Assignment Category: B12-A10_category-0003

Explain Video: B12A10-Cat-03.mp4

HomeHero – Local Household Service Finder

Project Theme

HomeHero is a modern web application that connects users with trusted local service providers such as electricians, plumbers, and cleaners. Users can browse services, book appointments, and leave ratings, while providers can manage their listings.

Assignment Rules

GitHub & Documentation

- Minimum **15 commits** on the **client side**
- Minimum **8 commits** on the **server side**
- A well-written **README.md** file must include:
  - Website name
  - Live site URL
  - Minimum **5 key features** using bullet points.

Responsiveness

Your application must be fully responsive — optimized for **mobile, tablet, and desktop** views.

Environment Variables

Hide Firebase config and MongoDB credentials using `.env`.

Restrictions

- ❌No Lorem Ipsum text
- ❌No default JS `alert()` — use toast or SweetAlert2 for messages

Design

Use your creativity! You can take UI inspiration from sites like **ThemeForest** or **Dribbble**, but your project must be **unique, not** similar to module examples or previous assignments.

Deployment

- **Client:** Netlify / Surge / Firebase

- **Server:** Vercel
- Must not throw any reload errors
- Add your domain for Firebase authorization
- Private routes should remain accessible after reload

<u>Main Requirements</u>

<u>UI Design Requirements:</u>

★ **Unique Design:** First, decide what kind of website you want to make. Then, search online or check out websites like **ThemeForest** to get ideas for the design. But remember, your website idea shouldn't be similar to any projects you've done before or to any examples in our modules or conceptual sessions.

★ You can also look for free resources on blogs to help with your website.

1. Keep the main heading style (font, size, color) consistent across all sections.
2. Keep paragraph spacing balanced and text easily readable.
3. Maintain uniform image sizes and spacing.
4. Use the same button style as on the home page.
5. Ensure good spacing and proper alignment.
6. Navbar, keep the heading/logo same style and size as on the home page.
7. Use a grid layout with equal image sizes.
8. Keep all cards equal height and width (especially in services, projects, or products sections)
9. Use the new X logo instead of the old Twitter bird to match the latest rebrand
10. Responsiveness: Make it responsive for all devices, including mobile, tablet, and desktop views.

Resources:

- https://uiverse.io/

- https://devmeetsdevs.com/
- https://bootcamp.uxdesign.cc/free-images-and-resources-collection-for-website-c77f2fc46ce5
- https://themeforest.net/?srsltid=AfmBOopTj6PNz51iuV2YJXUtBP8nt19_zT5LG2dToAjIHQqzNCzregn0
- https://codecanyon.net/?srsltid=AfmBOooRoUfeK7lOROpchCuA4hPVj5P9WRmtDQJ9K0E6Yhf4VTrHhXKt

Layout Structure

- **Header:** Logo + Navigation (Home, Services, My Services, Add Service, My Bookings, Profile, Login & Register)
- **Outlet** to show your page content
- **Footer:** Copyright, social links, contact info, and other info

Note: Here, My Services, Add Service, My Bookings, and Profile pages will be **private**. These links will be hidden when the user is not logged in.

Home Page

- **Hero Section:** a slider with a minimum of 3 slides showing service image, headline, details, and an explore button. Clicking the explore button will redirect the user to the services page.
- **Dynamic Section:** Display a total of six services (fetched from DB).
- **Animations**: using **motion(framer-motion)** or any other animation library to animate any of your homepage sections or the whole homepage.
- **Two Static Sections:** *(students' choice). Design is up to you.*
    - e.g., "Why Choose Us" / "Customer Testimonials"

Authentication

User Login

- Fields: Email, Password, Forget Password (optional), Login button
- Include a **Google Login** option
- On **success** → navigate to home or intended route

- On **failure** → show error via toast

□ **User Registration**

- Fields: Name, Email, Photo URL, Password, Register button
- **Password validation:**
    - At least 6 characters
    - One uppercase, one lowercase
- On success → navigate to home
- Include a **Google Login** button
- *Don't implement email verification, as it will inconvenience the examiner. If you want, you can add these after receiving the assignment result.*

Profile pages

- Design the user profile page as you wish, but it will show the user's basic info like: email, name, photo, last login time, etc.
- Update Profile option: implement the user update profile option to update the user name and image. Design is up to you.

□ CRUD Operations

□□ For Service Providers

Add Service Page

- Fields: Service Name, Category, Price, Description, Image URL, Provider Name, Email
- POST → Save to MongoDB (with provider email reference)

My Services Page

- Fetch only the logged-in provider's services. Show in table format. Also, some action buttons.

Update Service

- Provider can edit Service details (PATCH/PUT request)

Delete Service

- Provider can delete any of their own services

□ <u>For Customers</u>

<u>Service List Page</u>

- Display all services data in card format. Each card will include some info and a details button.
- Clicking the details button will navigate the user service details page.

<u>Service Details Page</u>

- Full info of that service and a **Book now** button.
- Service details page design is up to you. Use your creativity. Make sure to show all the detailed info stored in the DB.

Booking System

- Clicking the **book** button will open a modal. The modal will show some info about the service, and below, there will be a form with some info.
    - Store in bookings collection with fields:
        - userEmail (read-only), serviceId, bookingDate, price

□Note: The user can't edit the email field as it will come from Auth.

<u>My Bookings Page:</u>

- Display all user-booked services in a table format, including important information. Also,
- Include a cancel (DELETE) option
- Clicking the cancel button will cancel the booking. This means the data will be deleted from your database.

□ Database Design

Collections:

- Services → stores all service listings.
- Bookings → stores booking data.

Relations:

- Bookings.userEmail ↔ firebase user email
- bookings.serviceId ↔ services._id

⚡ Other Requirements

- **Loading State:** Show spinner/skeleton while fetching data
- **Error Page:** Custom 404 page with a “Back to Home” button
- **Notification:** Use **React Hot Toast** or **SweetAlert2** for success/error, also for all CRUD operations.

▯ Challenge Requirements (Advanced Features)

▯ These features add difficulty and help you stand out

1. ▯ **Price Filtering (MongoDB Operators)**
    - Use $gte and $lte to filter services by price range. The filter system will be implemented in the services page.
2. ⋆**Rating and reviews System**
    - Allow users to submit ratings for booked services. This functionality can be implemented in my booking page.
    - There will be property inside the single service object called reviews. When booked, the review will be stored in that service as an array of objects.
    - Show all reviews and the details page.
    - Also, only show the top six-rated services section on the homepage
3. **Theme Customization:**
    - Add a Theme Toggling Button for changing themes from light to dark/dark to light. By changing the theme, it will make the full system dark/light based on user interaction.
4. ▯ **Booking Restriction**
    - The user can’t book his service. Show toast or disabled booking button.

▯ Optional Features

These are not mandatory, but will help your project stand out:

1. **Search & Filter System:**
    - Use MongoDB $regex or $or for searching by service name/category. Search functionalities will be case-insensitive.
2. **Firebase Token and Route Verification:**

- Use **Firebase Authentication** to handle user login and token verification. When a user logs in, Firebase generates a **token**, which should be stored securely on the client side (for example, in localStorage or a cookie). For each API request, send this token in the **Authorization header**. On the server, use the **Firebase Admin SDK** to verify the token before allowing access to protected routes. If the token is valid, grant access; if it's missing or invalid, respond with 401 Unauthorized or 403 Forbidden.

3. **Provider Profile Page (Profile page):**
    - **On the profile page**, display all their services(count), total bookings of their service, total revenue, and average rating, show all the info using charts, graphs, etc., see this Google search result: Google

What to Submit

1. **Client Repository (GitHub)**
2. **Server Repository (GitHub)**
3. **Live Site URL**

Tip for Students:

Focus on completing all CRUD functionalities first. Then move to authentication, filtering, and deployment. Finally, polish your UI and add bonus features to stand out!

—------------------------------- **All The best** —-------------------------------