HOUSE OF FLAIR

A PROJECT REPORT

Submitted By

Sarthak Singh

University Roll No- 2100290140117

Arpit Mishra

University Roll No- 2100290140038

Krati Gupta

University Roll No- 2100290140077

Submitted in Partial Fulfillment
of the Requirements for the
Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Shashank Bhardwaj (ASSOCIATE PROFESSOR)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions,
Ghaziabad Uttar Pradesh 201206

(June 2023)

CERTIFICATE

Certified that **Arpit Mishra 2100290140038, Krati Gupta 2100290140077, and Sarthak Singh 2100290140117** have carried out the project work having "**House of Flair**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or anybody else from this or any other University/Institution.

Arpit Mishra (University Roll No. 2100290140038)

Krati Gupta (University Roll No. 2100290140077)

Sarthak Singh (University Roll No. 2100290140117)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:27/05/2023

Dr. Shashank Bhardwaj
Associate Professor
Department of Computer Applications
KIET Group of Institutions Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Arun Kumar Tripathi
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

The freelance market is growing rapidly, with more and more people choosing to work independently. This growth has created a need for an online marketplace that connects freelancers with potential clients.

An e-marketplace freelancers, website would provide several benefits for both freelancers and clients. For freelancers, it would provide a platform to showcase their skills and find work. For clients, it would provide a way to find qualified freelancers quickly and easily.

The website would be easy to use and would allow freelancers to create profiles, upload portfolios, and set their own rates. Clients would be able to search for freelancers by skill set, location, and price.

The website would also provide several features to help freelancers and clients connect and collaborate. These features would include messaging, project management tools, and payment processing.

House Of Flair would be a valuable resource for both freelancers and clients. It would make it easier for freelancers to find work and for clients to find qualified freelancers. The website would also help to promote the freelance market and it will also help in the growth of the economy.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Shashank Bhardwaj, Associate Professor,** for his guidance, help, and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi, Professor, and Head, of the Department of Computer Applications**, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot in many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other help. Without their support, the completion of this work would not have been possible on time. They keep my life filled with enjoyment and happiness.

Arpit Mishra (2100290140038)
Krati Gupta (2100290140077)
Sarthak Singh (2100290140117)

List of Chapters

List of Abbreviations

S No.	Name of Abbreviations	Details of Abbreviations	Page
1	CS	Computer Science	1
2	OS	Operating System	1 - 2
3	Database Tables	Database Tables	9 -11
4	ER	Entity Relationship	12
5	UCD	Use Case Diagram	13
6	SD	Sequence Diagram	14

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In today's digital age, the concept of work and employment has undergone a significant transformation. The rise of freelancing has empowered individuals to showcase their unique skills, work on diverse projects, and enjoy flexible work arrangements. However, navigating the freelancing landscape can be challenging, both for freelancers seeking clients and for clients looking to connect with the right freelancers. That's where House of Flair comes in.

House of Flair is an innovative e-marketplace designed specifically for freelancers, providing them with a dedicated platform to showcase their talents, connect with clients, and build successful freelance careers. This introduction provides an overview of House of Flair, highlighting its mission, key features, and the benefits it offers to both freelancers and clients.

At its core, House of Flair aims to empower freelancers by offering a centralized and user-friendly platform that enables them to effectively market their skills, attract clients, and establish their professional reputation. By creating a compelling profile and showcasing their portfolio, freelancers can stand out in a competitive market, effectively communicating their expertise and unique value proposition to potential clients.

For clients, House of Flair serves as a valuable resource, offering a vast pool of talented freelancers from various industries and disciplines. Whether seeking graphic designers, writers, programmers, marketing professionals, or any other specialized skills, clients can easily browse through profiles, review past work, and engage with freelancers who best match their project requirements.

One of the key strengths of House of Flair lies in its commitment to facilitating seamless and secure transactions between freelancers and clients. The platform provides integrated messaging and communication tools, allowing both parties to communicate effectively, discuss project details, and negotiate terms. Additionally, House of Flair

ensures a transparent and fair payment process, giving freelancers peace of mind while receiving compensation for their services.

1.1 PROJECT DESCRIPTION

House of Flair is an ambitious project aimed at developing an advanced e-marketplace tailored exclusively for freelancers. This platform will provide freelancers with a robust and user-friendly environment to showcase their skills, connect with potential clients, and establish successful freelance careers. The project focuses on creating an efficient and secure marketplace that benefits both freelancers and clients in the gig economy.

Key features of the House of Flair platform include user-friendly profile creation, where freelancers can highlight their expertise, experience, and portfolio. Clients can post detailed project listings, specifying their requirements and budget, while freelancers can browse through these listings and submit proposals. The platform will offer secure messaging and collaboration tools, enabling seamless communication and negotiation between freelancers and clients. Furthermore, a reliable and transparent payment system will facilitate secure transactions, ensuring freelancers receive timely compensation.

1.2 PROJECT SCOPE

To create a platform that connects businesses with freelancers who can provide a variety of services, such as writing, design, development, and marketing.

- Develop a secure user registration and authentication system for freelancers and clients.
- Design an intuitive and user-friendly interface for profile creation and management, allowing freelancers to showcase their skills, experience, and portfolio.
- Implement project listing functionality for clients to post detailed project descriptions, requirements, and budgets.
- Enable freelancers to browse project listings, submit proposals, and communicate with clients through a messaging system.

- Integrate a secure payment gateway to facilitate transparent and timely transactions between freelancers and clients.
- Develop a review and rating system to gather client feedback and establish a reputation system for freelancers.

1.3 SOFTWARE USED

The following software components are necessary for the implementation of the House of Flair project.

- Operating System: Windows 8/8.1/10/11, MacOS
- Browser: Google Chrome, Microsoft Edge, Safari
- Database: MongoDB Atlas, A cloud-based database service for hosting and managing MongoDB databases.
- Technology: MERN (MongoDB, ExpressJS, ReactJS, NodeJS)
- Development Tools and Packages:
    1. Visual Studio Code: A popular source code editor that provides a rich set of features for efficient coding.
    2. Git: A version control system for tracking changes in code and collaborating with a development team.
    3. npm: The Node Package Manager for installing and managing project dependencies.

1.4 HARDWARE USED

The following hardware components are necessary for the implementation of the House of Flair project.

- Disk Space: 500GB
- Processor: 1.8GHz
- RAM: 4GB

CHAPTER 2

FEASIBILITY STUDY

After studying and analyzing all the existing and required functionalities of the system, the next task is to do a feasibility study for the project. The feasibility study includes consideration of all the possible ways to provide a solution to a given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily made based on future upcoming requirements.

2.1 ECONOMIC FEASIBILITY

For economic feasibility, Economic analysis or cost/benefits analysis is the most frequently used technique for the effectiveness of a proposed system. It is a procedure to determine the benefits and savings that are expected from the proposed system and compare them with cost. If the benefits outweigh the costs, a decision is taken to design and implement the system. Otherwise, further justification or alternatives in the proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves accuracy at each phase of a system life cycle.

Economic costs and benefits are not always the same as financial costs and benefits. Economic analysis includes project impacts that do not have a market price and positive and negative impacts that are experienced by people who are not the direct users of the services. It is in this way that economic analysis casts a broader net than a financial assessment. Accounting tools such as depreciation and capital charges should not be included in an economic analysis.

The impacts that would need to be considered will vary depending on the nature of the project and the sector. For example, a highway, roads, or public transport project will provide direct benefits to the users of the infrastructure or services provided but will also provide benefits to other road users if congestion on existing roads is reduced.

2.2 TECHNICAL FEASIBILITY

This included the study of function, performance, and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionalities to be provided in the system, as described in the System Requirement Specification (SRS) and checked if everything was possible using different types of frontend and backend platforms. This would include:

- Field surveys of the project site, which may include (depending on the project) mapping, topographical and geotechnical surveys.
- Analysis of environmental conditions that impact on the technical design. There may be some overlap between the information collected for this task and for the environmental impact assessment.
- A preliminary technical design of facilities required to provide the project outputs. This should consider alternative design options, considering uncertainty in the demand projections and other site-related uncertainties.

The technical specification should offer the least cost solution to meet the projected demand, standards, and other objectives. The preliminary design will also assist the Sponsor in appraising proposals received later at the bidding stage.

At this stage the technical design would not be final and would not be completed to the level of detail required for the final specifications. The focus here is on the project's technical feasibility, determining minimum technical requirements to be specified in the RFP, and on providing a design benchmark for estimating project costing to be used in the economic and financial analysis.

2.3 OPERATIONAL FEASIBILITY

No doubt the technology-growing world needs more enhancement in technology, this application is very user-friendly and all inputs to be taken are all self- explanatory even to a layman. As far as our study is concerned, the clients will be comfortable and happy as the system has cut down their loads and brought the young generation to the same virtual world they are growing drastically.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives about development schedule, delivery date, corporate culture, and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability, and others. These parameters are required to be considered at the early stages of design if desired operational behaviors are to be realized. System design and development require appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

CHAPTER 3

SYSTEM DESIGN

After analysis we gathered sufficient information to model the system. It provides appropriate guidance to system implementation. The main purpose of system design is to precisely build the system based on design requirements.

To design the system for the House of Flair project, we followed a systematic approach. Start by understanding the project requirements, including user roles, features, and data flows. Define a client-server architecture with a web-based frontend and a backend server. For the front end, choose technologies like React for building the user interface. We chose MongoDB as the database and designed an efficient schema to store user profiles, project listings, proposals, messages, and reviews. Integrate third-party services like payment gateways and email services for secure transactions and user communications.

Prioritize security by conducting regular audits, following secure coding practices, and encrypting sensitive data. Finally, continuously test and iterate the system design to optimize performance, usability, and security.

3.1 DESIGN METHODOLOGY SELECTED

There are basically two design methodologies that are being used to design system are as follows:

- Function Oriented Design
- Object Oriented Design

3.1.1 JUSTIFICATION FOR SELECTION OF OBJECT ORIENTED DESIGN

The selection of Object-Oriented Design (OOD) for the House of Flair project is justified by its inherent advantages. OOD promotes modularity and reusability, allowing for the creation of self-contained objects that can be independently developed and modified without impacting the entire system. Encapsulation and abstraction principles enhance security and maintainability by hiding internal complexities and providing clear interfaces.

Inheritance enables the creation of subclasses, fostering code reuse and extensibility. Polymorphism allows objects to exhibit different behaviors based on their specific context, enhancing flexibility and adaptability. OOD's emphasis on these principles results in well-structured, maintainable, and scalable code, making it an ideal choice for developing a complex and evolving system like House of Flair.

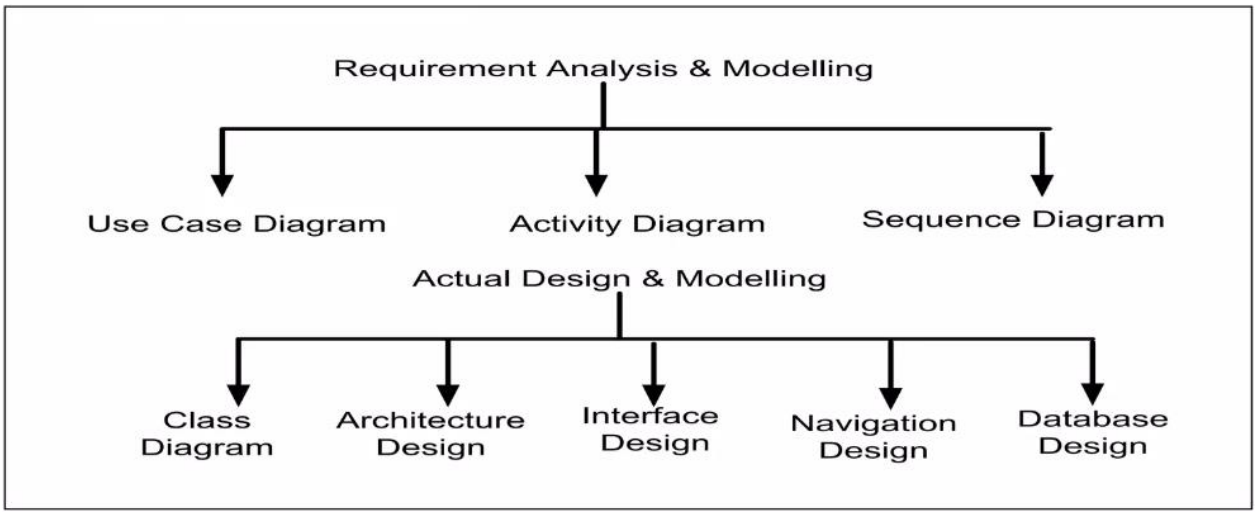


Fig. 3.1 Design Strategy Used

3.2 USE CASE DIAGRAM

Use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and tells how the user handles a system. Purposes of a use case diagram given below:

- It gathers the system's needs.
- It depicts the external view of the system.
- It recognizes the internal as well as external factors that influence the system.
- It represents the interaction between the actors.

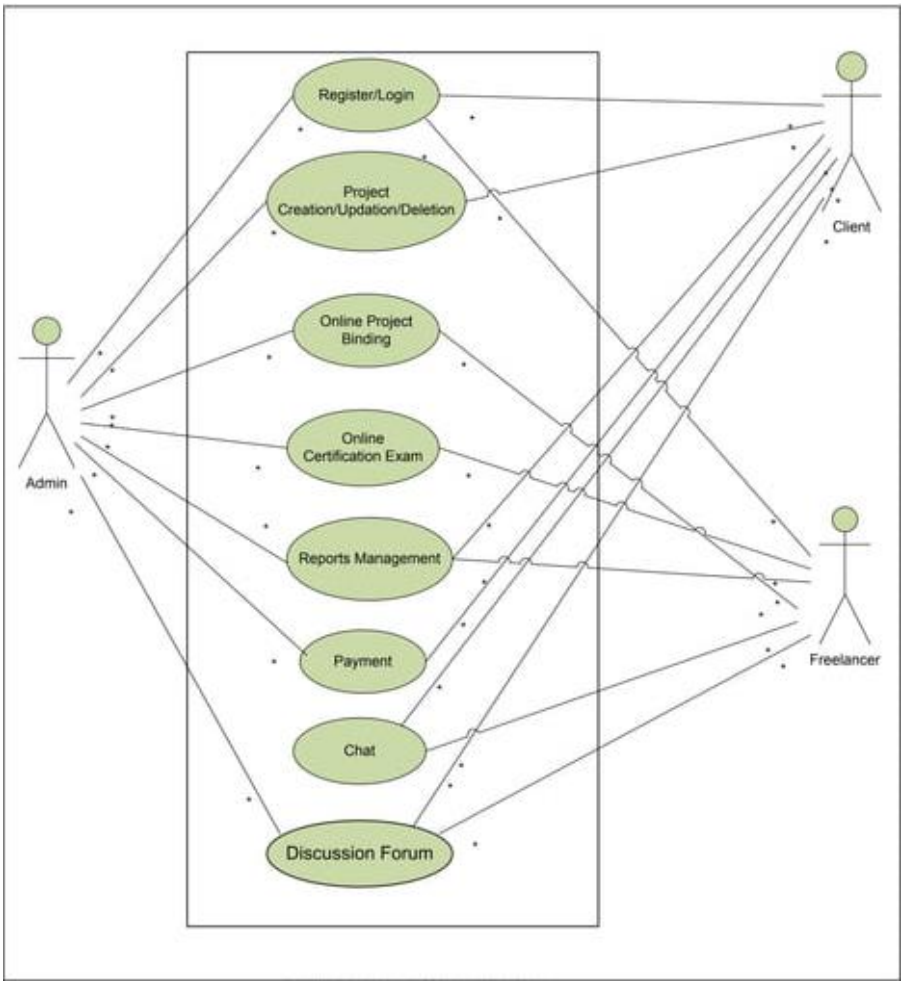


Fig. 3.2 Use Case Diagram

3.3 SEQUENCE DIAGRAM

The design shows a detailed illustration of events sequenced and happening in the Food Ordering System. This designed sequence diagram can show programmers and readers the sequence of messages between the actor and the objects.

As you can see through the illustration, the conditions and interactions are emphasized. These interactions are essential for the Online Food Ordering System development.

The series of messages are shown and labeled to guide you in building the System. You can modify the design if you have more ideas. You can also add more features to this design and use it as your project blueprint.

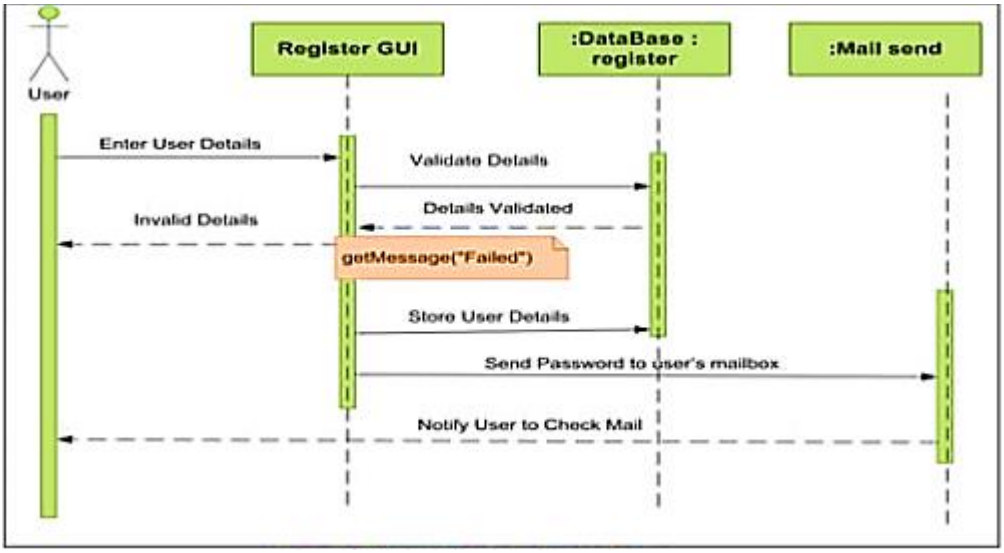


Fig. 3.3 Registration Sequence Diagram

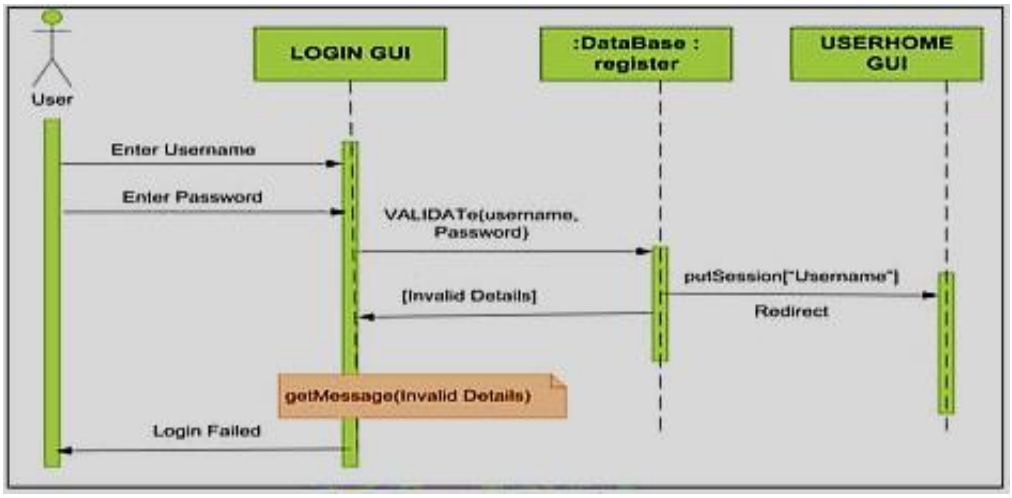


Fig. 3.4 Login Sequence Diagram

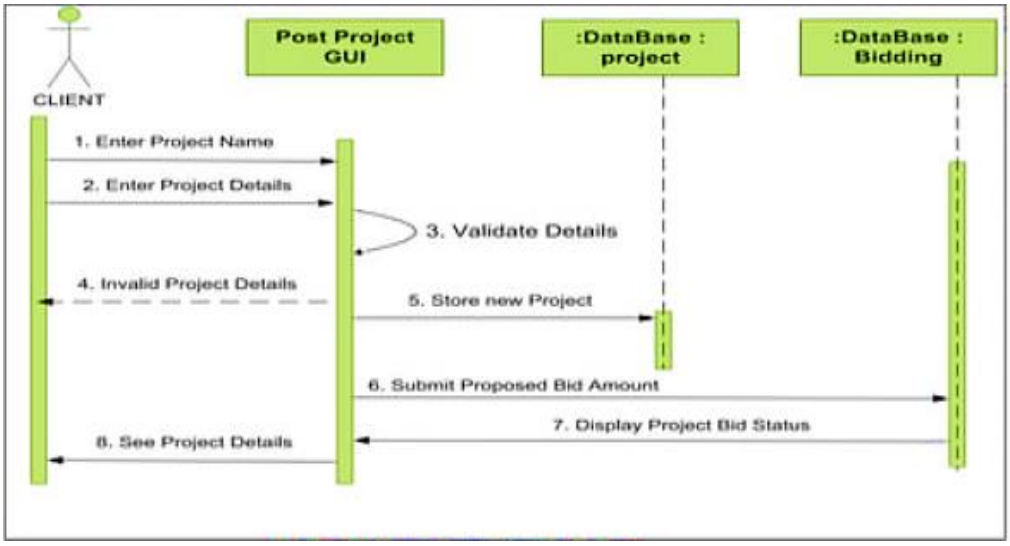


Fig. 3.5 Post Project Sequence Diagram

3.4 ACTIVITY DIAGRAM

An activity diagram is a graphical representation of the flow of activities within a system, process, or workflow. It is part of the Unified Modeling Language (UML) and is commonly used in software development to model the behavior of a system or to describe business processes. Activity diagrams consist of various elements, including:

- Initial node: Represents the starting point of the activity diagram.
- Activity: Represents a specific action or task that occurs within the system or process. It can be represented by a rectangle with rounded corners.
- Decision node: Represents a point in the diagram where a decision or choice needs to be made. It is represented by a diamond-shaped symbol.
- Fork and join nodes: Used to split the flow of activities into parallel paths (fork) or merge parallel paths back into a single flow (join).
- Control flow arrows: Arrows that indicate the sequence and order of activities in the diagram.
- Final node: Represents the end point of the activity diagram.

Activity diagrams provide a visual representation of how activities are organized, their dependencies, and the control flow between them. They help in understanding the overall behavior of a system or process, identifying potential bottlenecks or inefficiencies, and communicating the flow of activities to stakeholders involved in system development or process improvement.

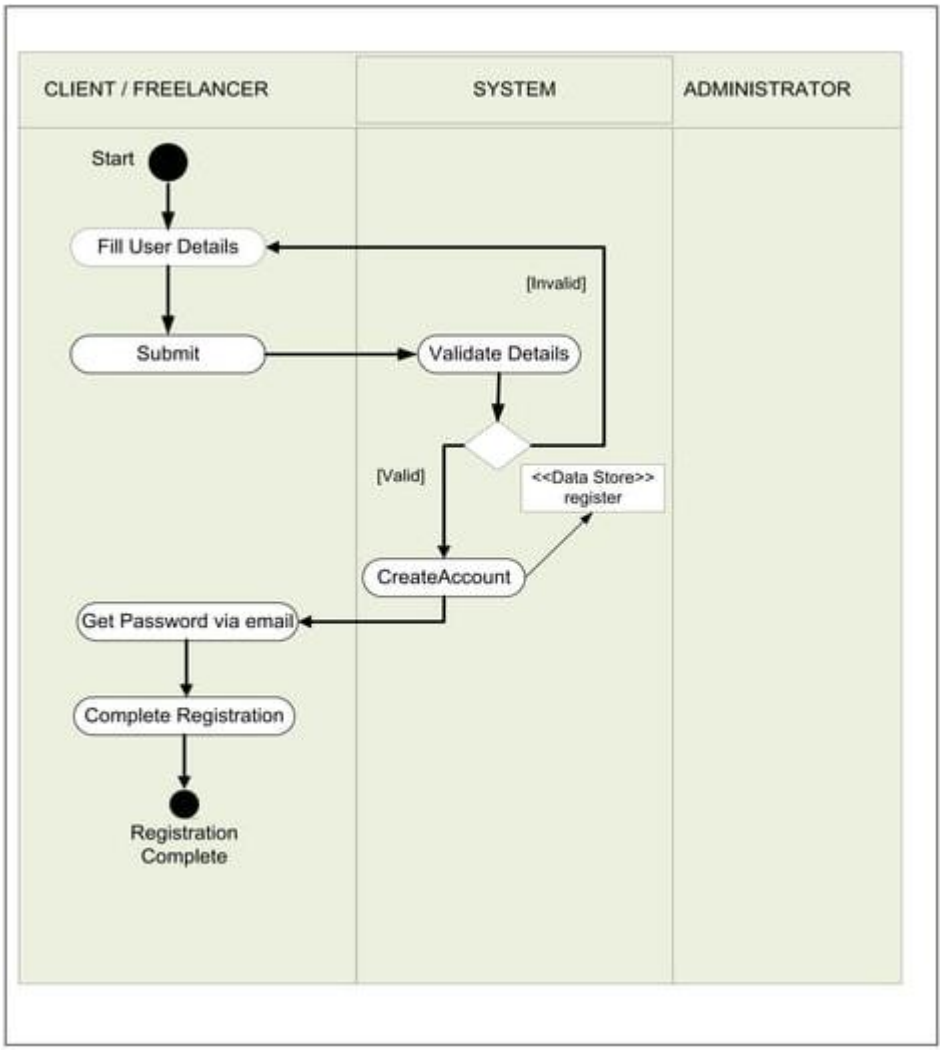


Fig. 3.6 Registration Activity Diagram

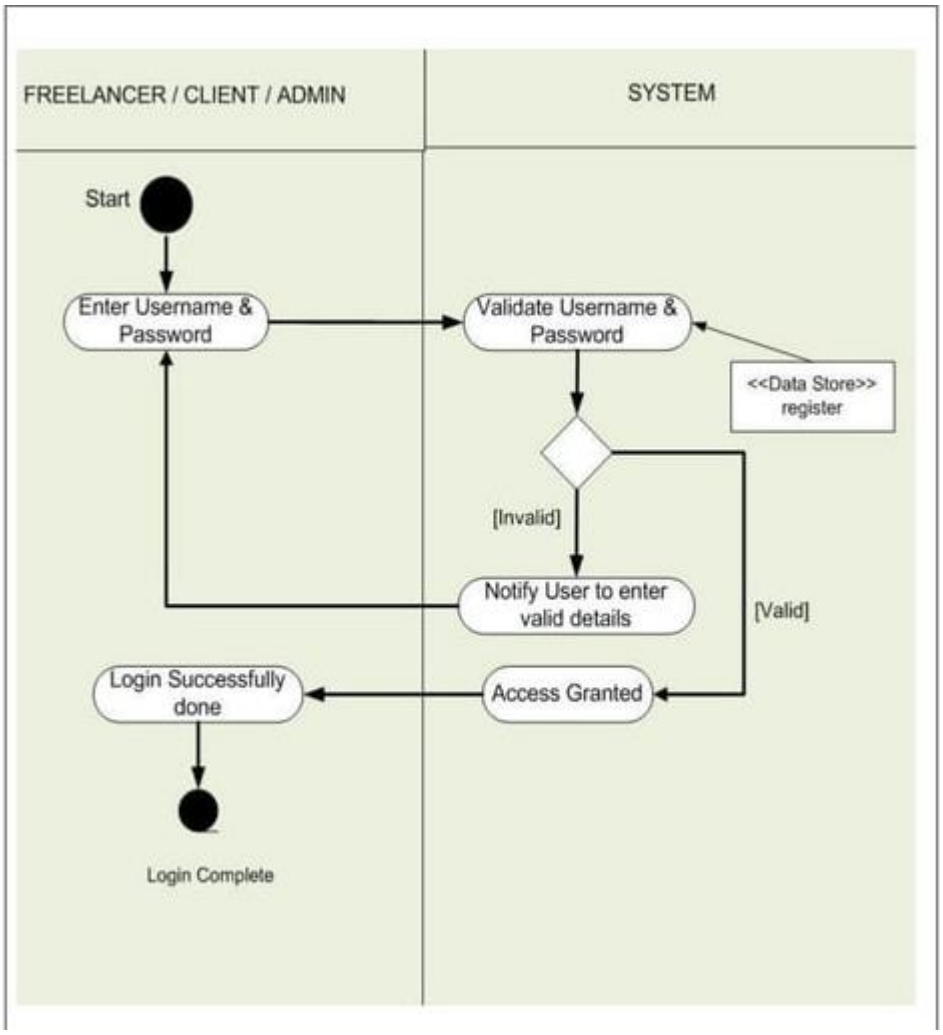


Fig. 3.7 Login Activity Diagram

3.5 ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship Diagram (ERD) is a visual representation of the relationships between entities in a database. It is a popular modeling technique used in database design to illustrate the logical structure of a system and how different entities relate to each other.

In an ERD, entities are represented by rectangles, and the relationships between entities are depicted using lines connecting them. There are three main components in an ERD:

- **Entities:** An entity is a distinct object, concept, or thing in the real world that can be uniquely identified. In an ERD, entities are represented by rectangles, and the entity name is written inside the rectangle. For example, in a database for a university, entities could include "Student," "Course," and "Professor."
- **Attributes:** Attributes are the properties or characteristics of an entity. They provide more details and describe the specific aspects of an entity. Attributes are usually represented as ovals connected to the entity rectangle. For example, attributes of a "Student" entity could include "Student ID," "Name," and "Date of Birth."
- **Relationships:** Relationships represent the associations between entities. They describe how entities are connected or related to each other. Relationships are typically represented by lines connecting the entity rectangles, with labels that describe the nature of the relationship. For example, a relationship between "Student" and "Course" entities could be labeled as "Enrolls In" to indicate that a student can enroll in multiple courses. There are different types of relationships that can be represented in an ERD, including:
    1. **One-to-One (1:1):** A single instance of an entity is associated with only one instance of another entity.
    2. **One-to-Many (1:N):** A single instance of an entity is associated with multiple instances of another entity.
    3. **Many-to-Many (N:N):** Multiple instances of an entity are associated with multiple instances of another entity.

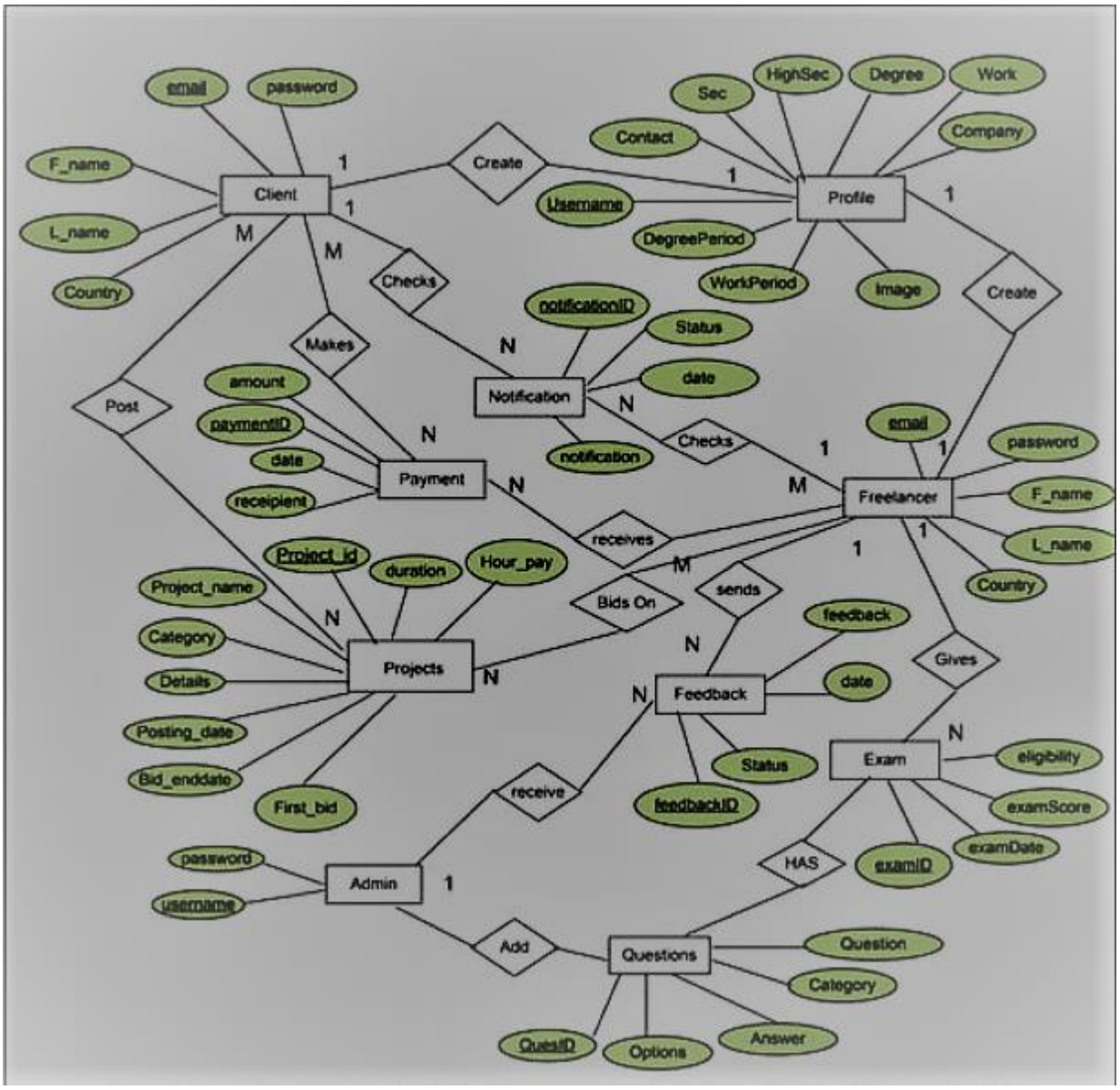


Fig. 3.8 Entity Relationship Diagram

CHAPTER 4

WEBSITE DESIGN

4.1 GUI (Screenshot)

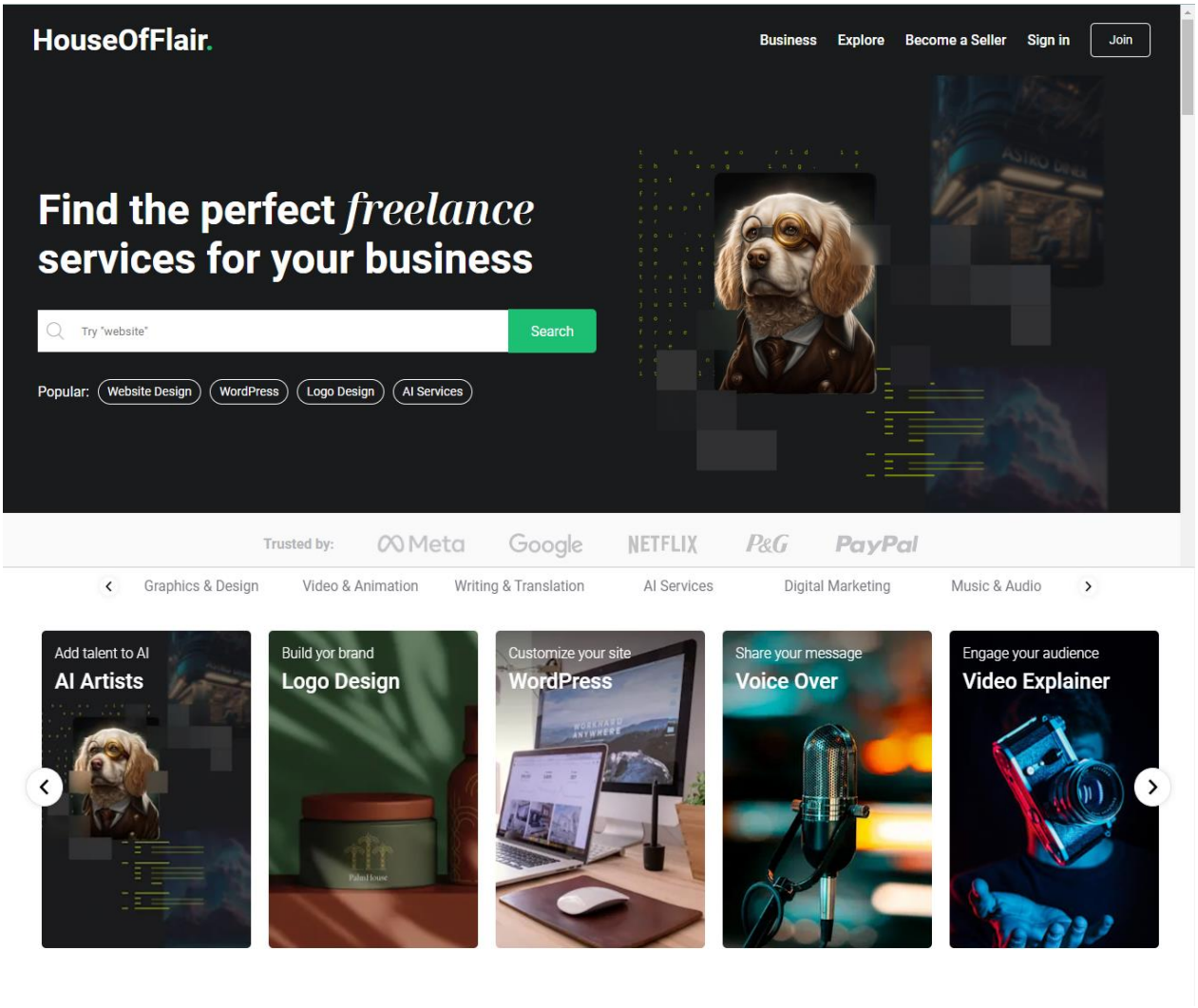
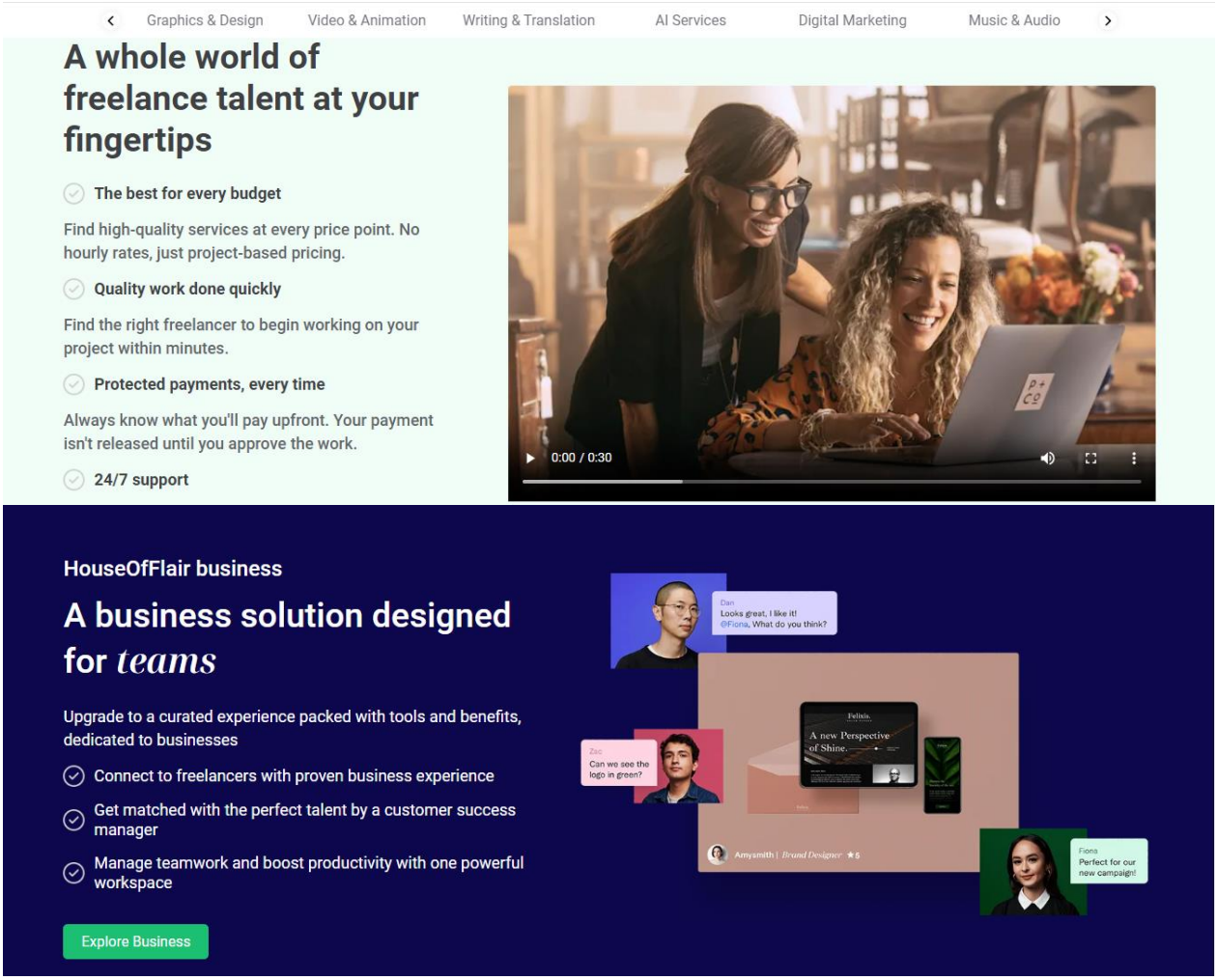


Fig. 4. 1 Home (Landing Page)



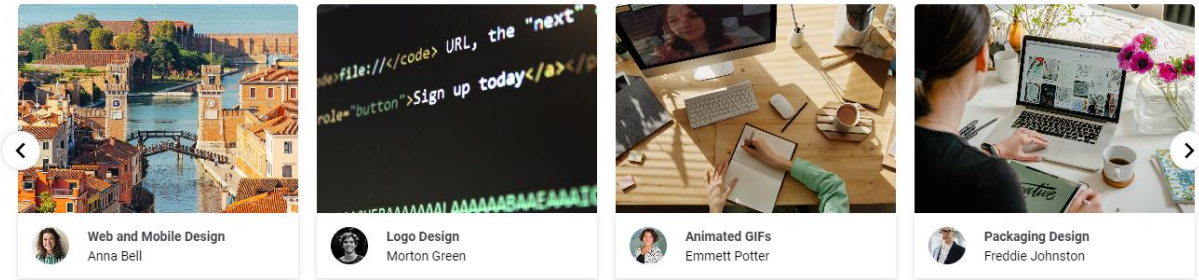


Fig. 4.2 Home (Landing Page) Contd.

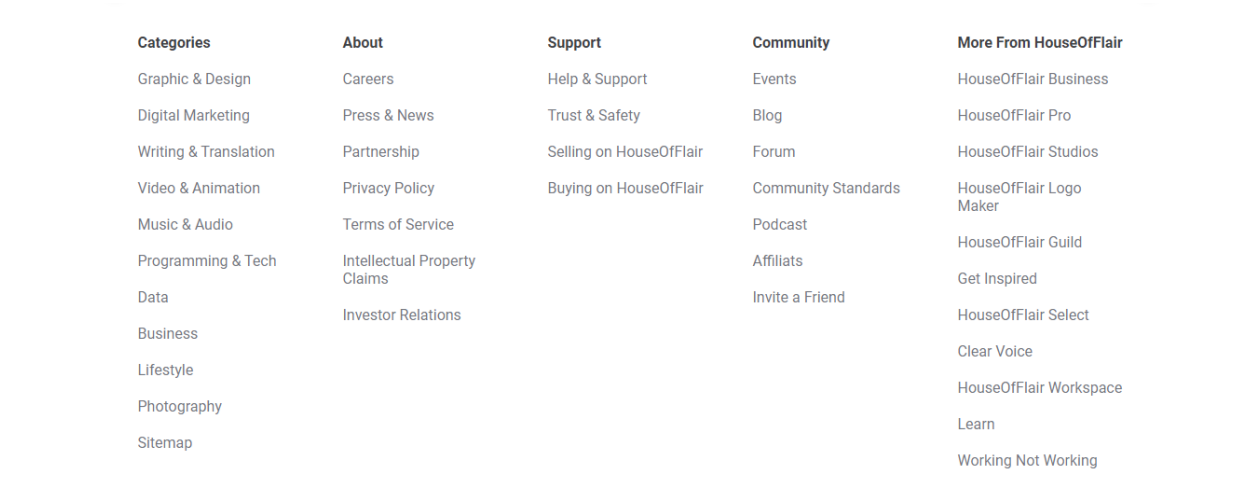


Fig. 4.3 Home (Landing Page) Contd.

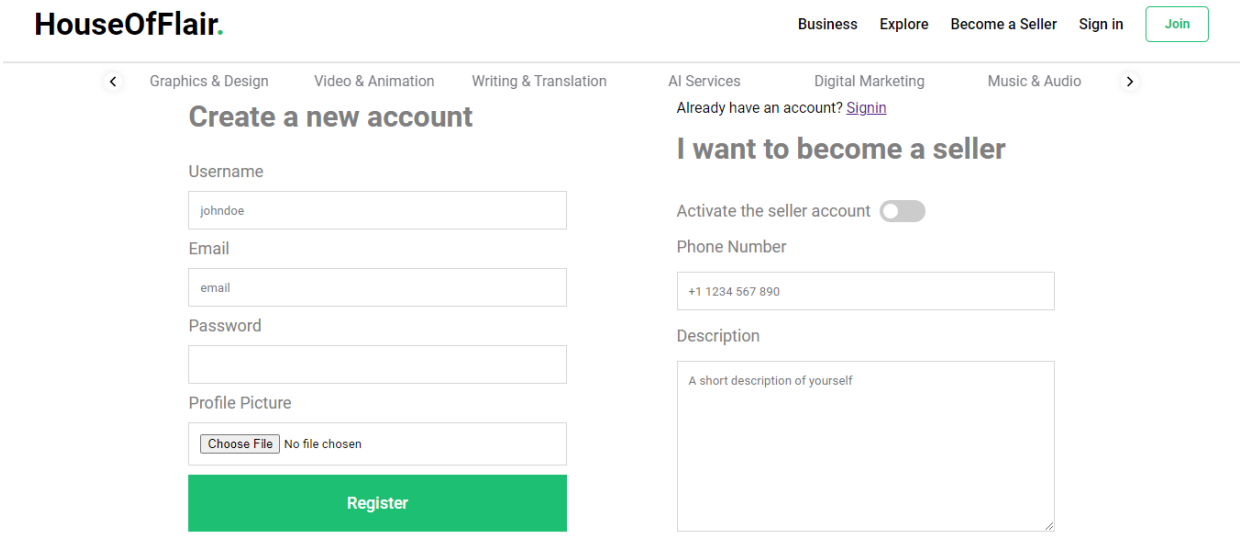


Fig. 4.4 Registration

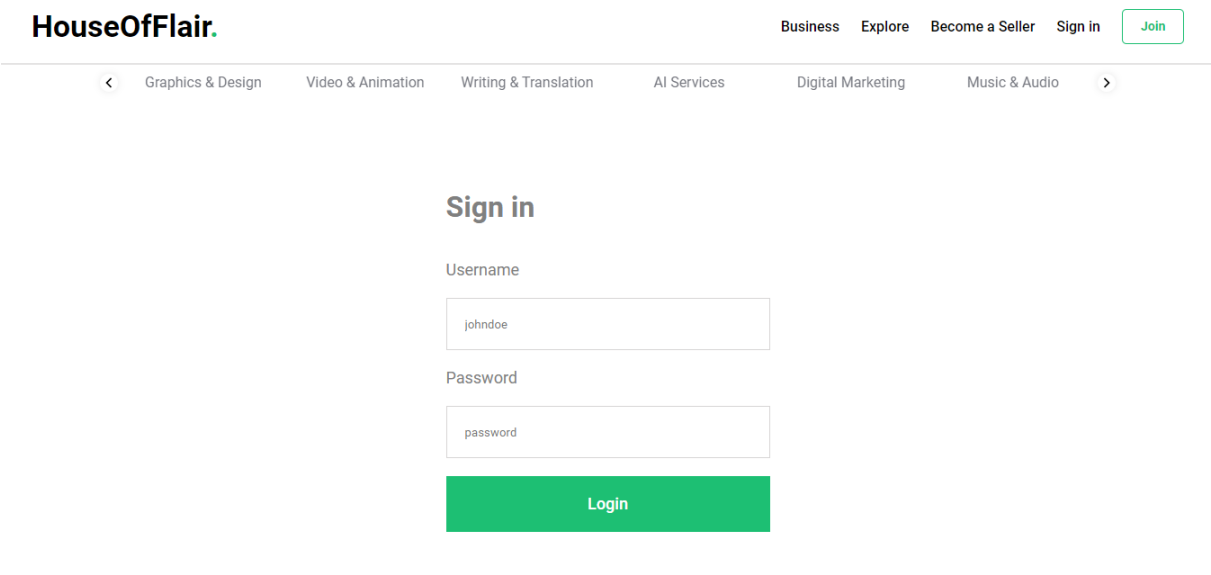


Fig. 4.5 Login

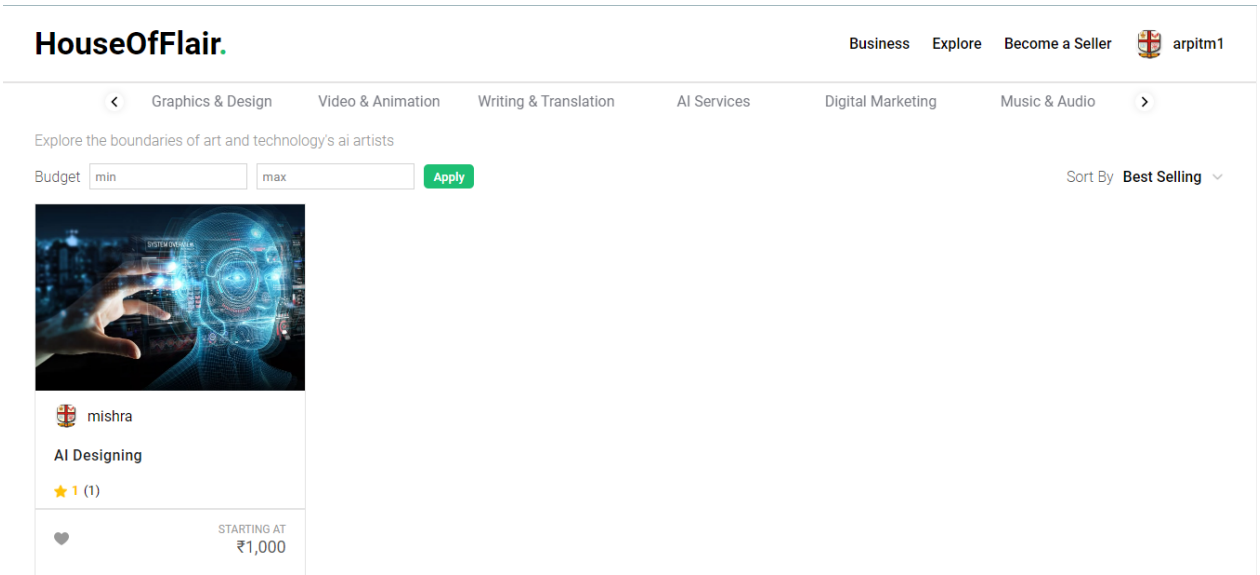
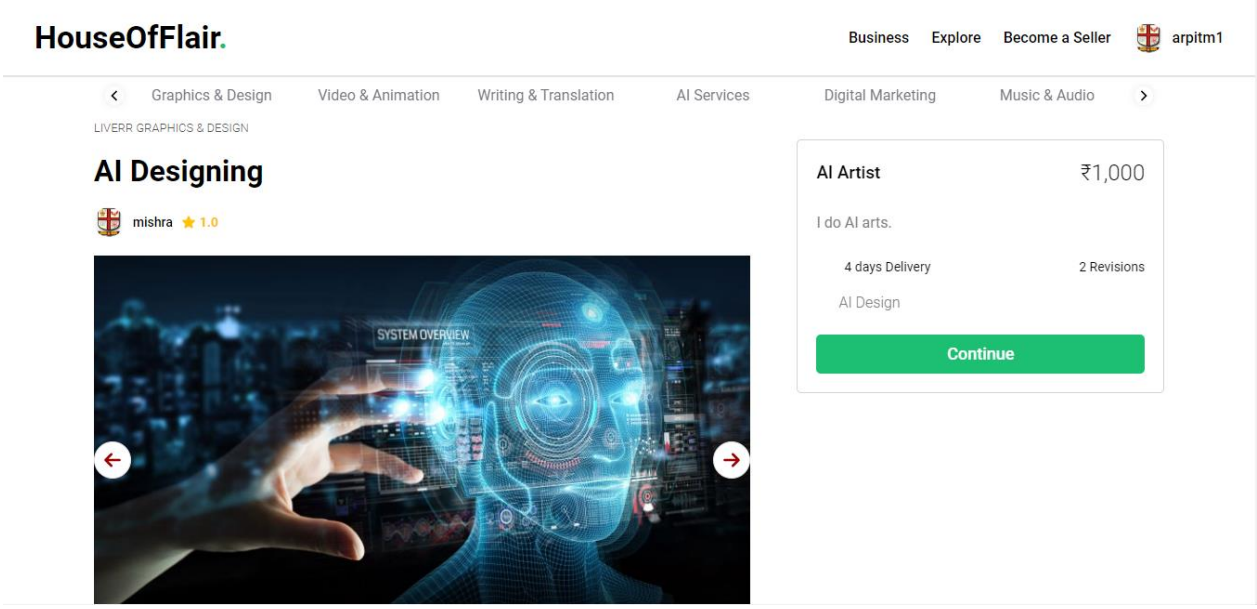


Fig. 4.6 Project (Buyer View)



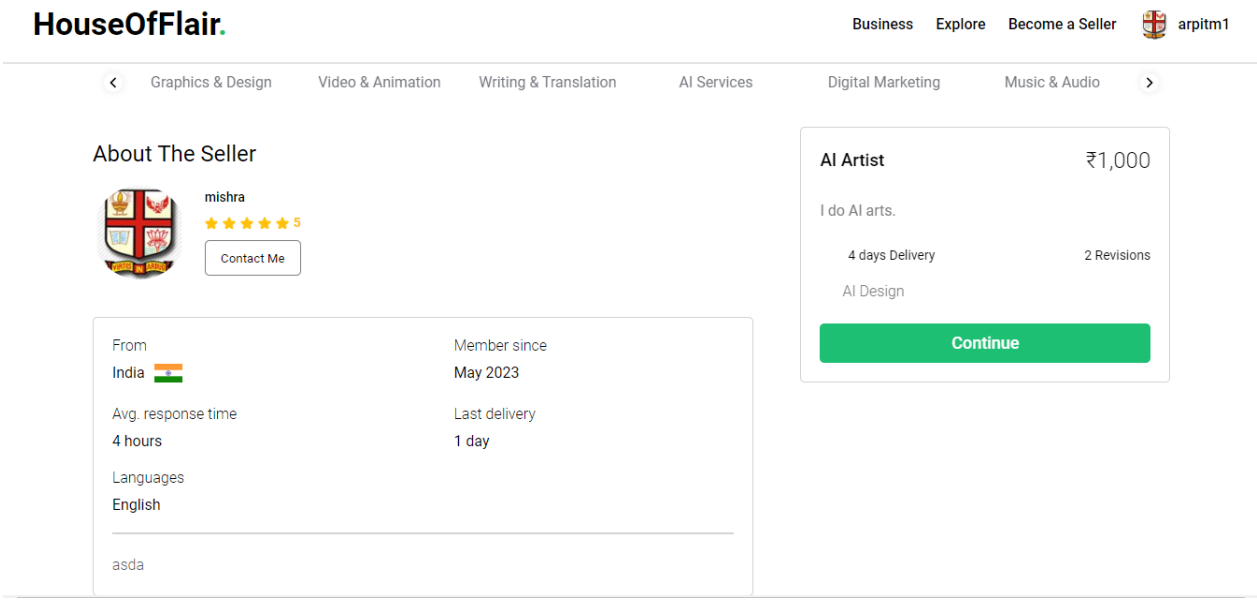


Fig. 4. 7 Item Description (Buyer View) Contd.

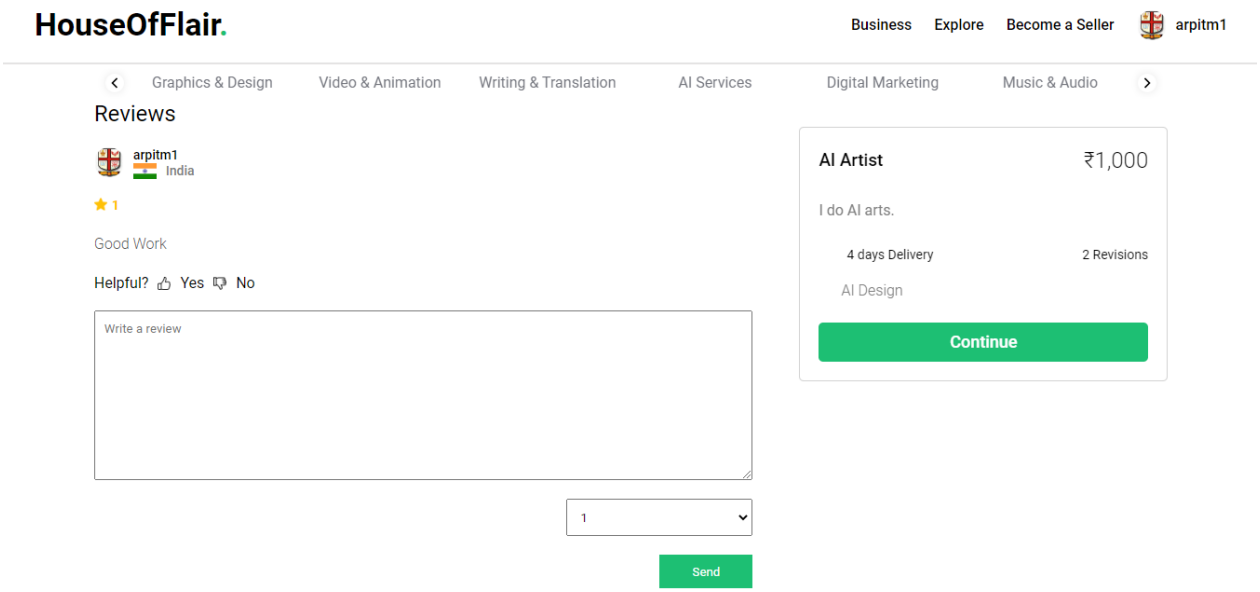


Fig. 4. 8 Item Description (Buyer View) Contd.

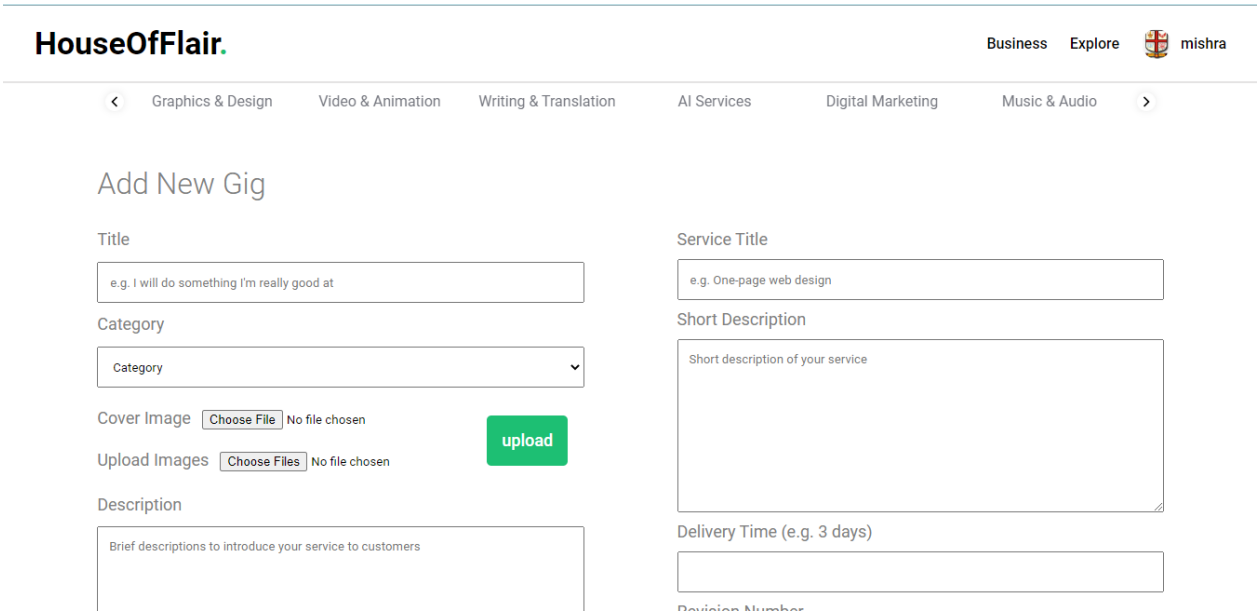


Fig. 4.9 Add Item (Seller View) Contd.

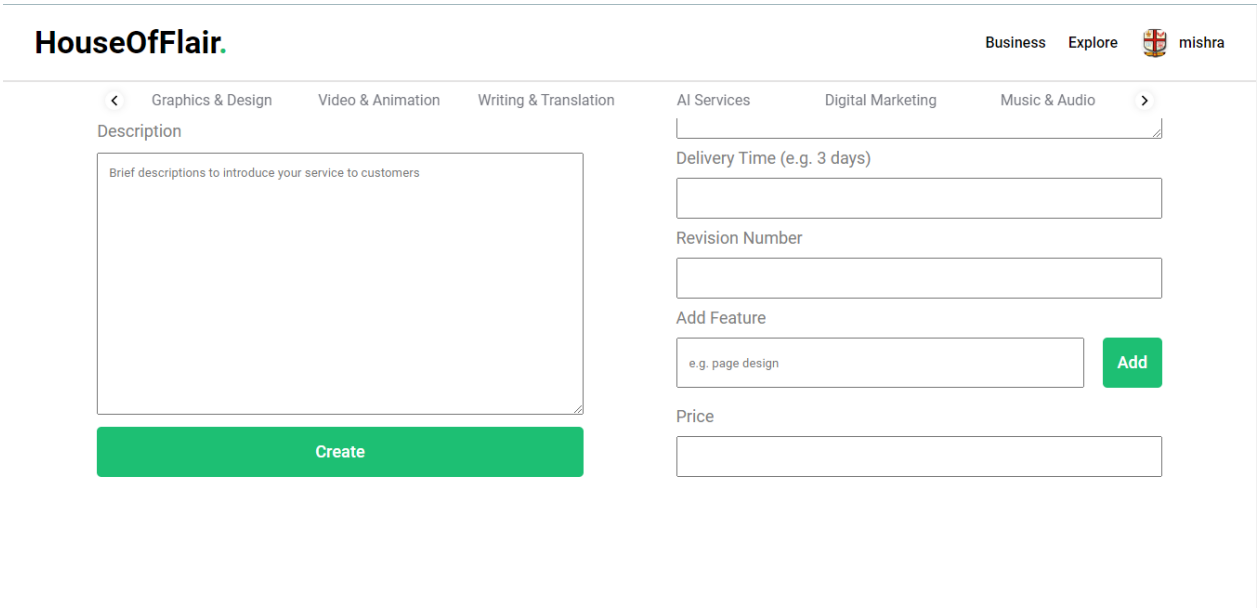


Fig. 4.10 Add Item (Seller View) Contd.

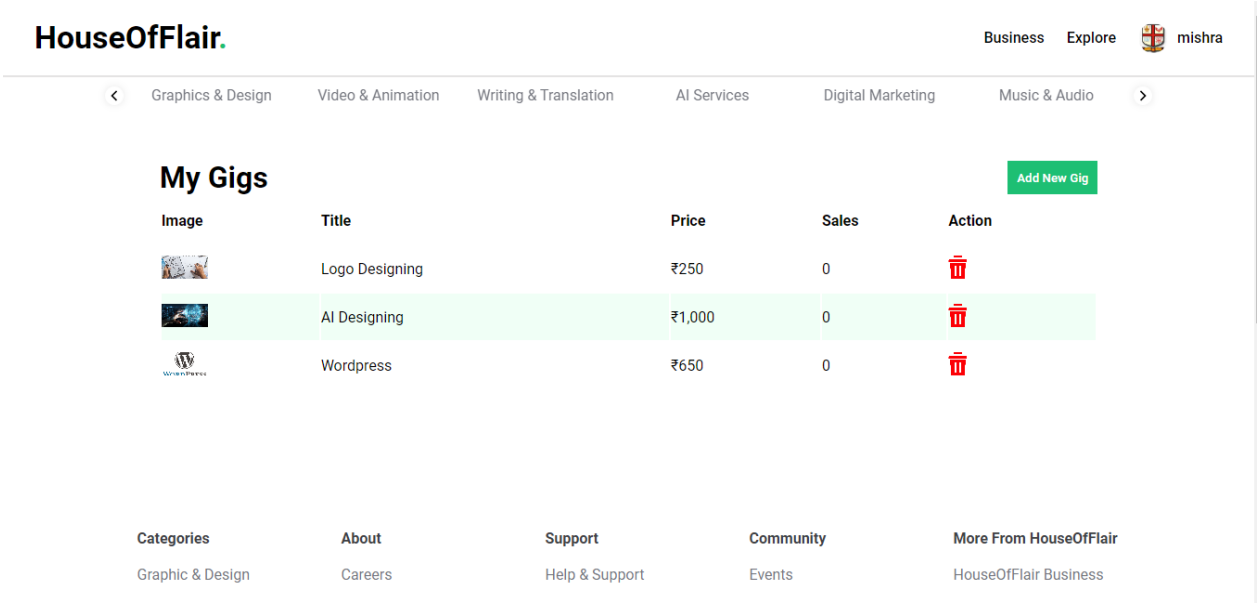


Fig. 4.11 Item List (Seller View)

CHAPTER 5

CODING

App:

```
import { createBrowserRouter, Outlet, RouterProvider } from 'react-router-dom';
import { QueryClient, QueryClientProvider } from '@tanstack/react-query';
import { Toaster } from 'react-hot-toast';

import { Navbar, PrivateRoute } from './components';
import { Home, Footer, Gig, Gigs, MyGigs, Add, Orders, Message, Messages, Login,
Register, Pay, Success, NotFound } from './pages';
import './App.scss';

const paths = [
  { path: '/', element: <Home /> },
  { path: '/gig/:_id', element: <Gig /> },
  { path: '/gigs', element: <Gigs /> },
  { path: '/login', element: <Login /> },
  { path: '/register', element: <Register /> },
  { path: '/orders', element: <PrivateRoute><Orders /></PrivateRoute> },
  { path: '/organize', element: <PrivateRoute><Add /></PrivateRoute> },
  { path: '/my-gigs', element: <PrivateRoute><MyGigs /></PrivateRoute> },
  { path: '/message/:conversationID', element: <PrivateRoute><Message
/></PrivateRoute> },
  { path: '/messages', element: <PrivateRoute><Messages /></PrivateRoute> },
  { path: '/pay/:_id', element: <PrivateRoute><Pay /></PrivateRoute> },
  { path: '/success', element: <PrivateRoute><Success /></PrivateRoute> },
  { path: '*', element: <NotFound /> }
];


function App() {
  const queryClient = new QueryClient();
  const Layout = () => {
    return (
      <QueryClientProvider client={queryClient}>
        <Navbar />
```

```
        <Outlet />
        <Footer />
      </QueryClientProvider>
    )
  }
  const router = createBrowserRouter([
    {
      path: '/',
      element: <Layout />,
      children: paths.map(({ path, element }) => ({ path, element }))
    }
  ])

  return (
    <div className="App">
      <RouterProvider router={router} />
      <Toaster />
    </div>
  )
}

export default App
```

Add:

```
import React, { useEffect, useReducer, useState } from 'react';
import { useQueryClient, useMutation } from '@tanstack/react-query';
import { useNavigate } from 'react-router-dom';
import { gigReducer, initialState } from '../../reducers/gigReducer';
import { cards } from '../../data';
import { axiosFetch, generateImageURL, getCurrentUser } from '../../utils';
import toast from 'react-hot-toast';
import './Add.scss';

const Add = () => {
  const currentUser = getCurrentUser();
  const [state, dispatch] = useReducer(gigReducer, initialState);
  const [coverImage, setCoverImage] = useState(null);
  const [gigImages, setGigImages] = useState([]);
  const [uploading, setUploading] = useState(false);
  const [disabled, setDisabled] = useState(false);
  const navigate = useNavigate();
  const queryClient = useQueryClient();

  useEffect(() => {
    window.scrollTo(0, 0);
```

```
}, [])

const mutation = useMutation({
  mutationFn: (gig) =>
    axiosFetch.post('/gigs', gig)
    .then(({data}) => {
      return data;
    })
    .catch(({response}) => {
      toast.error(response.data.message);
    })
  ,
  onSuccess: () =>
    queryClient.invalidateQueries(['my-gigs'])
})

const handleFormCange = (event) => {
  const { name, value } = event.target;
  dispatch({
    type: 'CHANGE_INPUT',
    payload: { name, value }
  })
}

const handleFormFeature = (event) => {
  event.preventDefault();
  dispatch({
    type: 'ADD_FEATURE',
    payload: event.target[0].value
  })
  event.target.reset();
}

const handleImageUploads = async () => {
  try {
    setUploading(true);
    const cover = await generateImageURL(coverImage);
    const images = await Promise.all(
      [...gigImages].map(async (img) => await generateImageURL(img))
    )
    dispatch({
      type: 'ADD_IMAGES',
      payload: { cover: cover.url, images: images.map((img) => img.url) }
    })
    setUploading(false);
```

```
      setDisabled(true);
    }
    catch (error) {
      console.log(error);
      setUploading(false);
    }
  }

  const handleFormSubmit = (event) => {
    event.preventDefault();
    const form = {...state, userID: currentUser._id}
    for(let key in form) {
      if(form[key] === '' || form[key].length === 0) {
        toast.error('Please fill input field: ' + key);
        return;
      }
    }
    toast.success("Congratulations! You're on the market!")
    mutation.mutate(form);
    setTimeout(() => {
      navigate('/my-gigs');
    }, 2000);
  }

  return (
    <div className='add'>
      <div className="container">
        <h1>Add New Gig</h1>
        <div className="sections">
          <div className="left">
            <label htmlFor="">Title</label>
            <input name='title' type="text" placeholder="e.g. I will do something
I'm really good at" onChange={handleFormCange} />

            <label htmlFor="">Category</label>
            <select name="category" onChange={handleFormCange}>
              <option value=''>Category</option>
              {
                cards.map((item) => (
                  <option key={item.id}
value={item.slug}>{item.slug[0].toUpperCase() + item.slug.slice(1)}</option>
                ))
              }
            </select>
```

```
            <div className="images">
              <div className="imagesInputs">
                <label htmlFor="">Cover Image</label>
                <input type="file" accept='image/*' onChange={(event) =>
setCoverImage(event.target.files[0])} />
                <br />
                <label htmlFor="">Upload Images</label>
                <input type="file" accept='image/*' multiple onChange={(event) =>
setGigImages(event.target.files)} />
              </div>
              <button disabled={!!disabled}
onClick={handleImageUploads}>{uploading ? 'uploading' : disabled ? 'Uploaded' :
'upload'}</button>
            </div>

            <label htmlFor="">Description</label>
            <textarea name='description' cols="30" rows="16" placeholder='Brief
descriptions to introduce your service to customers'
onChange={handleFormCange}></textarea>
            <button onClick={handleFormSubmit}>Create</button>
          </div>

          <div className="right">
            <label htmlFor="">Service Title</label>
            <input type="text" name='shortTitle' placeholder='e.g. One-page web
design' onChange={handleFormCange} />

            <label htmlFor="">Short Description</label>
            <textarea name='shortDesc' cols="30" rows="10" placeholder='Short
description of your service' onChange={handleFormCange}></textarea>

            <label htmlFor="">Delivery Time (e.g. 3 days)</label>
            <input type="number" name='deliveryTime' min='1'
onChange={handleFormCange} />

            <label htmlFor="">Revision Number</label>
            <input type="number" name='revisionNumber' min='1'
onChange={handleFormCange} />

            <label htmlFor="">Add Feature</label>
            <form className='add' onSubmit={handleFormFeature}>
              <input type="text" placeholder='e.g. page design'
onChange={handleFormCange} />
              <button type='submit'>Add</button>
            </form>
```

```
            <div className="addedFeatures">
              {
                state.features?.map((feature) => (
                  <div key={feature} className="item">
                    <button onClick={() => dispatch({ type: 'REMOVE_FEATURE',
payload: feature })}>{feature}
                      <span>X</span>
                    </button>
                  </div>
                ))
              }
            </div>
            <label htmlFor="">Price</label>
            <input name='price' type="number" min='1' onChange={handleFormCange}
/>
          </div>
        </div>
      </div>
    </div>
  )
}

export default Add
```

Login:

```
import React, { useEffect, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { axiosFetch } from '../../../utils';
import toast from 'react-hot-toast';
import './Login.scss';

const initialState = {
  username: '',
  password: ''
}

const Login = () => {
  const [formInput, setFormInput] = useState(initialState);
  const [error, setError] = useState(null);
  const navigate = useNavigate();

  useEffect(() => {
    window.scrollTo(0, 0)
  }, [])
```

```
  const handleFormInput = (event) => {
    const { value, name } = event.target;
    setFormInput({
      ...formInput,
      [name]: value
    });
  }

  const handleFormSubmit = async (event) => {
    event.preventDefault();
    try {
      const { data } = await axiosFetch.post('/auth/login', formInput);
      localStorage.setItem('currentUser', JSON.stringify(data.user));
      toast.success("Welcome back!", {
        duration: 3000,
        icon: "😃"
      });
      navigate('/');
    }
    catch ({ response: { data } }) {
      setError(data.message);
      toast.error("Invalid Credentials", {
        duration: 3000,
      });
    }
  }

  return (
    <div className='login'>
      <form action="" onSubmit={handleFormSubmit}>
        <h1>Sign in</h1>
        <label htmlFor="">Username</label>
        <input name='username' placeholder='johndoe' onChange={handleFormInput}
/>

        <label htmlFor="">Password</label>
        <input name='password' type='password' placeholder='password'
onChange={handleFormInput} />
        <button type='submit'>Login</button>
        <span>{error && error}</span>
      </form>
    </div>
  )
}
```

```
export default Login
```

Register:

```
import { useEffect, useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { axiosFetch, generateImageURL } from '../../../utils';
import toast from 'react-hot-toast';
import './Register.scss'

const Register = () => {
  const navigate = useNavigate();
  const [image, setImage] = useState(null);
  const [loading, setLoading] = useState(false);
  const [formInput, setFormInput] = useState({
    username: "",
    email: "",
    password: "",
    phone: '',
    description: '',
    isSeller: false,
  });

  useEffect(() => {
    window.scrollTo(0, 0)
  }, [])

  const handleSubmit = async (event) => {
    event.preventDefault();

    for (let key in formInput) {
      if (formInput[key] === '') {
        toast.error('Please fill all input field: ' + key);
        return;
      }
      else if (key === 'phone' && formInput[key].length < 9) {
        toast.error('Enter valid phone number!');
        return;
      }
    }

    setLoading(true);
    try {
```

```
      const { url } = await generateImageURL(image);
      const { data } = await axiosFetch.post('/auth/register', { ...formInput,
image: url });
      toast.success('Registration successful!');
      setLoading(false);
      navigate('/login');
    }
    catch ({ response }) {
      toast.error(response.data.message);
      setLoading(false);
    }
  }

  const handleChange = (event) => {
    const { value, name, type, checked } = event.target;
    const inputValue = type === 'checkbox' ? checked : value;
    setFormInput({
      ...formInput,
      [name]: inputValue
    });
  }

  return (
    <div className="register">
      <form onSubmit={handleSubmit}>
        <div className="left">
          <h1>Create a new account</h1>
          <label htmlFor="">Username</label>
          <input
            name="username"
            type="text"
            placeholder="johndoe"
            onChange={handleChange}
          />
          <label htmlFor="">Email</label>
          <input
            name="email"
            type="email"
            placeholder="email"
            onChange={handleChange}
          />
          <label htmlFor="">Password</label>
          <input name="password" type="password" onChange={handleChange} />
          <label htmlFor="">Profile Picture</label>
```

```
          <input type="file" onChange={(event) =>
setImage(event.target.files[0])} />
          <button type="submit" disabled={loading}>{loading ? 'Loading...' :
'Register'}</button>
        </div>
        <div className="right">
          <p>Already have an account? <Link to='/login'>Signin</Link></p>
          <h1>I want to become a seller</h1>
          <div className="toggle">
            <label htmlFor="">Activate the seller account</label>
            <label className="switch">
              <input type="checkbox" name='isSeller' onChange={handleChange} />
              <span className="slider round"></span>
            </label>
          </div>
          <label htmlFor="">Phone Number</label>
          <input
            name="phone"
            type="text"
            placeholder="+1 1234 567 890"
            onChange={handleChange}
          />
          <label htmlFor="">Description</label>
          <textarea
            placeholder="A short description of yourself"
            name="description"
            id=""
            cols="30"
            rows="10"
            onChange={handleChange}
          ></textarea>
        </div>
      </form>
    </div>
  )
}

export default Register
```

Footer:

```
import React, { useEffect } from 'react';
import './Footer.scss';

const Footer = () => {
```

```
useEffect(() => {
  window.scrollTo(0, 0)
}, [])

return (
  <div className='footer'>
    <div className="container">
      <div className="top">
        <div className="item">
          <h1>Categories</h1>
          <span>Graphic & Design</span>
          <span>Digital Marketing</span>
          <span>Writing & Translation</span>
          <span>Video & Animation</span>
          <span>Music & Audio</span>
          <span>Programming & Tech</span>
          <span>Data</span>
          <span>Business</span>
          <span>Lifestyle</span>
          <span>Photography</span>
          <span>Sitemap</span>
        </div>
        <div className="item">
          <h1>About</h1>
          <span>Careers</span>
          <span>Press & News</span>
          <span>Partnership</span>
          <span>Privacy Policy</span>
          <span>Terms of Service</span>
          <span>Intellectual Property Claims</span>
          <span>Investor Relations</span>
        </div>
        <div className="item">
          <h1>Support</h1>
          <span>Help & Support</span>
          <span>Trust & Safety</span>
          <span>Selling on HouseOfFlair</span>
          <span>Buying on HouseOfFlair</span>
        </div>
        <div className="item">
          <h1>Community</h1>
          <span>Events</span>
          <span>Blog</span>
          <span>Forum</span>
          <span>Community Standards</span>
```

```
            <span>Podcast</span>
            <span>Affiliats</span>
            <span>Invite a Friend</span>
          </div>
          <div className="item">
            <h1>More From HouseOfFlair</h1>
            <span>HouseOfFlair Business</span>
            <span>HouseOfFlair Pro</span>
            <span>HouseOfFlair Studios</span>
            <span>HouseOfFlair Logo Maker</span>
            <span>HouseOfFlair Guild</span>
            <span>Get Inspired</span>
            <span>HouseOfFlair Select</span>
            <span>Clear Voice</span>
            <span>HouseOfFlair Workspace</span>
            <span>Learn</span>
            <span>Working Not Working</span>
          </div>
        </div>
        <hr />
        <div className="bottom">
          <div className="left">
            <h2>HouseOfFlair</h2>
            <span>© HouseOfFlair International Ltd. {new
Date().getFullYear()}</span>
          </div>
          <div className="right">
            <div className="social">
              <img src="./media/twitter.png" alt="" />
              <img src="./media/facebook.png" alt="" />
              <img src="./media/linkedin.png" alt="" />
              <img src="./media/pinterest.png" alt="" />
              <img src="./media/instagram.png" alt="" />
            </div>
            <div className="link">
              <img src="./media/language.png" alt="" />
              <span>English</span>
            </div>
            <div className="link">
              <img src="./media/coin.png" alt="" />
              <span>USD</span>
            </div>
            <div className="link">
              <img src="./media/accessibility.png" alt="" />
              <span>USD</span>
```

```
            </div>
          </div>
        </div>
      </div>
    </div>
  )
}

export default Footer
```

Gig:

```
import { useState, useRef, useEffect } from 'react';
import { GigCard } from '../../components';
import { useQuery } from "@tanstack/react-query";
import { useLocation } from 'react-router-dom';
import { axiosFetch } from '../../utils';
import './Gigs.scss';

const Gigs = () => {
  const [openMenu, setOpenMenu] = useState(false);
  const [sortBy, setSortBy] = useState('sales');
  const [category, setCategory] = useState('.');
  const minRef = useRef();
  const maxRef = useRef();
  const { search } = useLocation();

  useEffect(() => {
    window.scrollTo(0, 0);
  }, []);


  const { isLoading, error, data, refetch } = useQuery({
    queryKey: ['gigs'],
    queryFn: () =>
      axiosFetch.get(`/gigs${search}&min=${minRef.current.value}&max=${maxRef.cur
rent.value}&sort=${sortBy}`)
        .then(({ data }) => {
          setCategory(data[0].category);
          return data;
        })
        .catch((error) => {
          console.log(error);
        })
  });
```

```
  useEffect(() => {
    refetch();
  }, [sortBy, search]);

  const handleSortBy = (type) => {
    setSortBy(type);
    setOpenMenu(false);
    refetch();
  }

  const handlePriceFilter = () => {
    refetch();
  }

  return (
    <div className='gigs'>
      <div className="container">
        <span className="breadcrumbs">HouseOfFlair {category[0]?.toUpperCase() +
category.slice(1)}</span>
        <h1>{category[0]?.toUpperCase() + category.slice(1)}</h1>
        <p>Explore the boundaries of art and technology's {category} artists</p>
        <div className="menu">
          <div className="left">
            <span>Budget</span>
            <input ref={minRef} type="number" placeholder='min' />
            <input ref={maxRef} type="number" placeholder='max' />
            <button onClick={handlePriceFilter}>Apply</button>
          </div>
          <div className="right">
            <span className='sortBy'>Sort By</span>
            <span className='sortType'>{sortBy === 'sales' ? 'Best Selling' :
'Newest'}</span>
            <img src="./media/down.png" alt="" onClick={() =>
setOpenMenu(!openMenu)} />
            {
              openMenu && (<div className="rightMenu">
                {
                  sortBy === 'sales' ? <span onClick={() =>
handleSortBy('createdAt')}>Newest</span>
                    : <span onClick={() => handleSortBy('sales')}>Best Selling
</span>
                }
              </div>)
            }
          </div>
```

```
        </div>
        <div className="cards">
          {
            isLoading
              ? '...loading'
              : error
                ? 'Something went wrong!'
                : data.map((gig) => <GigCard key={gig._id} data={gig} />)
          }
        </div>
      </div>
    </div>
  )
}

export default Gigs
```

Home:

```
import React, { useEffect } from 'react';
import { Featured, Slide, TrustedBy } from '../../components';
import { CategoryCard, ProjectCard } from '../../components';
import { cards, projects } from '../../data';

import './Home.scss';

const Home = () => {

  useEffect(() => {
    window.scrollTo(0, 0)
  }, [])

  return (
    <div className='home'>
      <Featured />
      <TrustedBy />
      <Slide slidesToShow={5}>
        {
          cards.map((card) => (
            <CategoryCard key={card.id} data={card} />
          ))
        }
      </Slide>
      <div className="features">
        <div className="container">
          <div className="item">
```

```
            <h1>A whole world of freelance talent at your fingertips</h1>
            <div className="title">
              <img src="./media/check.png" alt="check" />
              <h6>The best for every budget</h6>
            </div>
            <p>Find high-quality services at every price point. No hourly rates,
just project-based pricing.</p>
            <div className="title">
              <img src="./media/check.png" alt="check" />
              <h6>Quality work done quickly</h6>
            </div>
            <p>Find the right freelancer to begin working on your project within
minutes.</p>
            <div className="title">
              <img src="./media/check.png" alt="check" />
              <h6>Protected payments, every time</h6>
            </div>
            <p>Always know what you'll pay upfront. Your payment isn't released
until you approve the work.</p>
            <div className="title">
              <img src="./media/check.png" alt="check" />
              <h6>24/7 support</h6>
            </div>
            <p>Questions? Our round-the-clock support team is available to help
anytime, anywhere.</p>
          </div>
          <div className="item">
            <video poster='https://fiverr-
res.cloudinary.com/q_auto,f_auto,w_700,dpr_1.0/v1/attachments/generic_asset/asset
/089e3bb9352f90802ad07ad9f6a4a450-1599517407052/selling-proposition-still-1400-
x1.png' src="./media/video.mp4" controls></video>
          </div>
        </div>
      </div>

      {/*  Business Component */}
      <div className="features dark">
        <div className="container">
          <div className="item">
            <h2>HouseOfFlair business</h2>
            <h1>A business solution designed for <span>teams</span></h1>
            <p>Upgrade to a curated experience packed with tools and benefits,
dedicated to businesses</p>
            <div className="title">
              <img src="./media/check.png" alt="check" />
```

```
              <h6>Connect to freelancers with proven business experience</h6>
            </div>
            <div className="title">
              <img src="./media/check.png" alt="check" />
              <h6>Get matched with the perfect talent by a customer success
manager</h6>
            </div>
            <div className="title">
              <img src="./media/check.png" alt="check" />
              <h6>Manage teamwork and boost productivity with one powerful
workspace</h6>
            </div>
            <button>Explore Business</button>
          </div>
          <div className="item">
            <img src="https://fiverr-
res.cloudinary.com/q_auto,f_auto,w_870,dpr_1.0/v1/attachments/generic_asset/asset
/d9c17ceebda44764b591a8074a898e63-1599597624757/business-desktop-870-x1.png"
alt="" />
          </div>
        </div>
      </div>

      <Slide slidesToShow={4}>
        {
          projects.map((card) => (
            <ProjectCard key={card.id} data={card} />
          ))
        }
      </Slide>
    </div>
  )
}

export default Home
```

Message:

```
import React, { useEffect } from "react";
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query';
import { axiosFetch } from '../../utils';
import { Link, useParams } from "react-router-dom";
import toast from 'react-hot-toast';
import "./Message.scss";

const Message = () => {
```

```
const currentUser = JSON.parse(localStorage.getItem('currentUser')) || {};
const { conversationID } = useParams();

useEffect(() => {
  window.scrollTo(0, 0)
}, [])

const { isLoading, error, data } = useQuery({
  queryKey: ['messages'],
  queryFn: () =>
    axiosFetch.get(`/messages/${conversationID}`)
      .then(({ data }) => {
        return data;
      })
      .catch(({ response }) => {
        toast.error(response.data.message)
      })
});

const queryClient = useQueryClient();
const mutation = useMutation({
  mutationFn: (message) =>
    axiosFetch.post('/messages', message)
  ,
  onSuccess: () =>
    queryClient.invalidateQueries(['messages'])
})

const handleMessageSubmit = (event) => {
  event.preventDefault();

  mutation.mutate({
    conversationID,
    description: event.target[0].value
  });

  event.target.reset();
}

return (
  <div className="message">
    <div className="container">
      <span className="breadcrumbs">
        <Link to="/messages" className="link">Messages</Link>
      </span>
```

```
        {
          isLoading
            ? '...loading'
            : error
              ? 'Something went wrong'
              : <div className="messages">
                {
                  data.map((message) => (
                    <div className={message.userID._id === currentUser._id ?
'owner item' : 'item'} key={message._id}>
                      <img
                        src={message.userID.image || '/media/noavatar.png'}
                        alt=""
                      />
                      <p>
                        {message.description}
                      </p>
                    </div>
                  ))
                }
              </div>
        }
        <hr />
        <form className="write" onSubmit={handleMessageSubmit}>
          <textarea cols="30" rows="10" placeholder="Write a message"></textarea>
          <button type='submit'>Send</button>
        </form>
      </div>
    </div>
  );
};

export default Message;
```

Messages:

```
import React, { useEffect } from 'react';
import { Link } from 'react-router-dom';
import { useQueryClient, useMutation, useQuery } from '@tanstack/react-query';
import { axiosFetch } from '../../utils';
import moment from 'moment';
import './Messages.scss';

const Messages = () => {
  const currentUser = JSON.parse(localStorage.getItem('currentUser')) || {};
  const queryClient = useQueryClient();
```

```
useEffect(() => {
  window.scrollTo(0, 0)
}, [])

const { isLoading, error, data } = useQuery({
  queryKey: ['conversations'],
  queryFn: () =>
    axiosFetch.get('/conversations')
      .then(({ data }) => {
        return data;
      })
      .catch(({ response }) => {
        console.log(response);
      })
})

const mutation = useMutation({
  mutationFn: (id) =>
    axiosFetch.patch(`/conversations/${id}`)
  ,
  onSuccess: () =>
    queryClient.invalidateQueries(['conversations'])
})

const handleMessageRead = (id) => {
  mutation.mutate(id);
}

return (
  <div className='messages'>
    <div className="container">
      <div className="title">
        <h1>Messages</h1>
      </div>
      {
        isLoading
          ? '...loading'
          : error
            ? 'Something went wrong!'
            : <table>
              <thead>
                <tr>
                  <th>{currentUser.isSeller ? 'Buyer' : 'Seller'}</th>
                  <th>Last Message</th>
```

```
                    <th>Date</th>
                    <th>Action</th>
                  </tr>
                </thead>
                <tbody>
                  {
                    data.map((conv) => (
                      <tr key={conv._id} className={((currentUser.isSeller &&
!conv.readBySeller) || (!currentUser.isSeller && !conv.readByBuyer)) &&
                        "active" || ''}>
                        <td>{currentUser.isSeller ? conv.buyerID.username :
conv.sellerID.username}</td>
                        <td>
                          <Link className='link'
to={`/message/${conv.conversationID}`}>
                            {conv?.lastMessage?.slice(0, 100)}...
                          </Link>
                        </td>
                        <td>{moment(conv.updatedAt).fromNow()}</td>
                        <td>
                          {
                            ((currentUser.isSeller && !conv.readBySeller) ||
(!currentUser.isSeller && !conv.readByBuyer)) &&
                            (<button onClick={() =>
handleMessageRead(conv.conversationID)}>Mark as read</button>)
                          }
                        </td>
                      </tr>
                    ))
                  }
                </tbody>
              </table>
        }
      </div>
    </div>
  )
}

export default Messages
```

MyGigs:

```
import React, { useEffect } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'
import { axiosFetch, getCurrentUser } from '../../utils';
```

```
import toast from 'react-hot-toast';
import './MyGigs.scss';

const MyGigs = () => {
  const currentUser = getCurrentUser();
  const navigate = useNavigate();

  const queryClient = useQueryClient();
  const { isLoading, error, data } = useQuery({
    queryKey: ['my-gigs'],
    queryFn: () =>
      axiosFetch(`/gigs?userID=${currentUser._id}`)
        .then(({ data }) => {
          console.table(data)
          return data;
        })
        .catch(({ response }) => {
          console.log(response.data);
        })
  });

  const mutation = useMutation({
    mutationFn: (_id) =>
      axiosFetch.delete(`/gigs/${_id}`)
    ,
    onSuccess: () =>
      queryClient.invalidateQueries(['my-gigs'])
  });

  const handleGigDelete = (gig) => {
    mutation.mutate(gig._id);
    toast.success(gig.title + ' deleted successfully!');
  }

  useEffect(() => {
    window.scrollTo(0, 0)
  }, [])

  return (
    <div className='myGigs'>
      {
        isLoading
          ? '...loading'
          : error
            ? 'Something went wrong'
```

```
            : <div className="container">
              <div className="title">
                <h1>My Gigs</h1>
                <Link to='/organize' className='link'>
                  <button>Add New Gig</button>
                </Link>
              </div>
              <table>
                <thead>
                  <tr>
                    <th>Image</th>
                    <th>Title</th>
                    <th>Price</th>
                    <th>Sales</th>
                    <th>Action</th>
                  </tr>
                </thead>
                <tbody>
                  {
                    data.map((gig) => (
                      <tr key={gig._id} onClick={() =>
navigate(`/gig/${gig._id}`)}>
                        <td>
                          <img
                            className="cover"
                            src={gig.cover}
                            alt=""
                          />
                        </td>
                        <td>{gig.title}</td>
                        <td>{gig.price.toLocaleString("en-IN", {
                          maximumFractionDigits: 0,
                          style: "currency",
                          currency: "INR",
                        })}</td>
                        <td>{gig.sales}</td>
                        <td>
                          <img className='delete' src="./media/delete.png"
alt="delete" onClick={() => handleGigDelete(gig)} />
                        </td>
                      </tr>
                    ))
                  }
                </tbody>
              </table>
```

```
            </div>
      }
    </div>
  )
}

export default MyGigs
```

Not Found:

```
import React from 'react'
import './NotFound.scss';

const NotFound = () => {
  return (
    <div className='notFound'>
      <div className='container'>
        <h1>404</h1>
        <div className='text'>
          <h2>Page not found</h2>
          <p>Sorry, we couldn’t found the page you’re looking for.</p>
        </div>
      </div>
    </div>
  )
}

export default NotFound;
```

Orders:

```
import React, { useEffect } from 'react';
import { useQuery } from '@tanstack/react-query';
import { useNavigate } from 'react-router-dom';
import { axiosFetch } from '../../utils';
import './Orders.scss';

const Orders = () => {
  const currentUser = JSON.parse(localStorage.getItem('currentUser')) || {};
  const navigate = useNavigate();

  useEffect(() => {
    window.scrollTo(0, 0)
  }, [])

  const { isLoading, error, data } = useQuery({
    queryKey: ['orders'],
```

```
    queryFn: () =>
      axiosFetch.get(`/orders`)
        .then(({ data }) => {
          console.log(data);
          return data;
        })
        .catch(({ response }) => {
          console.log(response.data);
        })
  });

  const handleContact = async (order) => {
    const sellerID = order.sellerID.hasOwnProperty('_id') ? order.sellerID._id :
order.sellerID;
    const buyerID = order.buyerID.hasOwnProperty('_id') ? order.buyerID._id :
order.buyerID;

    axiosFetch.get(`/conversations/single/${sellerID}/${buyerID}`)
      .then(({ data }) => {
        navigate(`/message/${data.conversationID}`);
      })
      .catch(async ({ response }) => {
        if (response.status === 404) {
          const { data } = await axiosFetch.post('/conversations', {
            to: currentUser.isSeller ? buyerID : sellerID,
            from: currentUser.isSeller ? sellerID : buyerID
          });
          navigate(`/message/${data.conversationID}`)
        }
      })
  }

  return (
    <div className='orders'>
      {
        isLoading
          ? '...loading'
          : error
            ? 'Something went wrong!'
            : <div className="container">
              <div className="title">
                <h1>Orders</h1>
              </div>
              <table>
                <thead>
```

```
                  <tr>
                    <th>Image</th>
                    <th>{currentUser.isSeller ? 'Buyer' : 'Seller'}</th>
                    <th>Title</th>
                    <th>Price</th>
                    <th>Contact</th>
                  </tr>
                </thead>
                <tbody>
                  {
                    data.map((order) => (
                      <tr key={order._id}>
                        <td>
                          <img
                            className="img"
                            src={order.image}
                            alt=""
                          />
                        </td>
                        <td>{currentUser.isSeller ? order.buyerID.username :
order.sellerID.username}</td>
                        <td>{order.title}</td>
                        <td>{order.price.toLocaleString('en-IN', {
                          maximumFractionDigits: 0,
                          style: 'currency',
                          currency: 'INR',
                        })}</td>
                        <td>
                          <img className='message' src="./media/message.png"
alt="message" onClick={() => handleContact(order)} />
                        </td>
                      </tr>
                    ))
                  }
                </tbody>
              </table>
            </div>
      }
    </div>
  )
}

export default Orders
```

Pay:

```
import React, { useEffect, useState } from 'react';
import { loadStripe } from '@stripe/stripe-js';
import { useParams } from 'react-router-dom';
import { Elements } from '@stripe/react-stripe-js';
import { axiosFetch } from '../../utils';
import { CheckoutForm } from '../../components';
import './Pay.scss';

const stripePromise =
loadStripe('pk_test_51JT2jqSFY9RyfRMlOFUJ42d70JBSFztwI5hLDeUR4qLKJYOqIGu2tCIu2cD9
lc9rVlZthqsqGgasEfk2s2Z2eVJ100T2nqQNZf');

const Pay = () => {
  const { _id } = useParams();
  const [clientSecret, setClientSecret] = useState('');

  useEffect(() => {
    ( async () => {
      try {
        const { data } = await axiosFetch.post(`/orders/create-payment-
intent/${_id}`);
        setClientSecret(data.clientSecret);
      }
      catch({response}) {
        console.log(response);
      }
    })();
    window.scrollTo(0, 0)
  }, []);

  const appearance = {
    theme: 'stripe',
  };

  const options = {
    clientSecret,
    appearance,
  };

  return (
    <div className='pay'>
      <h2>Pay Securely with Stripe</h2>
      {clientSecret && (
        <Elements options={options} stripe={stripePromise}>
          <CheckoutForm />
```

```
        </Elements>
      )}
    </div>
  )
}

export default Pay
```

Success:

```
import React, { useEffect } from 'react';
import './Success.scss';
import { useLocation, useNavigate } from 'react-router-dom';
import { axiosFetch } from '../../utils';

const Success = () => {
  const { search } = useLocation();
  const navigate = useNavigate();
  const params = new URLSearchParams(search);
  const payment_intent = params.get('payment_intent');

  useEffect(() => {
    (async () => {
      try {
        await axiosFetch.patch('/orders', { payment_intent });
        setTimeout(navigate('/orders'), 5000);
      }
      catch ({ response }) {
        console.log(response);
      }
    })();
    window.scrollTo(0, 0);
  }, []);


  return (
    <div>Payment successful. You are being redirected to the orders page. Please
do not close the page</div>
  )
}

export default Success
```

Category Card:

```
import React from 'react';
```

```
import { Link } from 'react-router-dom';
import './CategoryCard.scss';

const Card = (props) => {
  const { data } = props;

  return (
    <Link to={`/gigs?category=${data.slug}`}>
        <div className='cardContainer'>
            <img src={data.img} alt={data.title} />
            <span className='desc'>{data.desc}</span>
            <span className='title'>{data.title}</span>
        </div>
    </Link>
  )
}

export default Card
```

Checkout Form:

```
import { useEffect, useState } from 'react';
import {
  PaymentElement,
  LinkAuthenticationElement,
  useStripe,
  useElements
} from "@stripe/react-stripe-js";
import './CheckoutForm.scss';

const CheckoutForm = () => {
  const stripe = useStripe();
  const elements = useElements();

  const [email, setEmail] = useState('');
  const [message, setMessage] = useState(null);
  const [isLoading, setIsLoading] = useState(false);

  useEffect(() => {
    if (!stripe) {
      return;
    }

    const clientSecret = new
URLSearchParams(window.location.search).get("payment_intent_client_secret");
```

```
    if (!clientSecret) {
      return;
    }

    stripe.retrievePaymentIntent(clientSecret).then(({ paymentIntent }) => {
      switch (paymentIntent.status) {
        case "succeeded":
          setMessage("Payment succeeded!");
          break;
        case "processing":
          setMessage("Your payment is processing.");
          break;
        case "requires_payment_method":
          setMessage("Your payment was not successful, please try again.");
          break;
        default:
          setMessage("Something went wrong.");
          break;
      }
    });
  }, [stripe]);

  const handleSubmit = async (event) => {
    event.preventDefault();

    if (!stripe || !elements) {
      // Stripe.js has not yet loaded.
      // Make sure to disable form submission until Stripe.js has loaded.
      return;
    }

    setIsLoading(true);

    const { error } = await stripe.confirmPayment({
      elements,
      confirmParams: {
        // Make sure to change this to your payment completion page
        return_url: import.meta.env.VITE_PAYMENT_SUCCESS_REDIRECT,
      },
    });

    // This point will only be reached if there is an immediate error when
    // confirming the payment. Otherwise, your customer will be redirected to
    // your `return_url`. For some payment methods like iDEAL, your customer will
    // be redirected to an intermediate site first to authorize the payment, then
```

```
    // redirected to the `return_url`.
    if (error.type === "card_error" || error.type === "validation_error") {
      setMessage(error.message);
    } else {
      setMessage("An unexpected error occurred.");
    }

    setIsLoading(false);
  }

  const paymentElementOptions = {
    layout: "tabs"
  }

  return (
    <form className='payment-form' id="payment-form" onSubmit={handleSubmit}>
      <LinkAuthenticationElement
        id="link-authentication-element"
        onChange={(e) => setEmail(e.target.value)}
      />
      <PaymentElement id="payment-element" options={paymentElementOptions} />
      <button disabled={isLoading || !stripe || !elements} id="submit">
        <span id="button-text">
          {isLoading ? <div className="spinner" id="spinner"></div> : "Pay now"}
        </span>
      </button>
      {/* Show any error or success messages */}
      {message && <div id="payment-message">{message}</div>}
    </form>
  )
}

export default CheckoutForm
```

Feature:

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import './Featured.scss';

const Featured = () => {
  const [search, setSearch] = useState('');
  const navigate = useNavigate();

  const handleSearch = () => {
```

```
    if(search) {
      navigate(`/gigs?search=${search}`);
    }
  }

  return (
    <div className='featured'>
      <div className="container">

        <div className="left">
          <h1>Find the perfect <span>freelance</span> services for your
business</h1>
          <div className="search">
            <div className="searchInput">
              <img src="./media/search.png" alt="search" />
              <input type="search" placeholder='Try "website"' onChange={(({
target: { value } }) => setSearch(value))} />
            </div>
            <button onClick={handleSearch}>Search</button>
          </div>
          <div className="popular">
            <span>Popular:</span>
            <button>Website Design</button>
            <button>WordPress</button>
            <button>Logo Design</button>
            <button>AI Services</button>
          </div>
        </div>

        <div className="right">
          <img src="./media/hero.png" alt="hero" />
        </div>

      </div>
    </div>
  )
}

export default Featured
```

Gig Card:

```
import React from 'react';
import { Link } from 'react-router-dom';
import './GigCard.scss';
```

```
const GigCard = (props) => {
    const { data } = props;

  return (
    <Link to={`/gig/${data._id}`} className="link">
      <div className="gigCard">
        <img src={data.cover} alt="" />
        <div className="info">
          <div className="user">
            <img src={data.userID.image || './media/noavatar.png'} alt="" />
            <span>{data.userID.username}</span>
          </div>
          <p>{data.title}</p>
          <div className="star">
            <img src="./media/star.png" alt="" />
            <span>{Math.round(data.totalStars / data.starNumber) || 0}</span>
            <span className='totalStars'>({data.starNumber})</span>
          </div>
        </div>
        <hr />
        <div className="detail">
          <img src="./media/heart.png" alt="" />
          <div className="price">
            <span>STARTING AT</span>
            <h2>
              {data.price.toLocaleString('en-IN', {
                maximumFractionDigits: 0,
                style: 'currency',
                currency: 'INR',
              })}
            </h2>
          </div>
        </div>
      </div>
    </Link>
  )
}

export default GigCard
```

Navbar:

```
import React, { useEffect, useState } from "react";
import { Link, useLocation, useNavigate } from "react-router-dom";
import Slider from 'react-slick';
import { GrFormNext, GrFormPrevious } from 'react-icons/gr';
```

```
import { axiosFetch, getCurrentUser } from "../../utils";
import "./Navbar.scss";

import "slick-carousel/slick/slick.css";
import "slick-carousel/slick/slick-theme.css";

const Navbar = () => {
  const currentUser = getCurrentUser();
  const [showMenu, setShowMenu] = useState(false);
  const [showPanel, setShowPanel] = useState(false);
  const { pathname } = useLocation();
  const navigate = useNavigate();

  const isActive = () => {
    window.scrollY > 0 ? setShowMenu(true) : setShowMenu(false);
  }

  useEffect(() => {
    window.addEventListener('scroll', isActive);
    return () => {
      window.removeEventListener('scroll', isActive);
    }
  }, []);


  const menuLinks = [
    { path: '/gigs?category=design', name: 'Graphics & Design' },
    { path: '/gigs?category=video', name: 'Video & Animation' },
    { path: '/gigs?category=books', name: 'Writing & Translation' },
    { path: '/gigs?category=ai', name: 'AI Services' },
    { path: '/gigs?category=social', name: 'Digital Marketing' },
    { path: '/gigs?category=voice', name: 'Music & Audio' },
    { path: '/gigs?category=wordpress', name: 'Programming & Tech' },
  ];

  const settings = {
    infinite: true,
    slidesToShow: 6,
    slidesToScroll: 2,
    prevArrow: <GrFormPrevious />,
    nextArrow: <GrFormNext />,
    swipeToSlide: true,
    responsive: [
      {
        breakpoint: 1024,
```

```
      settings: {
        slidesToShow: 3,
      }
    },
    {
      breakpoint: 600,
      settings: {
        slidesToShow: 3,
      }
    },
    {
      breakpoint: 480,
      settings: {
        slidesToShow: 2,
      }
    }
  ]
}

const handleLogout = async () => {
  try {
    await axiosFetch.post('/auth/logout');
    localStorage.removeItem('currentUser');
    navigate('/');
  }
  catch ({ response }) {
    console.log(response.data);
  }
}

return (
  <nav className={showMenu || pathname !== '/' ? 'navbar active' : 'navbar'}>
    <div className="container">
      <div className="logo">
        <Link to='/' className="link">
          <span className="text">HouseOfFlair</span>
        </Link>
        <span className="dot">.</span>
      </div>

      <div className="links">
        <div className="menu-links">
          <span><a href="#business1" className="link">Business</a></span>
          {/* <span>Explore</span> */}
          {/* <span>English</span> */}
```

```
            {!currentUser.isSeller && <span><Link to='/register'
className="link">Become a Seller</Link></span>}
          </div>
          {!currentUser && <span><Link to='/login' className="link">Sign
in</Link></span>}
          {!currentUser.username && <button className={showMenu || pathname !==
'/' ? 'join-active' : ''}><Link to='/register'
className="link">Join</Link></button>}
          {currentUser.username && (
            <div className="user" onClick={() => setShowPanel(!showPanel)}>
              <img src={currentUser.image || '/media/noavatar.png'} />
              <span>{currentUser?.username}</span>
              {showPanel && (
                <div className="options">
                  {
                    currentUser?.isSeller && (
                      <>
                        <Link className="link" to='/my-gigs'>Gigs</Link>
                        <Link className="link" to='/organize'>Add New Gig</Link>
                      </>
                    )
                  }
                  <Link className="link" to='/orders'>Orders</Link>
                  <Link className="link" to='/messages'>Messages</Link>
                  <Link className="link" to='/'
onClick={handleLogout}>Logout</Link>
                </div>
              )}
            </div>
          )}
        </div>
      </div>
      {(showMenu || pathname !== '/') && <>
        <hr />
        <Slider className="menu" {...settings}>
          {
            menuLinks.map(({ path, name }) => (
              <div key={name} className="menu-item">
                <Link className='link' to={path}>{name}</Link>
              </div>
            ))
          }
        </Slider>
      </>}
    </nav>
```

```
  );
};

export default Navbar;
```

Project Card:

```
import React from 'react';
import { Link } from 'react-router-dom';
import './ProjectCard.scss';

const ProjectCard = (props) => {
  const { data } = props;

  return (
    <Link className='link' to='/'>
        <div className='projectContainer'>
            <img src={data.img} alt={data.title} />
            <div className="info">
              <img src={data.pp} alt={data.title} />
              <div className="text">
                <h2>{data.cat}</h2>
                <span>{data.username}</span>
              </div>
            </div>
        </div>
    </Link>
  )
}

export default ProjectCard
```

Review:

```
import React from 'react';
import { getCountryFlag } from '../../utils';
import './Review.scss';

const Review = (props) => {
  const { review } = props;
  const country = getCountryFlag(review?.userID?.country);

  return (
  <div className="review">
      <div className="user">
```

```
        <img
          className="pp"
          src={review.userID?.image || '/media/noavatar.png'}
          alt=""
        />
        <div className="info">
          <span>{review?.userID?.username}</span>
          <div className="country">
            <img
              src={country?.normal}
              alt=""
            />
            <span>{review?.userID?.country}</span>
          </div>
        </div>
      </div>
      <div className="stars">
        {
          new Array(review.star).fill(0).map((star, i) => (
            <img key={i} src='/media/star.png' alt='' />
          ))
        }
        <span>{review.star}</span>
      </div>
      <p>{review.description}</p>
      <div className="helpful">
        <span>Helpful?</span>
        <img src="/media/like.png" alt="" />
        <span>Yes</span>
        <img src="/media/dislike.png" alt="" />
        <span>No</span>
      </div>
    </div>
  )
}

export default Review
```

Message:

```
import React, { useEffect } from "react";
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query';
import { axiosFetch } from '../../utils';
import { Link, useParams } from "react-router-dom";
import toast from 'react-hot-toast';
import "./Message.scss";
```

```
const Message = () => {
  const currentUser = JSON.parse(localStorage.getItem('currentUser')) || {};
  const { conversationID } = useParams();

  useEffect(() => {
    window.scrollTo(0, 0)
  }, [])

  const { isLoading, error, data } = useQuery({
    queryKey: ['messages'],
    queryFn: () =>
      axiosFetch.get(`/messages/${conversationID}`)
        .then(({ data }) => {
          return data;
        })
        .catch(({ response }) => {
          toast.error(response.data.message)
        })
  });

  const queryClient = useQueryClient();
  const mutation = useMutation({
    mutationFn: (message) =>
      axiosFetch.post('/messages', message)
    ,
    onSuccess: () =>
      queryClient.invalidateQueries(['messages'])
  })

  const handleMessageSubmit = (event) => {
    event.preventDefault();

    mutation.mutate({
      conversationID,
      description: event.target[0].value
    });

    event.target.reset();
  }

  return (
    <div className="message">
      <div className="container">
        <span className="breadcrumbs">
```

```
          <Link to="/messages" className="link">Messages</Link>
        </span>
        {
          isLoading
            ? '...loading'
            : error
              ? 'Something went wrong'
              : <div className="messages">
                {
                  data.map((message) => (
                    <div className={message.userID._id === currentUser._id ?
'owner item' : 'item'} key={message._id}>
                      <img
                        src={message.userID.image || '/media/noavatar.png'}
                        alt=""
                      />
                      <p>
                        {message.description}
                      </p>
                    </div>
                  ))
                }
              </div>
        }
        <hr />
        <form className="write" onSubmit={handleMessageSubmit}>
          <textarea cols="30" rows="10" placeholder="Write a message"></textarea>
          <button type='submit'>Send</button>
        </form>
      </div>
    </div>
  );
};

export default Message;
```

Reviews:

```
import React from 'react';
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query';
import { useNavigate } from 'react-router-dom';
import Review from '../Review/Review';
import { axiosFetch, userLogout } from '../../utils';
import toast from 'react-hot-toast';
import './Reviews.scss';
```

```
const Reviews = (props) => {
    const { gigID } = props;
    const navigation = useNavigate();
    const queryClient = useQueryClient();
    const { isLoading, error, data, refetch } = useQuery({
        queryKey: ['reviews'],
        queryFn: () =>
            axiosFetch.get(`/reviews/${gigID}`)
                .then(({ data }) => {
                    return data;
                })
                .catch(({ response }) => {
                    console.log(response.data);
                })
    });

    const mutation = useMutation({
        mutationFn: (review) =>
            axiosFetch.post('/reviews', review)
            .then(({data}) => {
                return data;
            })
            .catch(({ response: { data } }) => {
                if(data.message == 'jwt expired') {
                    userLogout();
                    navigation('/login');
                }
                toast.error(data.message);
            })
        ,
        onSuccess: () => {
            queryClient.invalidateQueries(['reviews'])
        }
    })

    const handleReviewSubmit = (event) => {
        event.preventDefault();

        const description = event.target[0].value;
        const star = event.target[1].value;

        if(star && description) {
            mutation.mutate({ gigID, description, star });
            event.target.reset();
        }
```

```
    }

    return (
        <div className="reviews">
            <h2>Reviews</h2>
            {
                isLoading
                    ? '...loading'
                    : error
                        ? 'Something went wrong!'
                        : data.map((review) => <Review key={review._id}
review={review} />)
            }
            <div className="add">
                <form className='addForm' onSubmit={handleReviewSubmit}>
                    <textarea cols="20" rows="10" placeholder='Write a
review'></textarea>
                    <select>
                        <option value={1}>1</option>
                        <option value={2}>2</option>
                        <option value={3}>3</option>
                        <option value={4}>4</option>
                        <option value={5}>5</option>
                    </select>
                    <button>Send</button>
                </form>
            </div>
        </div>
    )
}

export default Reviews;
```

Slide:

```
import React from 'react';
import Slider from 'react-slick';

import { PrevArrow, NextArrow } from '../../components';

import "slick-carousel/slick/slick.css";
import "slick-carousel/slick/slick-theme.css";

import './Slide.scss';

const Slide = (props) => {
```

```
  const { children, slidesToShow } = props;

  const settings = {
    infinite: true,
    slidesToShow: slidesToShow,
    slidesToScroll: slidesToShow,
    nextArrow: <NextArrow />,
    prevArrow: <PrevArrow />,
    swipeToSlide: true,
    responsive: [
      {
        breakpoint: 900,
        settings: {
          slidesToShow: 2,
          slidesToScroll: 2,
        }
      },
      {
        breakpoint: 600,
        settings: {
          slidesToShow: 1,
          slidesToScroll: 1,
        }
      },
      {
        breakpoint: 480,
        settings: {
          slidesToShow: 1,
          slidesToScroll: 1
        }
      }
    ]
  };

  return (
    <div className='slide-Container'>
      <Slider {...settings}>
        {children}
      </Slider>
    </div>
  )
}

export default Slide;
```

Trusted By:

```
import React from 'react';
import './TrustedBy.scss';

const TrustedBy = () => {
  return (
    <div className='trustedBy'>
      <div className="container">
        <span>Trusted by:</span>
        <img src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/meta.12b5e5c.png" alt="" />
        <img src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/google2x.4fa6c20.png" alt="" />
        <img src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/netflix2x.887e47e.png" alt="" />
        <img src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/pandg2x.6dc32e4.png" alt="" />
        <img src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/paypal2x.22728be.png" alt="" />
      </div>
    </div>
  )
}

export default TrustedBy;
```

CHAPTER 6

TESTING

6.1 TEST CASE 1: USER REGISTRATION

6.1.1 Test Steps:

- Open the website.
- Click on the "Join" button.
- Fill in the required fields with valid information, including username, email, password, profile picture, phone number and description.
- Click on the "Register" button.
- Verify that the user is redirected to a confirmation page or receives a confirmation email.

6.1.2 Expected Result:

- The user should be able to register successfully without any errors or exceptions.
- The user should be redirected to a confirmation page or receive a confirmation email to verify their registration.

6.1.3 Test Data:

- Username: JohnSmith
- Email: johnsmith@example.com
- Password: Password123
- Profile Picture: untitled.jpg
- Phone Number: 8446656652
- Description: I am a designer

6.1.4 Test Environment:

- Web browser (Chrome, Firefox, etc.)
- Test account or a dummy email address for receiving confirmation email (if applicable)

Notes:

- Ensure that all required fields are properly validated to prevent invalid or incomplete registrations.
- Check for any potential error messages or notifications during the registration process.
- Verify the accuracy and completeness of the confirmation email, if applicable.
- Repeat the test with different sets of valid and invalid data to cover various scenarios.

6.2 TEST CASE 2: ADDING GIG TO THE ACCOUNT

6.2.1 Test Steps:

- Open the website.
- Log in with valid credentials.
- Navigate to the Add New Gig Section.
- Fill in the required fields with valid information, such as title of project, category, cover image, description, and other required details.
- Create the gig.
- Verify that the gig is added.
- Go to the gig section and can check added gigs there.

6.2.2 Expected Result:

- The user should be able to add gigs by providing required information.
- The user should be able to edit added gigs and change according to the requirements.
- The user should be able to check previously added gigs.
- The user should be able to add new gigs and can be able to delete previously added gigs.

CHAPTER 7

LITERATURE REVIEW

6.1 INTRODUCTION:

The following literature review provides an overview of the existing research and literature related to e-marketplaces for freelancers, with a specific focus on House of Flair. The review explores the key themes, benefits, challenges, and potential impacts of such platforms in the freelance industry.

6.1.1 Rise of E-Marketplaces for Freelancers: The freelance industry has witnessed significant growth in recent years, fueled by the rise of digital platforms connecting freelancers with clients. Research by Katz and Krueger (2016) highlights the increased utilization of e-marketplaces, leading to the expansion of the freelance workforce and enhanced access to job opportunities.

6.1.2 Benefits of E-Marketplaces for Freelancers: E-marketplaces like House of Flair offer several benefits to freelancers, including:

- Enhanced visibility and access to clients: By joining an e-marketplace, freelancers gain exposure to a wider client base, enabling them to showcase their skills and attract potential clients (McKinsey Global Institute, 2016).
- Efficient project management: E-marketplaces provide freelancers with tools for managing projects, communicating with clients, and tracking progress, facilitating streamlined workflows and effective collaboration (Seethamraju, 2018).
- Credibility and trust-building: E-marketplaces often incorporate rating and review systems that help freelancers build credibility, establish a positive reputation, and increase their chances of securing future projects (Aljohani et al., 2021).

6.1.3 Challenges in E-Marketplaces for Freelancers: While e-marketplaces offer numerous benefits, there are also challenges that freelancers may face:

- **Increased competition:** The accessibility and ease of joining e-marketplaces result in a highly competitive environment where freelancers must differentiate themselves to stand out (Bou-Ghannam et al., 2018).
- **Pricing pressures:** E-marketplaces may foster price competition, leading to downward pressure on freelancers' rates, impacting their earning potential (Reed)
- **Platform dependency:** Freelancers relying solely on e-marketplaces may become dependent on these platforms for project acquisition, which can introduce uncertainties and risks (Lehdonvirta and Ernkvist, 2011).

6.1.4 Impact of E-Marketplaces on the Freelance Industry: The emergence of e-marketplaces has had significant impacts on the freelance industry:

- Economic empowerment: E-marketplaces contribute to the economic empowerment of freelancers by connecting them with a global market and facilitating entrepreneurship (Lindley and Toombs, 2017).
- Workforce transformation: The freelance workforce, supported by e-marketplaces, plays a crucial role in the gig economy and remote work trends, challenging traditional employment models (Rani et al., 2019).
- Skill development and lifelong learning: E-marketplaces provide opportunities for freelancers to continually develop their skills, adapt to market demands, and engage in lifelong learning (Sundararajan, 2017).

6.2 CONCLUSION:

The literature review highlights the growing importance and impact of e-marketplaces for freelancers, with House of Flair representing a significant platform in this context. The review demonstrates the benefits of e-marketplaces in terms of increased visibility, efficient project management, and trust-building. However, challenges such as increased competition and pricing pressures must also be acknowledged. Overall, e-marketplaces have transformed the freelance industry, empowered freelancers and reshaping the nature of work.

6.3 REFERENCES:

- Aljohani, N. R., et al. (2021). Factors influencing reputation and trust building in online freelancing marketplaces. Internet Research, 31(4), 1236-1262.
- Bou-Ghannam, D., et al. (2018). Freelancers on crowdworking platforms: Understanding motivations and experiences. Information Systems Journal, 28(3), 567-599.
- Katz, L. F., & Krueger, A. B. (2016). The rise and nature of alternative work arrangements in the United States, 1995–2015. NBER Working Paper No. 22667.
- Lehdonvirta, V., & Ernkvist, M. (2011). Knowledge map of the virtual economy. Centre for International Mobility and Cooperation.
- Lindley, S. E., & Toombs, A. (2017). "Uber and the economic empowerment of women? Not entirely..." Work, Employment and Society, 31(2), 412-426.
- McKinsey Global Institute. (2016). Independent work: Choice, necessity, and the gig economy. McKinsey & Company.
- Rani, U., et al. (2019). Impact of online freelancing on job quality and employment structure: Evidence from a South Asian platform. World Development, 115, 78-93.
- Reed, C. (2017). Freelancing in the gig economy: A review of the literature. New Technology, Work and Employment, 32(3), 195-212.
- Seethamraju, R. (2018). Development of a classification framework for digital platforms. Information Systems Journal, 28(1), 161-194.
- Sundararajan, A. (2017). The sharing economy: The end of employment and the rise of crowd-based capitalism. MIT Press.

CHAPTER 8

CONCLUSION

In conclusion, House of Flair has emerged as a prominent e-marketplace for freelancers, providing a user-friendly platform that connects freelancers with clients seeking their services. The platform offers a range of benefits, including increased visibility, efficient project management tools, and trust-building mechanisms through its rating and review system.

By leveraging the power of e-marketplaces, House of Flair has contributed to the economic empowerment of freelancers by providing access to a global market and facilitating entrepreneurship. The platform has also played a part in the transformation of the freelance industry, shaping the gig economy and remote work trends.

The platform has several features that make it easy for businesses and freelancers to connect and work together. These features include:

- A user-friendly interface that makes it easy to find and hire freelancers.
- A rating system that allows businesses to see how well freelancers have performed in the past.
- A messaging system that allows businesses and freelancers to communicate with each other.
- A payment system that makes it easy for businesses to pay freelancers.

While House of Flair brings significant advantages, it is important to recognize the challenges faced by freelancers, such as heightened competition and potential pricing pressures within the platform. Freelancers should adopt strategies to differentiate themselves and demonstrate their unique value to attract clients effectively.

Overall, House of Flair's impact on the freelance industry has been positive, creating opportunities for freelancers to showcase their skills, engage in meaningful collaborations, and enhance their professional growth.

CHAPTER 9

FUTURE SCOPE OF PROJECT

The House of Flair project has a promising future with several potential areas for expansion and improvement. The future scope of the project can include:

- Expansion into new markets: House of Flair can consider expanding its reach into new geographical regions, allowing freelancers and clients from different countries to connect and collaborate. This expansion would increase the platform's user base and diversify the available talent pool.
- Integration of additional services: To enhance the user experience, House of Flair can explore integrating additional services such as project management tools, time tracking software, or collaboration platforms.
- Specialization and niche markets: House of Flair can explore catering to specific industries or niche markets by creating specialized categories or sub-platforms.
- Enhanced user features: Continuously improving and expanding the platform's features based on user feedback is crucial. House of Flair can introduce features like video conferencing, portfolio showcases, or skill verification mechanisms to further enhance the user experience and promote trust among users.
- Mobile application development: Developing a mobile application for House of Flair would provide users with greater accessibility and convenience, allowing them to manage projects, communicate, and collaborate on the go.
- Continuous marketing and user acquisition: Ongoing marketing efforts will be essential to attract new users and retain existing ones.

By considering these future scope areas, House of Flair can continue to evolve and adapt to the changing needs and demands of the freelance industry, ensuring its sustained growth and success as an e-marketplace for freelancers.

CHAPTER 10

BIBLIOGRAPHY

- https://www.tutorialspoint.com
- https://www.javatpoint.com
- https://www.w3school.com
- https://www.youtube.com
- https://www.google.com
- https://www.wikepedia.com
- https://www.geeksforgeeks.com
- https://www.studocu.com