**Full Stack Development with MERN**

**Project Documentation format**

**1. Introduction**

* **Project Title:** Hire Connect
* **Team Members:** Kritikraj, V.Sai Harika, P.Sri Charan, D.Moksha Sree

**2. Project Overview**

* **Purpose:** Hire Connect is a platform designed to connect freelancers with clients. It enables users to search for freelance jobs, apply for them, and manage their profiles. Clients can post jobs, manage applications, and process payments seamlessly.
* **Features:**

**User Profiles:** Create and manage detailed profiles.

**Job Listings:** Post and apply for freelance jobs.

**Subscription Plans:** Free and premium plans for freelancers.

**Secure Payment Processing:** Automated payments and subscription handling.

**Chat Functionality:** Direct communication between clients and freelancers.

**Rating System:** Rate and review completed jobs.

**3. Architecture**

**Frontend:**

* Built using React.js.
* Key components include user profile pages, job listings, and chat interface.
* App.js is the entry point of the application.

**Backend:**

* Node.js and Express.js are used for building the server.
* MongoDB is used for database management.
* Models include Users, Jobs, Applications, and Payments.
* Controllers handle business logic, and routes are defined in Express..

**Database:**

**Schemas:**

* Users: Stores user information.
* Jobs: Contains job details.
* Applications: Tracks applications made by freelancers.
* Payments: Manages payment transactions and subscription statuses.

**4. Setup Instructions**

* **Prerequisites:** Node.js, express.js, mongoDB, React.js
* **Installation:**

**client**:

npm install

npm start

**server**:

create a mongodb database in mongodb and then connect with mongodb compass and get the link; then save it as MONGOURI in .env file

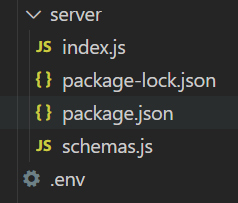
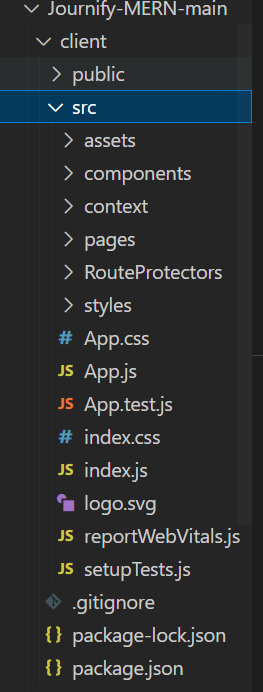
run the commands: npm install npx nodemon index.js

**5. Folder Structure**

* **Client and Server:**

/client: Contains React.js application.

/server: Contains backend files including models, controllers, and routes.



**6. Running the Application**

**client:**

npm install

npm start

**server:**

create a mongodb database in mongodb and then connect with mongodb compass and get the link; then save it as MONGOURI in .env file

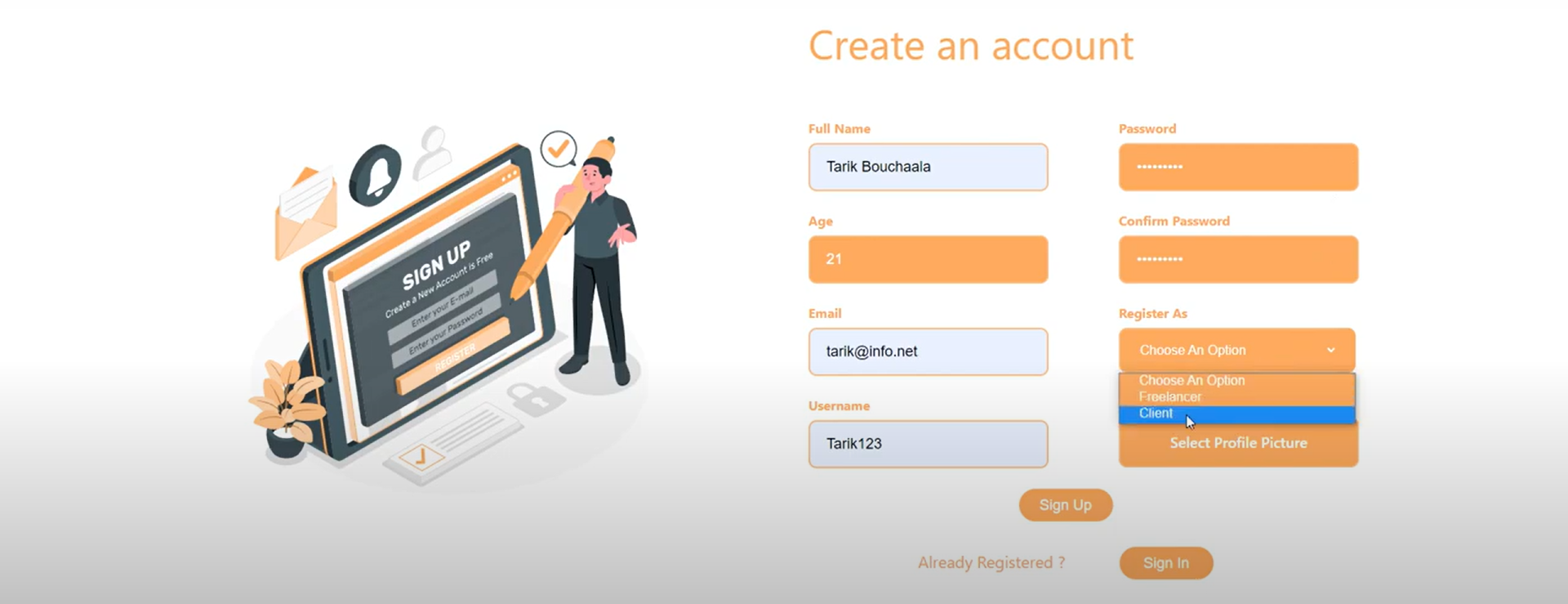
run the commands: npm install npx nodemon index.js

**7. Authentication**

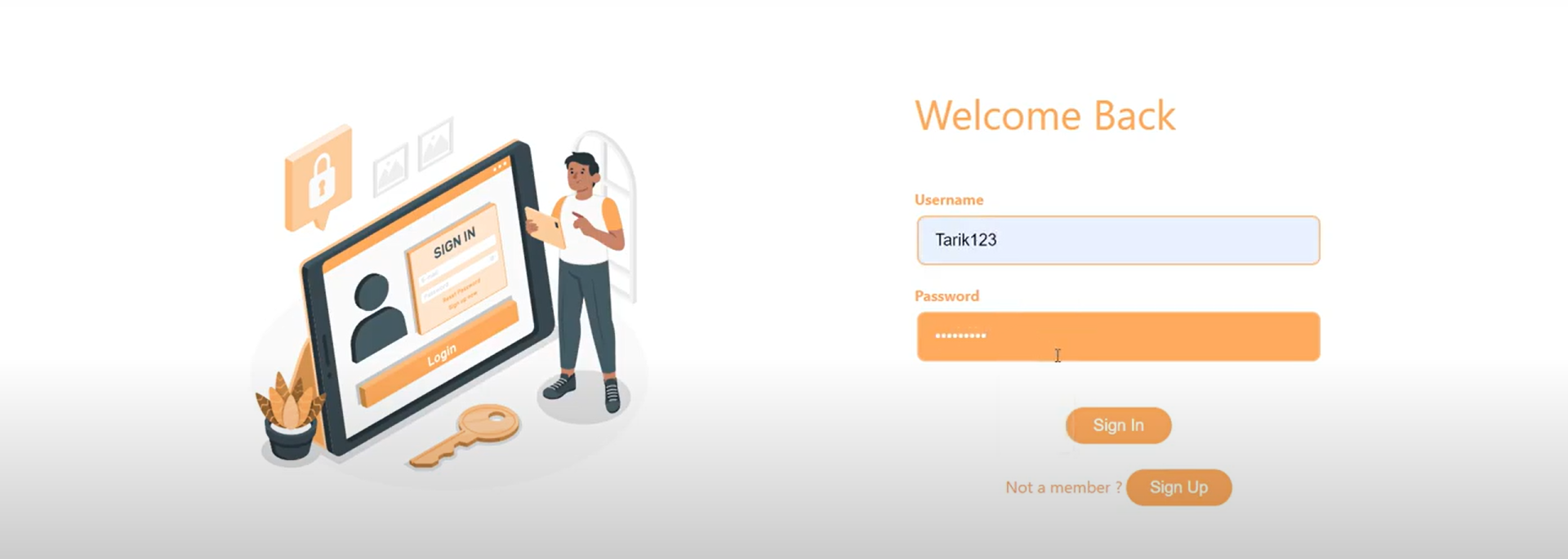
* The Authentication is done by the backend server using MongoDB which will encrypt the password entered by the user in the signup page.

**9. User Interface of Journify**

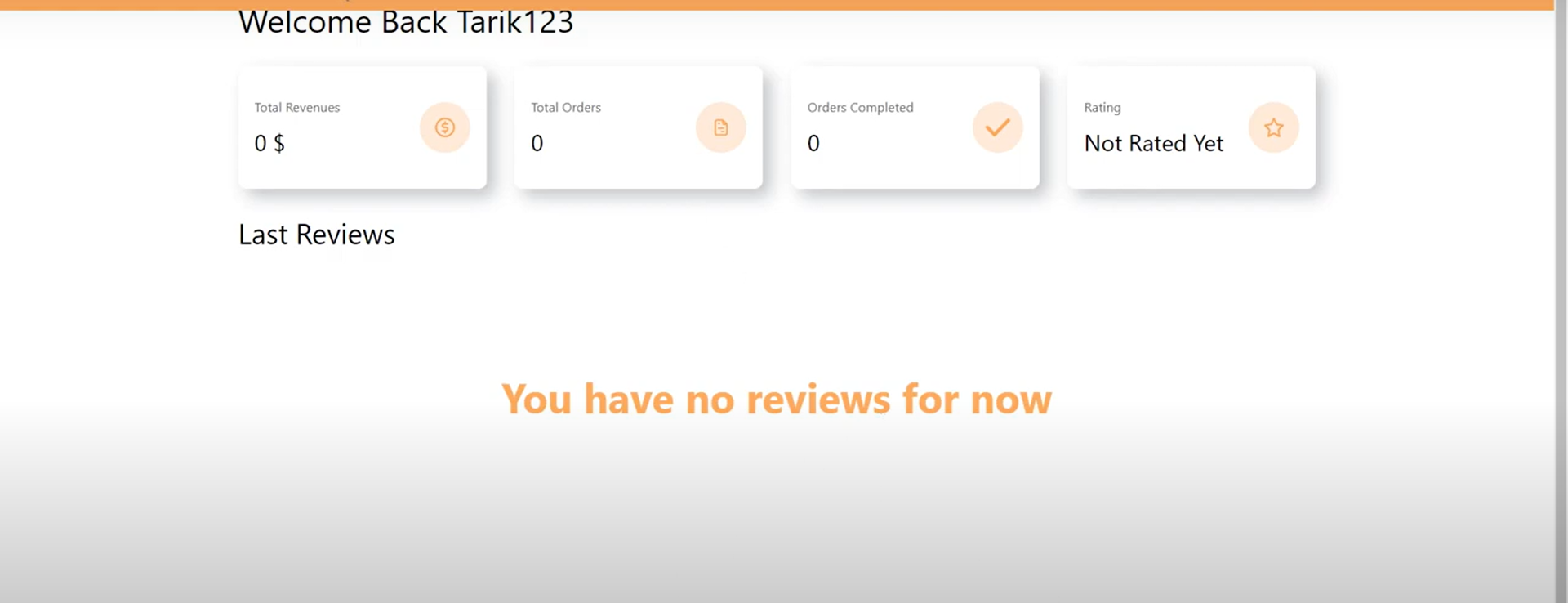
**Registering yourself:**



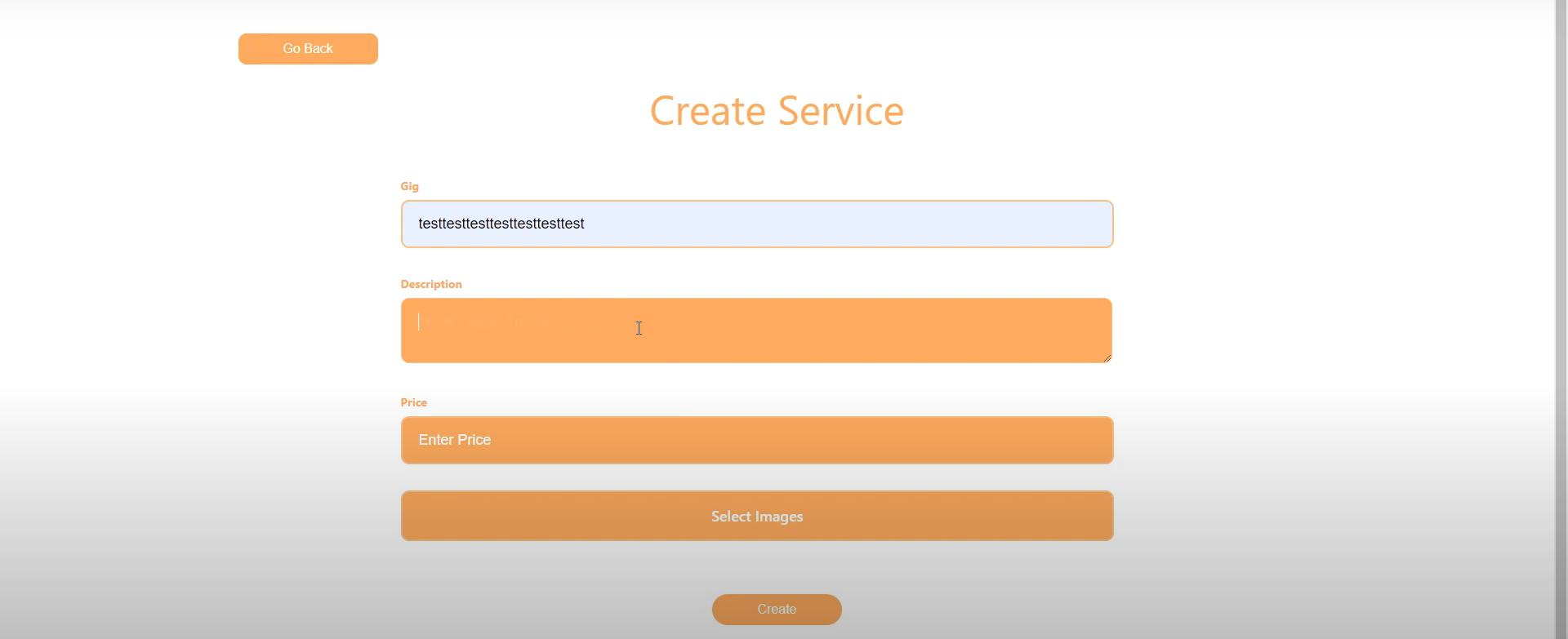
**Login Page:**



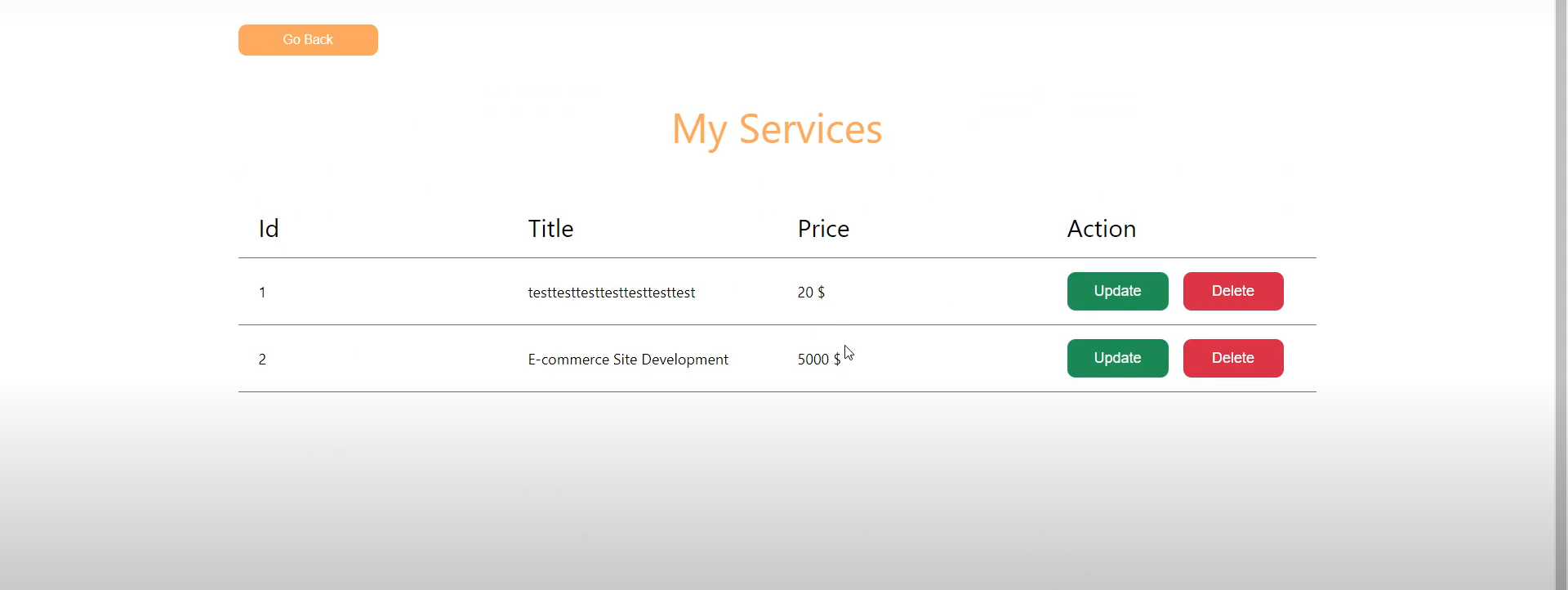
**Home Page:**



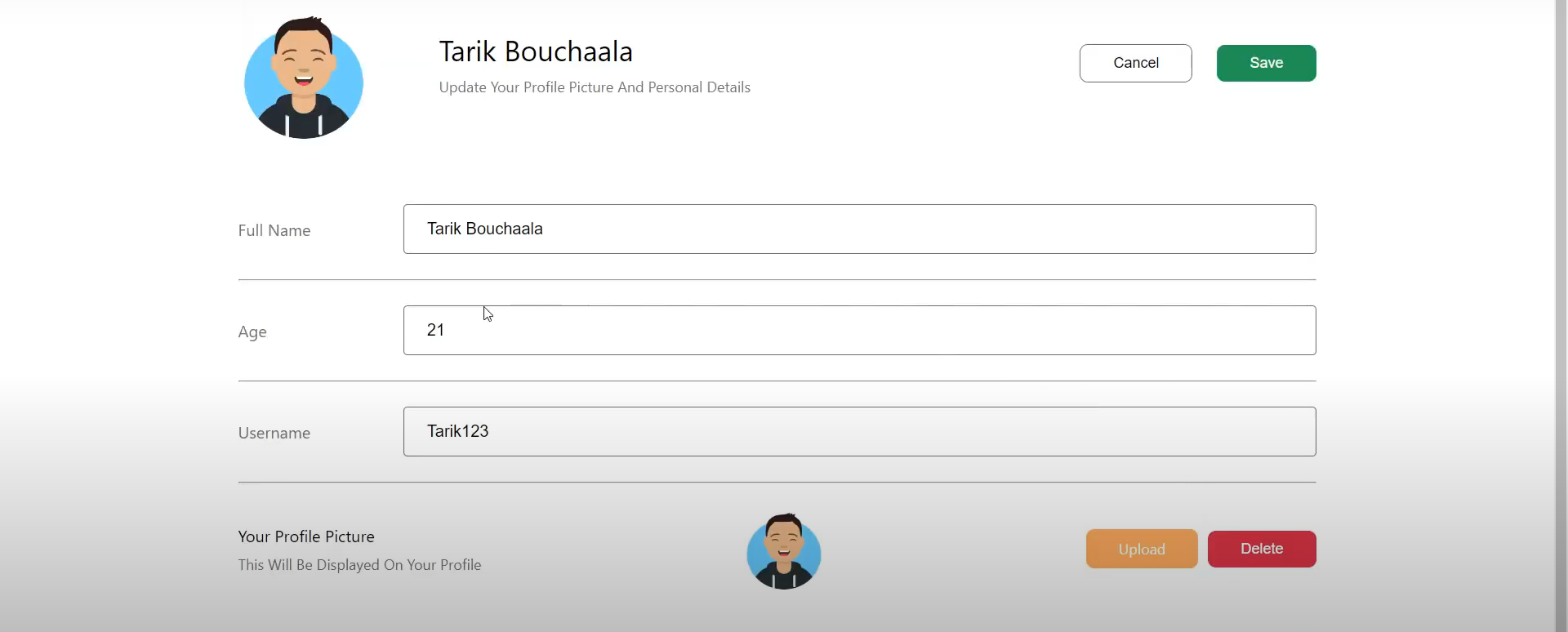
**Creating Services:**



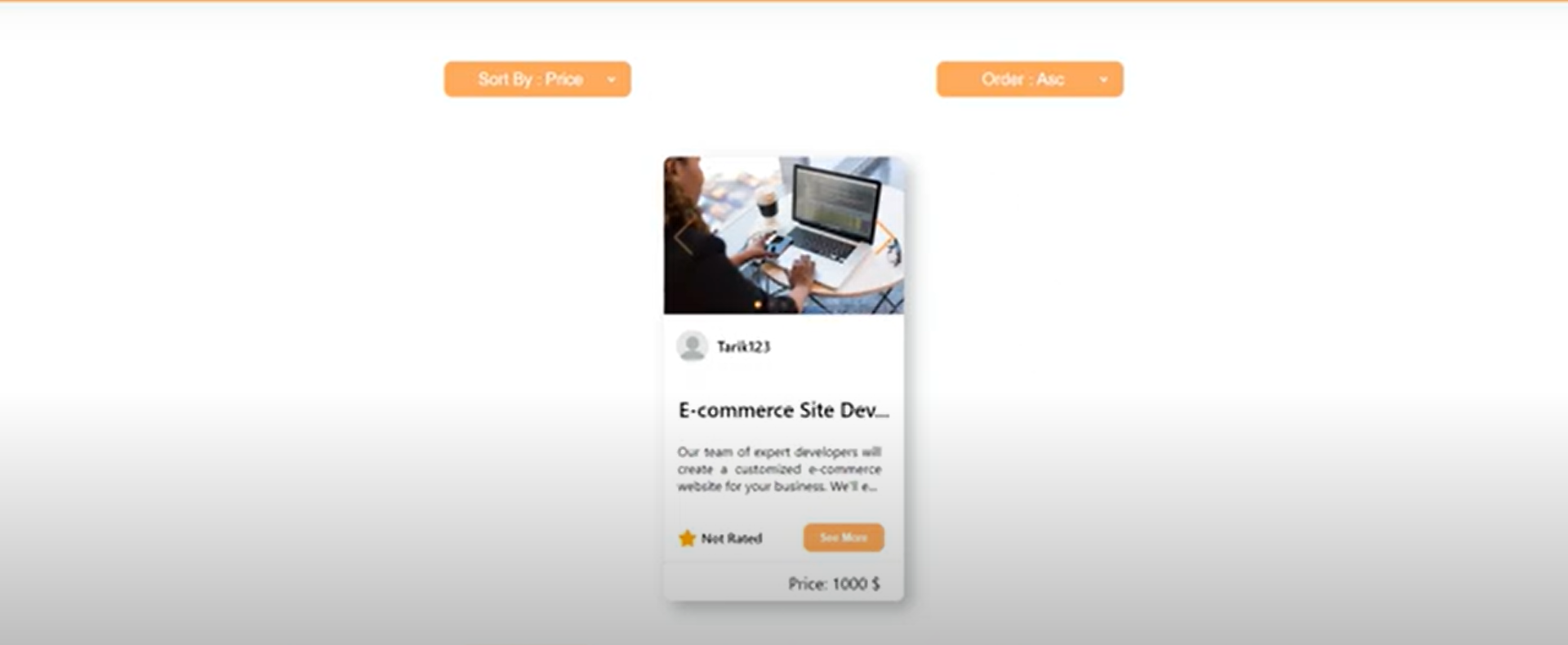
**Managing services:**

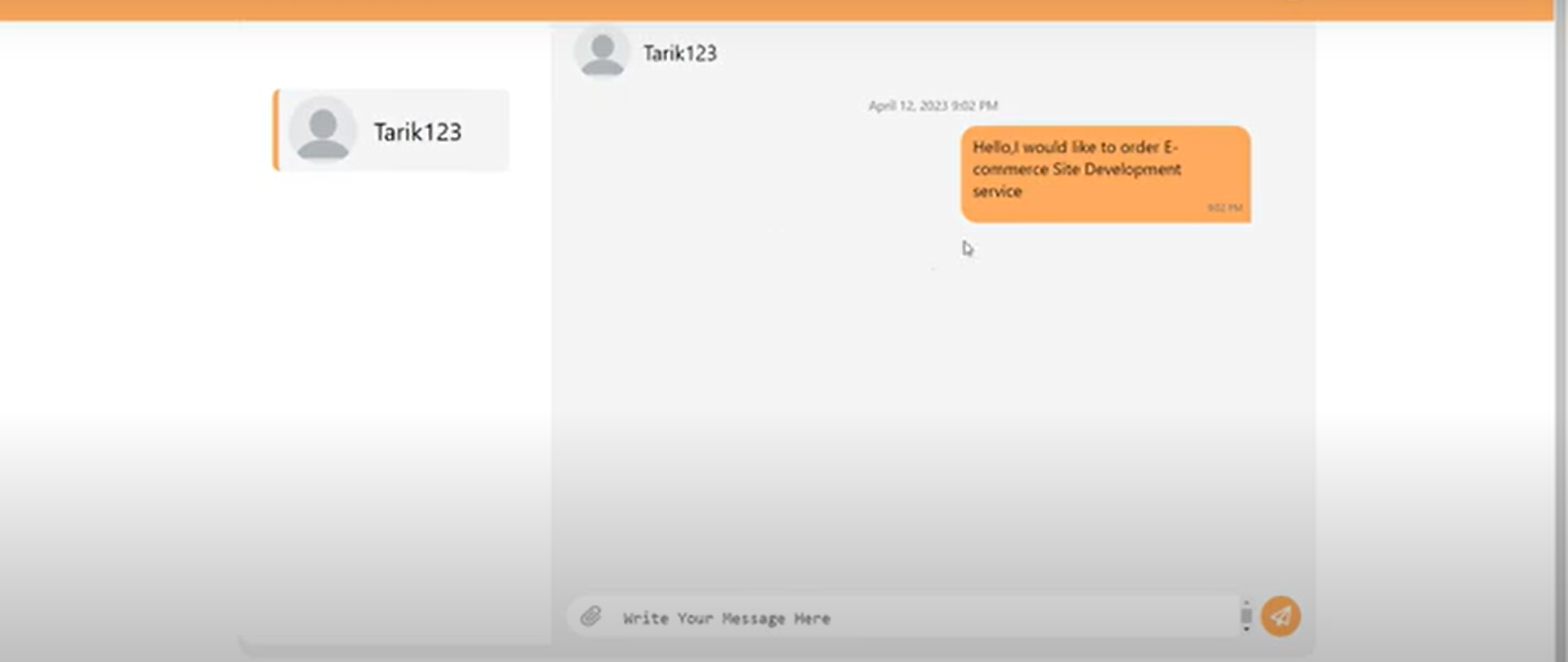


**Profile updation:**

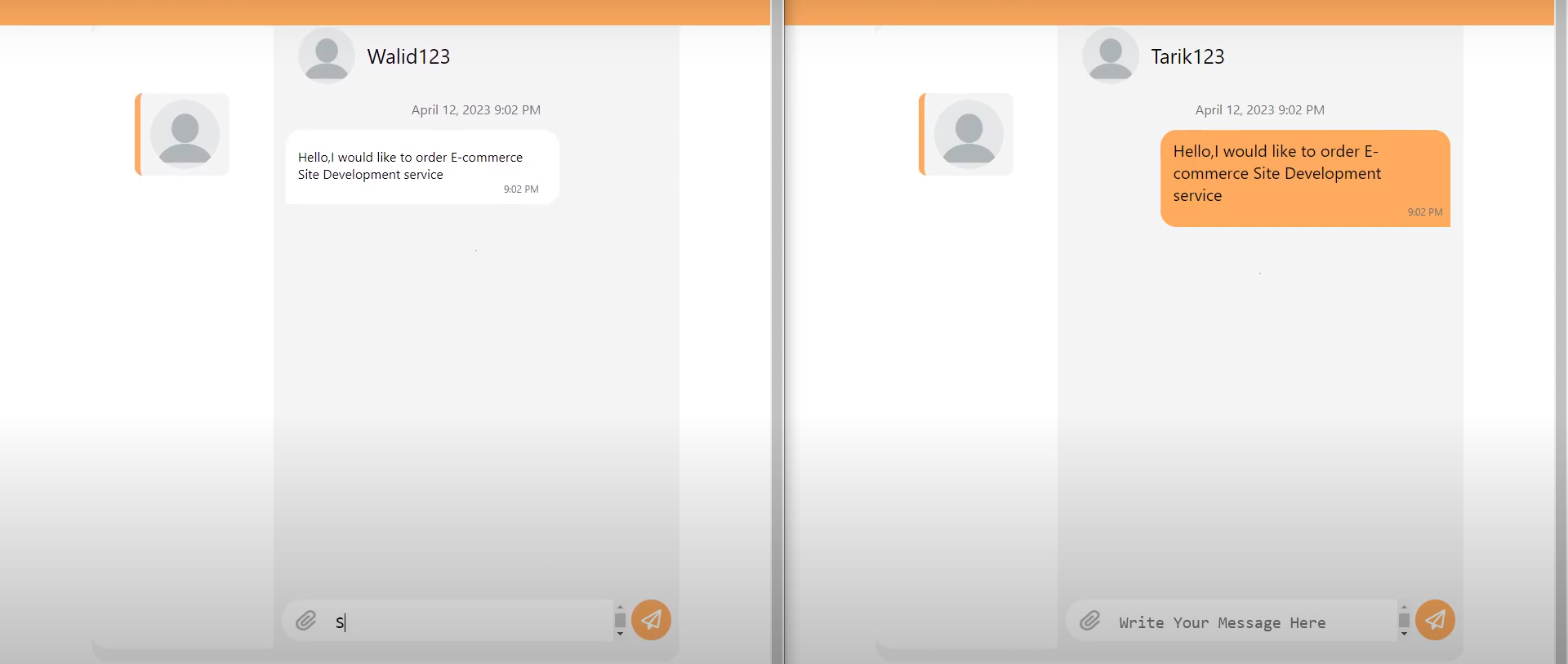


**Order summary:**

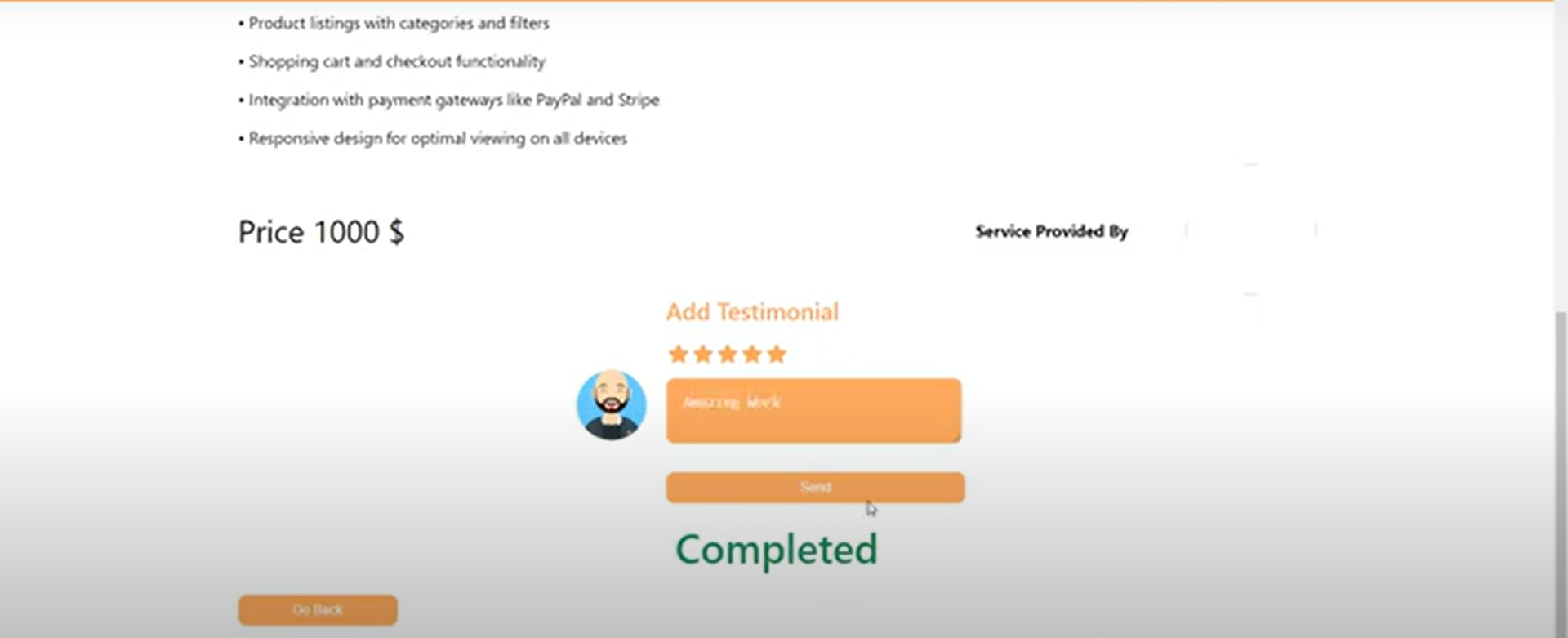
 **chat option:**

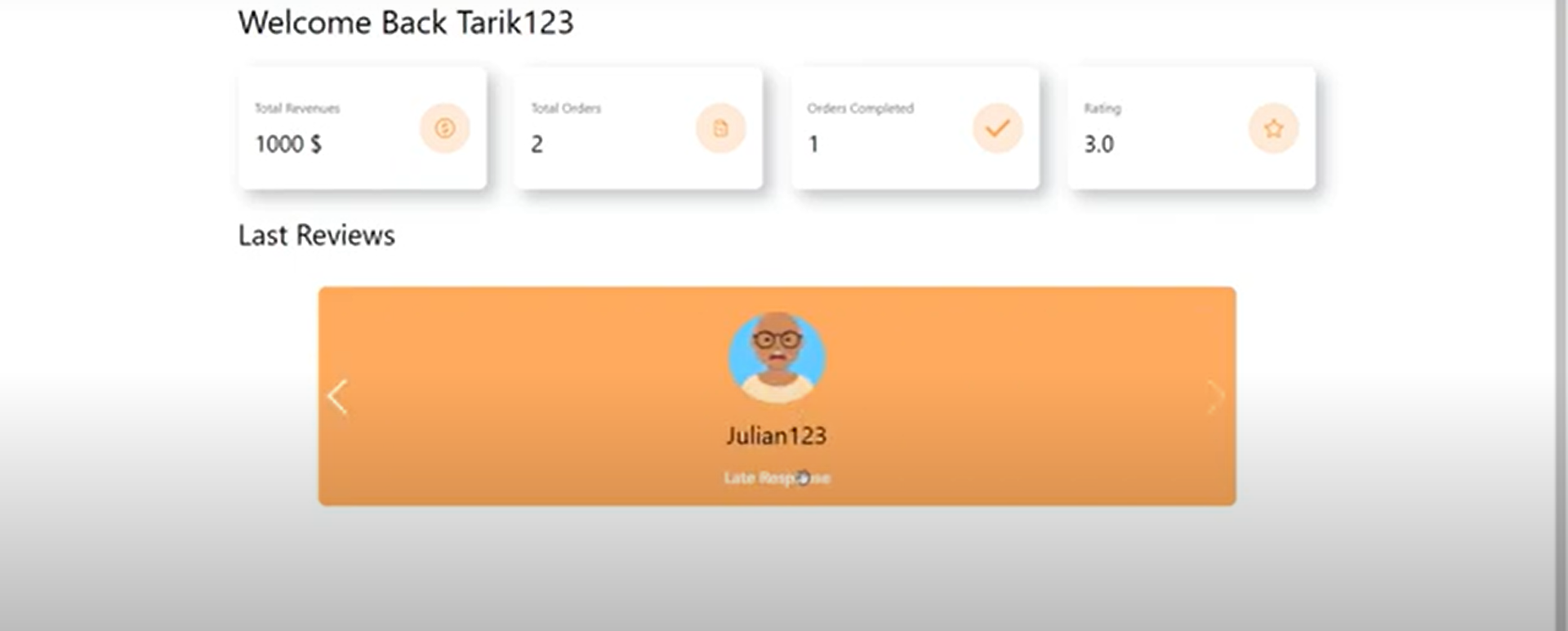


**Chat interaction between freelancer and client:**



**Testimonial updation:**

 **Reviewing:**



**10. Testing**

We have tested our page manually by giving it to our college friends, and other associates. Their responses and feedback were received by us and we worked on them to make Journify reach its maximum potential.

**11. Screenshots or Demo**

<https://drive.google.com/file/d/1fYF2ySJIU6uUGvyfwfpwN1yAd7eUJY0X/view>

**12. Known Issues**

No known issues at the time of finalization.

**13. Future Enhancements**

* By contacting Airline Agencies, we can make the project more reality-oriented, as well as give it more functionalities.
* Feature Expansion: Consider integrating with third-party services like GitHub for portfolio showcasing.
* Automated Testing: Implement automated testing to ensure consistent functionality across updates.