client to rate the artisan based on predefined criteria, such as:
        - **Punctuality:** Was the artisan on time?
        - **Quality:** Was the work performed to satisfaction?
        - **Professionalism:** Was the artisan courteous and professional?
3. **Submit Feedback:**
    - The client assigns a score (e.g., 1–5 stars) for each criterion and optionally provides written feedback.
4. **Store Ratings:**
    - The system saves the ratings and feedback in the database, associating them with the relevant artisan and service booking.
5. **Calculate Average Rating:**
    - The system recalculates the artisan’s average rating based on all submitted reviews.
6. **Update Artisan Profile:**

    - The system updates the artisan's profile to reflect the new average rating and displays the latest feedback in their reviews section.
7. **Client Confirmation:**
    - The client receives a confirmation that their feedback has been successfully submitted.
8. **View Ratings:**
    - Future clients can view the aggregated ratings and reviews when browsing artisans or selecting one for a service.

8. Admin Panel

- **User Management**: Admins can manage both customers and artisans.

Description:
Admins can manage platform users, including both customers and artisans, by viewing and updating their accounts.

Actors:

- **Admin Users**
- **System**

Preconditions:

1. The admin must be logged in with valid credentials.
2. User data (customers and artisans) must exist in the system.

Postconditions:

1. Changes to user accounts are saved and reflected in the system.
2. Audit logs are updated to track admin actions.

Steps:

- Admin logs in and navigates to the "User Management" section of the admin panel.
- The system displays a list of users with relevant details (e.g., names, emails, roles, and account status).
- The admin can search for users using filters (e.g., role: artisan or customer.
- Admin selects a user to view detailed account information.
- Admin performs actions like updating user details,
- The system confirms the action and updates the database.
- The system logs the admin's action for accountability.
- Admin logs out after completing the task.

- **Service Monitoring**: View and track all service requests and bookings on the platform.

2. Service Monitoring

Description:

Admins can view and track service requests and bookings on the platform to ensure smooth operations.

Actors:

- **Admin Users**
- **System**

Preconditions:

1. The admin must be logged in with valid credentials.
2. Service request and booking data must exist in the system.

Postconditions:

1. The admin gains visibility into ongoing and completed service requests.
2. The system tracks admin views for audit purposes.

Steps:

- Admin logs in and navigates to the "Service Monitoring" section.
- The system displays a dashboard with all service requests and bookings, including details like:
  - Service ID
  - Customer name
  - Artisan name
  - Booking status (pending, in progress, completed)
  - Service date and time
- Admin can filter bookings by status, customer, or artisan.
- Admin selects a booking to view detailed information (e.g., service description, payment status, or customer feedback).
- The admin resolves any flagged issues (e.g., disputes or incomplete services).
- The system updates logs to reflect the admin's activities.
- Admin logs out after completing monitoring tasks.

For the structrue

services/: Contains service-related logic for communicating with APIs.

authService.js: Handles API calls related to authentication (login, registration, etc.).

apiClient.js: A generic API client used to make HTTP requests.

interceptors.js: Used to intercept and modify HTTP requests/responses, such as for adding authorization tokens.

user authentication state.

ThemeContext.js: Context for managing the theme of the app (e.g., light/dark mode).

hooks/: Custom React hooks to simplify and reuse functionality.

useAuth.js: Custom hook to manage authentication-related actions (e.g., login, logout).

useForm.js: A hook for handling form inputs and validation.

utils/: Utility functions used throughout the frontend.

validation.js: Functions for form validation (e.g., checking if an email is valid).

helpers.js: Other helper functions, likely to include common logic like formatting or manipulating data.

styles/: Stylesheets for the frontend.

index.css: Main CSS file for global styles.

tailwind.css: TailwindCSS file for utility-first CSS styling.

App.js: The root component of your React app that organizes and renders all other components.

package.json: Lists the dependencies and scripts required to run and build the frontend React app.

.env: Stores environment variables, such as API URLs or sensitive keys, that are needed by the frontend.

README.md: Provides documentation and setup instructions for the frontend application.