

## **Field Project**

**Nature's Nest**

**O7 SERVICES**

**A Field Project Report**

Submitted in partial fulfilment of the requirements for the

**Award of the degree of**

**“Bachelor of Computer Applications”**

**By**

**Mandeep Kaur**

**(Registration Number: 22307625485)**



**Centre for Distance and Online Education**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA, PUNJAB**

**Year 2025**

## **Contents of Report**

1.	Cover page– as per Annexure–I.....	1
2.	Declaration by student (as per Annexure–II) .....	4
3.	Training completion certificate from organization /Company (as per Annexure–III).....	5
4.	Acknowledgement.....	6
5.	List of figures.....	7
6.	List of abbreviations.....	8
7.	Chapter-1 INTRODUCTION TO THE PROJECT .....	9
	1.1 Objective and work undertaken .....	9
	1.2 Scope and Work .....	9
	1.3 Importance and Applicability.....	10
	1.4 Role .....	10
8.	Chapter–2 INTRODUCTION TO THE COMPANY.....	11
	2.1 Company Profile .....	11
	2.2 Project Done .....	12
9.	Chapter - 3 DESCRIPTION OF WORK DONE.....	13
	3.1 HTML .....	14
	3.2 CSS.....	15
	3.3 Java Script.....	18
	3.4 Bootstrap .....	20
	3.5 Angular .....	24
	3.6 Firebase.....	29
	3.7 Cloud firestore .....	31
	3.8 Firebase Authentication.....	32

3.9	Clouldinary .....	34
3.10	Tools and Technology used.....	35
10.	Chapter- 4 MY PROJECT	
4.1	Main page.....	37
4.3	Login Page.....	38
4.5	Register Page .....	39
4.7	Register & Login Services File .....	40
4.11	Routing Page .....	42
4.12	Firebase Database .....	44
4.14	DFDs.....	45
11.	Chapter -5 Final Chapter–CONCLUSION.....	48
5.1	Conclusion .....	48
5.2	Key Features .....	48
5.3	Future Scope.....	49
12.	Reference .....	50




## Student Declaration

To whom so ever it may concern

I, Mandeep Kaur, 22307625485, hereby declare that the work done by me on “Nature’s Nest” from March 2025 to April 2025, is a record of original work for the partial fulfillment of the requirements for the award of the degree, Bachelor of Computer Application. This work has not been submitted in part or full to any other university or Institution for the award of any degree.

Mandeep Kaur (Regd. No. 22307625485)



Signature of the student

Dated: 15/05/2025

Ref No: O7S/6W25/18

Date: 14/05/2025

### Certificate for the Field Project

This is to certify that Mr. / Ms. Mandeep Kaur has completed Summer Training titled Nature's Nest under the supervision of Ms. Palak from March 2025 to April 2025 in our organization.

His / her contribution during this summer training has been Excellent.

A circular stamp with "ISO 9001-2015" around the perimeter and "O7" in the center. Overlaid on the stamp is a handwritten signature in black ink.

(Authorized Signatory)

## **Acknowledgement**

I would like to acknowledge the invaluable support and contributions of the following individuals and institutions during the course of my field project.

- **Academic Supervisor:** I am sincerely grateful to Miss. Palak, my supervisor, for their expert advice, continuous encouragement, and constructive suggestions which were instrumental in shaping this project.
- **Institution/Organization:** My appreciation extends to O7 Services, where I conducted my fieldwork. The cooperation and openness of its members made the data collection process efficient and insightful.
- **Faculty and Department:** I also wish to thank the faculty of Computer Department of Lovely Professional University, for their academic support and for creating an environment conducive to research.
- **Family and Friends:** Lastly, my heartfelt thanks go to my family and peers for their emotional and moral support throughout this journey.

This project would not have been possible without the combined support of all the above.

Thanks

Mandeep Kaur

## **List of figures**

<b>Serial No</b>	<b>Figure Description</b>	<b>Page No</b>
1	Website Landing Page	32
2	Dashboard code snapshot	32
3	Login Page	33
4	Login Page code snapshot	33
5	Register Page	34
6	Register Page code snapshot	34
7	Register and Login file ts snapshot	35
9	Model	36
11	Routing code snapshot	37
13	Art pieces	38
14	Firebase Database	38
17	Level 0 Data Flow Diagram	40
18	Level 1 DFD Admin	40
19	Level 1 DFD Customer	41

### **List of Abbreviations**

<b>S.No</b>	<b>Abbreviation</b>	<b>Full Form</b>
1	API	Application Programming Interface
2	CMS	Content Management System
3	CSV	Comma-Separated Values
4	DB	Database
5	FTP	File Transfer Protocol
6	GUI	Graphical User Interface
7	HTML	HyperText Markup Language
8	HTTP	HyperText Transfer Protocol
9	JSON	JavaScript Object Notation
10	PDF	Portable Document Format
11	REST	Representational State Transfer
12	UI	User Interface
13	URL	Uniform Resource Locator
14	UX	User Experience



## **CHAPTER -1**

### **INTRODUCTION TO THE PROJECT**

Wood is one of the oldest materials used in art, having been employed since prehistoric times to create sculptures, paintings, and objects of beauty. The wood's versatility, beauty, and availability have made it a popular choice among artists of all ages.

- Welcome to [Nature's Nest], your premier destination for exquisite wooden art pieces crafted with passion and precision. Immerse yourself in the beauty of natural materials expertly transformed into stunning works of art that blend traditional craftsmanship with contemporary design.
- The aim is to show our creativity in art pieces and fulfill the customer's demands as per requirements(Customized Wooden Art Pieces).
- We provide a user-friendly platform for people to customize their design.
- At [Nature's Nest], we celebrate the timeless appeal of wood, harnessing its warmth, texture, and versatility to create a diverse collection that caters to every taste and decor. Whether you're looking for intricately carved sculptures, functional yet elegant furniture, or unique decor items that breathe life into your spaces, our curated selection promises to inspire and captivate.
- Explore our website and discover a world where craftsmanship meets artistry. Whether you're decorating your home, searching for a meaningful gift, or enhancing your collection, [Nature's Nest] invites you to experience the beauty and craftsmanship of wooden art pieces like never before.

#### **1.1 Objectives of the work undertaken**

- To design and develop a single-page web application using Angular.
- To implement Firebase as a backend service for real-time database operations and authentication.
- To understand and apply secure user login, registration, and role-based access controls.
- To gain hands-on experience in full-stack web development and deployment

#### **1.2 Scope of the Work**

The scope of the project included:

- Building responsive UI components using Angular.
- Implementing CRUD operations with Firebase Firestore.
- Integrating Firebase Authentication for user management.
- Hosting the web application on Firebase Hosting.
- Ensuring basic security and data validation measures.
- Advanced areas such as API integration, analytics, and user activity tracking were optionally explored based on time and requirements.

### 1.3 Importance and Applicability

This project is highly relevant in today's development landscape where cloud-based platforms and real-time applications are in demand. The integration of Angular and Firebase provides a scalable, serverless architecture suitable for startups, admin panels, content management systems, and real-time applications like chat or task tracking.

- Enhanced Visual Experience: Utilize high-resolution images and 360-degree views to showcase the intricate details and textures of wooden art pieces.
- User-Friendly Customization: Implement an intuitive interface for custom orders, allowing customers to specify dimensions, finishes, and personalized details easily.
- Streamlined Logistics: Improve packaging and shipping processes to ensure safe and timely delivery of delicate wooden art pieces, with tracking capabilities for customer peace of mind.

### 1.4 Role

As a Frontend Developer Intern, my role was to:

- Collaborate in planning the project structure and modules
- Develop user interfaces with Angular, ensuring usability and responsiveness
- Integrate Firebase services for authentication and database operations
- Conduct testing, debugging, and deploy the application to a live environment.


## **CHAPTER 2**


### **2.1 INTRODUCTION OF THE COMPANY / WORK**



O7 Services is an established company founded in 2015 and authorized by the government. They specialize in a variety of services including Web Development, Mobile Application Development, Custom Software Development, UI/UX Designing, Hosting services, Digital Marketing, Registration of Domain Names with various extensions, AMC & MMC Services, Bulk SMS and voice calls. O7 Services offers advanced IT solutions supporting the entire business cycle - from consulting to system development, deployment, quality assurance, and 24x7 support. With over 10 years of experience, the company aims to form long-lasting strategic partnerships with clients, offering affordable prices, timely delivery, and measurable business results. Their headquarters is located in Jalandhar with a branch office in Hoshiarpur.

Some of the products developed by O7 Services include Vehicle Tracking System, Invoice Software, School Management System, Hospital Management System, Parents-Teacher Communication App, Fee Management system, Task Management System, Online Food Ordering App, Security App, Admission system, Inventory Software, and Car Servicing App. Additionally, O7 Services provides training programs including 6 Weeks/6 Months Industrial Training, Project-Based Training, Corporate Training, and Job-Oriented Courses Training, covering major IT trends such as Full Stack Development (MEAN/MERN), Flutter, Kotlin Android, Swift UI (iOS), Firebase, Python, Angular, React Js, Vue Js, Node Js, ASP.NET, .NET Core, PHP, Laravel, CodeIgniter, Software Testing, Cloud Computing, Blockchain, DevOps, Data Science, Artificial Intelligence, Machine Learning, UI/UX Designing, Digital Marketing, WordPress, Linux, CCNP, CCNA Security, Network Security, Cyber Security, MCSE, MCITP, Java, Spring, Hibernate, C/C++, Photoshop, Adobe Illustrator, Figma, CorelDraw and many more

 +91- 8437365007, +91-181-5015007

 E-Mail: [enquiry@o7services.com](mailto:enquiry@o7services.com) , [hr@o7services.com](mailto:hr@o7services.com) [www.o7services.com](http://www.o7services.com)

## 2.2 PROJECT DONE



## **CHAPTER 3**

### **DESCRIPTION OF WORK DONE**

**Position of Training:** Frontend Web Developer Intern

**Role and Responsibilities:** During the training period, I worked as a frontend Web Developer Intern in a project focused on developing a dynamic web application using **Angular** and **Firebase** technologies. My role involved both individual and collaborative tasks in the software development life cycle. The key responsibilities included:

- Designing and developing user interfaces using **Angular** (components, services, modules, routing).
- Implementing responsive layouts using HTML, CSS, and Angular Material.
- Connecting the application to **Firebase, Firestore** for real-time data storage and retrieval.
- Integrating **Firebase Authentication** to manage user registration, login, and session control.
- Writing clean, maintainable code following modern development practices.
- Testing and debugging to ensure functionality across various devices and browsers.
- Deploying the application to **Firebase Hosting** and performing version control with Git/GitHub.
- Collaborating with mentors or team members for code reviews and project updates.

This role helped me gain hands-on experience with **frontend development, cloud database integration, and real-world project deployment**, significantly enhancing my technical and problem-solving skills.

#### **Activities performed:**

- Frontend development using Angular (components, templates, modules)
- Integrated Firebase services (authentication and Firestore)
- Created responsive layouts using CSS and Angular Material
- Deployed the application using Firebase Hosting
- Used Git for version control and collaboration
- Conducted testing and debugging using browser dev tools

#### **Equipment/Technologies Handled:**

- **Development Tools:** Angular CLI, Visual Studio Code, Chrome DevTools

- **Backend Services:** Firebase Firestore, Firebase Authentication
- **Deployment Platform:** Firebase Hosting
- **Version Control:** Git, GitHub
- **Languages/Frameworks:** HTML, CSS, Bootstrap, Java Script, Angular, Firebase

### 3.1 HTML



HTML (**HyperText Markup Language**) is the standard language used to create and structure content on the web. It uses **tags** to define elements like headings, paragraphs, images, links, forms, and more.




HTML uses a system of **tags** to tell the web browser how to display the content. Tags are typically enclosed in angle brackets (<>), and most tags come in pairs: an opening tag (<tag>) and a closing tag (</tag>). The closing tag usually has a / before the tag name.

#### **Key Components of an HTML Document**

Tag	What It Does
<!DOCTYPE html>	Declares the document is HTML5
<html>	Root element of the HTML page
<head>	Contains info like <title> and metadata
<title>	Sets the page title (shown in browser tab)
<body>	The main visible part of the web page
<h1> to <h6>	Headings; <h1> is biggest, <h6> is smallest
<p>	Paragraph of text
<ul> and <li>	Unordered list and list items
<img>	Displays an image
<a>	Creates a hyperlink

#### **Main Uses of HTML:**

Use Case	Description
 <b>Structure Web Pages</b>	Defines headings, paragraphs, lists, tables, etc.
 <b>Create Links</b>	Connects to other pages or websites using hyperlinks

Use Case	Description
 <b>Display Media</b>	Embeds images, videos, and audio
 <b>Form Creation</b>	Builds interactive forms for user input (e.g., login, signup)
 <b>Integrate with CSS &amp; JS</b>	HTML works with CSS (style) and JavaScript (behavior) to build full sites

## **3.2 CSS (CASCADING STYLE SHEET)**

CSS is a style sheet language used to describe the presentation (look and formatting) of a document written in HTML or XML. It allows you to apply styles (such as colors, fonts, spacing, layout, and animations) to web pages, making them visually appealing and user-friendly.

Here's a breakdown of its key features:

### **1. Style and Design Control**

CSS allows precise control over the appearance of HTML elements, including:

Text color and font

Backgrounds

Borders

Spacing (margin and padding)

Positioning and layout

### **2. Separation of Content and Style**

HTML handles structure and content

CSS handles visual presentation

This separation improves code organization, readability, and maintenance.

### **3. Reusability**

One CSS file can style multiple web pages.

Styles can be reused using classes, IDs, and selectors.

Makes updates easier—change once, and it applies everywhere.

### **4. Cascading and Inheritance**

The cascading nature allows multiple styles to apply with priority rules.

CSS also supports inheritance, where child elements can inherit styles from parent elements.

## 5. Multiple Styling Methods

CSS can be applied in three ways:

Inline (inside HTML tags)

Internal (within <style> tags)

External (linked .css files)

## 6. . Selectors and Grouping

CSS supports powerful selectors to target elements:

By element type (p)

By class (.menu)

By ID (#header)

Combinations and grouping (div p, h1, h2)

## 7. Responsive Design

CSS enables responsive layouts using media queries.

Adapts your site to various devices (mobile, tablet, desktop).

css

CopyEdit

```
@media (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

## 8. Animations and Transitions

CSS can animate elements without JavaScript:

Smooth transitions (e.g., fade, slide)

Keyframe-based animations

css

CopyEdit

```
div {  
  transition: background-color 0.5s ease;
```



## 9. Framework Support

CSS works well with frameworks and libraries like:

Bootstrap

Tailwind CSS

Foundation

These speed up development with pre-written, responsive styles.

## 10. Cross-Browser Compatibility

When written properly, CSS styles can work across all major web browsers, providing a consistent look and feel.

### Advantages of CSS:

- Clean separation of content and design
- Easier website maintenance
- Enables responsive and mobile-friendly design
- Reduces code repetition (reusability)
- Faster loading (especially with external CSS)

### Disadvantages of CSS:

- Browser compatibility issues (may display differently)
- Learning curve for layout systems like Flexbox or Grid
- Complex stylesheets can become hard to manage

Feature	Benefit
Style control	Colors, fonts, layout, and more
Separation of concerns	Cleaner, maintainable code
Reusability	Define once, use multiple times
Cascading & inheritance	Smart application of multiple rules
Responsive design	Mobile- and device-friendly layouts
Animation & effects	Add motion and interaction
Framework compatibility	Fast and professional UI development

### 3.3 JAVA SCRIPT

JavaScript is a high-level, interpreted programming language primarily used to create dynamic and interactive content on websites. It is one of the core technologies of the World Wide Web, alongside HTML and CSS. Java Script has evolved significantly and is now used not just on the client-side (in browsers) but also on the server-side (with environments like Node.js).

#### **Key Characteristics of JavaScript**

- **Interpreted Language:** JavaScript code is executed line by line by the browser without needing to be compiled.
- **Dynamically Typed:** Variables do not require a type declaration; types are determined at runtime.
- **Object-Oriented:** JavaScript uses prototypes and can create objects and inheritance.
- **Event-Driven:** JavaScript can respond to user interactions such as clicks, key presses, or page loading.
- **Multi-Paradigm:** It supports procedural, object-oriented, and functional programming styles.
- **Platform Independent:** Runs in any modern web browser without needing additional plugins.

#### **Key Features of JavaScript:**

Feature	Description
<b>Lightweight &amp; Interpreted</b>	Code runs directly in the browser without compiling.
<b>Dynamically Typed</b>	Variable types are determined at runtime.
<b>Object-Oriented</b>	Uses objects and classes (via prototypes or ES6 class syntax).
<b>First-Class Functions</b>	Functions are treated as variables – can be passed, returned, assigned.
<b>Event-Driven</b>	Responds to user actions like clicks, form inputs, etc.
<b>Asynchronous Support</b>	Manages tasks using callbacks, promises, and async/await.
<b>Platform Independent</b>	Runs on any OS or device with a web browser.
<b>Client &amp; Server-Side</b>	Works in both front-end (browser) and back-end (Node.js).
<b>Rich Standard Library</b>	Built-in functions for arrays, strings, math, dates, etc.
<b>DOM Manipulation</b>	Interacts with and modifies HTML/CSS in real time.
<b>Module Support (ES6)</b>	Supports modular code using import and export.
<b>Multi-Paradigm</b>	Supports functional, imperative, and object-oriented styles.

Feature	Description
<b>Large Ecosystem</b>	Wide range of libraries and frameworks (React, Angular, etc.).
<b>Browser API Integration</b>	Access to web APIs like fetch, localStorage, and Geolocation.

## JavaScript Used

Front-End Development: Dynamic websites and web apps.

Back-End Development: Using Node.js.

Mobile Apps: With frameworks like React Native.

Desktop Apps: Using Electron.js.

Game Development: Simple browser-based games.

Internet of Things (IoT) and machine learning (emerging areas).

## Advantages of Java script

Advantage	Description
1. Fast Execution	JavaScript runs directly in the browser, making it quick and responsive.
2. Client-Side Processing	Reduces server load by handling logic in the browser.
3. Easy to Learn & Use	Simple syntax and widespread documentation make it beginner-friendly.
4. Versatile (Full Stack)	Works for frontend (browser) and backend (Node.js) development.
5. Interoperable	Can be used with other languages and frameworks easily.
6. Rich Ecosystem & Community	Huge number of libraries, frameworks (React, Angular), and active community.
7. Dynamic Content	Makes web pages interactive with animations, validations, and real-time updates.
8. Wide Browser Support	Supported by all modern browsers without plugins.
9. Event-Based Programming	Efficient for user interaction handling.
10. Asynchronous Programming	Supports async operations with promises and async/await.

## Disadvantages of Java Script

Disadvantage	Description
1. Security Issues	Vulnerable to attacks like XSS (Cross-Site Scripting) since it runs on the client.

## Disadvantage

## Description

2. Browser Inconsistencies	Different browsers may interpret code differently (though this is improving).
3. No File or OS Access (in Browser)	Limited access to system-level operations for security reasons.
4. Debugging Can Be Complex	Debugging large, async JavaScript apps can be challenging.
5. Single Threaded	JavaScript uses a single-threaded model, which can lead to performance issues.
6. Heavy Dependency on Browser	Performance and features may depend on the browser used.
7. Code Can Be Viewed and Misused	JavaScript is visible in browser developer tools, which can lead to reverse engineering.
8. Runtime Errors	Errors appear only during execution, not at compile time.

## Why Use JavaScript?

- Makes websites interactive and engaging.
- Enables real-time updates and dynamic content.
- Can be used both on the front-end and back-end.
- Huge community and lots of learning resources.

## 3.4 BOOTSTRAP

Bootstrap is a free, open-source front-end framework used to design responsive and mobile-first websites quickly. It includes HTML, CSS, and JavaScript components for layout, typography, forms, buttons, navigation, and more. Originally developed by Twitter, Bootstrap is now maintained by the open-source community.

### Features of Bootstrap:

Feature	Description
<b>Responsive Design</b>	Adapts layout to different screen sizes (mobile, tablet, desktop).
<b>Grid System</b>	12-column layout system makes it easy to structure pages.
<b>Predefined CSS Classes</b>	Quick styling for elements like buttons, forms, and alerts.

Feature	Description
<b>JavaScript Components</b>	Includes modals, dropdowns, carousels, tooltips, and more.
<b>Customizable</b>	You can override Bootstrap styles or customize them using SASS.
<b>Cross-Browser Compatibility</b>	Works consistently across all major browsers.
<b>Open Source</b>	Free to use and modify.

## Bootstrap Elements:

### 1. Layout Elements

Element	Description
Container	A responsive fixed-width or full-width layout wrapper.
Grid System	Based on 12-column rows and columns to structure content.
Breakpoints	Responsive design based on screen sizes (sm, md, lg, xl, xxl).

### 2. Content Elements

Element	Description
Typography	Styles for headings, paragraphs, lists, blockquotes, etc.
Tables	Pre-styled tables with hover, striped rows, borders, etc.
Images	Classes like img-fluid make images responsive.
Figures	Combines images with captions.

### 3. Form Elements

Element	Description
Inputs	Text fields, checkboxes, radios, selects, etc.
Form Layouts	Inline forms, form groups, and rows for structured input.
Validation	Classes for client-side feedback on user input.

### 4. Component Elements

Component	Purpose
Alerts	Display messages or warnings.
Buttons	Pre-styled button styles and sizes.
Navbar	Responsive top navigation bar.
Modals	Pop-up dialog boxes.
Cards	Containers for text, images, and links.
Carousel	Slideshow/slider for images or content.
Dropdowns	Expandable menus.

## 5. Helper/Utility Elements

Utility	Description
Spacing	m-* for margin, p-* for padding.
Display	Show/hide elements with d-* classes.
Text Alignment	Align text using text-start, text-center, etc.
Colors	Text and background color helpers like text-danger, bg-warning.
Flex/Grid	Use flexbox/grid utilities for layout control.

## 6. JavaScript Plugins (Interactive elements)

JS Element	Functionality
Tooltips	Hover-based info boxes.
Toasts	Notification popups.
Collapse	Show/hide content.
Tabs	Toggle between content panels.
Offcanvas	Sidebar panels.

## How to Use Bootstrap

1. **Include Bootstrap via CDN:** The easiest way to use Bootstrap is by linking to its CSS and JavaScript files directly from a CDN (Content Delivery Network):

<!-- CSS -->

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel = "style sheet">
```

<!-- JavaScript (optional, for interactivity) -->

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js">
```

```
</script>
```

## Bootstrap 4 Advantages

- The first and foremost advantage of using Bootstrap is that it is very easy to use and implement. If a person has some basic knowledge of HTML and CSS, that user can easily use Bootstrap.
- The fact that Bootstrap can adapt to the size of any phone, tablet, desktop and so on is also very interesting feature.
- Bootstrap 4 is also useful because it is compatible with all modern browsers which include Google Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera.
- It also produces less cross-browser bugs.
- It is light weighted and consequently it can be widely used as a framework for creating responsive sites.
- Lastly, Bootstrap 4 is a very simple and yet very effective grid system.

## Why Use Bootstrap?

- **Faster Development:** It speeds up development by providing ready-made, customizable UI components and layouts.
- **Consistency:** It ensures consistency across your website with its uniform design.
- **Responsive:** It helps you create mobile-first, responsive websites without much hassle.
- **Community Support:** Bootstrap has a huge community and plenty of resources for learning and troubleshooting.

### 3.5 ANGULAR:

Angular is a TypeScript-based open-source front-end web application framework developed and maintained by Google. It is used to build dynamic, single-page web applications (SPAs) with a structured, component-based approach.

#### **Key Features of Angular:**

<b>Feature</b>	<b>Description</b>
Component-Based	UI is built using reusable components.
Two-Way Data Binding	Synchronizes data between the model and the view in real-time.
Dependency Injection	Promotes modular, testable, and maintainable code.
Routing	Built-in support for navigation and deep linking in SPAs.
Directives	Extend HTML with custom behavior (*ngIf, *ngFor, etc.).
RxJS and Observables	For handling asynchronous events and streams.
TypeScript Support	Strong typing, OOP support, and enhanced tooling via TypeScript.
CLI (Command Line Interface) Helps create, build, test, and deploy Angular projects efficiently.	

#### **Core Concepts**

<b>Concept</b>	<b>Purpose</b>
Module	Group of components, services, and other code.
Component	Controls a section of the UI; defined by a .ts, .html, and .css file.
Template	HTML + Angular syntax to define view layout.
Service	Shared logic (e.g., HTTP calls) separate from UI.
Routing	Enables navigation between views/pages.

#### **Angular CLI Commands**

<b>Command</b>	<b>Purpose</b>
ng new app-name	Create a new Angular project
ng serve	Run the app locally



Command	Purpose
ng generate component name	Create a new component
ng build	Compile and bundle for deployment

## Key Files & Folders in an Angular Project

### 1. angular.json

- Configuration file for the Angular CLI.
- Controls how the project is built, tested, and served.

### 2. package.json

- Lists project dependencies and scripts.
- Defines which packages (Angular, RxJS, etc.) are used.

### 3. tsconfig.json

- TypeScript configuration file.
- Sets compiler options for TypeScript.

### 4. src/ (*Main source code folder*)

Contains all the app logic, styles, and components.

#### Inside src/:

File/Folder	Description
main.ts	Entry point of the application. Bootstraps the root module.
index.html	Single HTML page Angular loads into the browser.
styles.css (or .scss)	Global styles for the application.
app/	Core application folder that contains components, modules, services, etc.

#### Inside src/app/ Folder

File/Folder	Description
app.module.ts	The root module that declares and imports components and other modules.
app.component.ts	Root component TypeScript file (logic).
app.component.html	Root component template (view).
app.component.css	Root component styles.
app-routing.module.ts	(If routing is enabled) Manages navigation and routes.

### Workspace Configuration Files:

All projects within a workspace share a configuration. The top level of the workspace contains workspace-wide configuration files, configuration files for the root-level application, and subfolders for the root-level application source and test files.

Workspace configuration files	Purpose
.editorconfig	Configuration for code editors.
.gitignore	Specifies intentionally untracked files that Git should ignore.
README.md	Documentation for the workspace.
angular.json	CLI configuration for all projects in the workspace, including configuration options for how to build, serve, and test each project.
package.json	Configures npm package dependencies that are available to all projects in the workspace. See npm documentation for the specific format and contents of this file.
package-lock.json	Provides version information for all packages installed into node_modules by the npm client. See npm documentation for details.
src/	Source files for the root-level application project.
public/	Contains image and other asset files to be served as static files by the dev server and copied as-is when you build your application.
node_modules/	Installed npm packages for the entire workspace. Workspace-wide node_modules dependencies are visible to all projects.

## Workspace

### configuration files

tsconfig.json

### Purpose

The base TypeScript configuration for projects in the workspace. All other configuration files inherit from this base file. For more information, see the relevant TypeScript documentation.

## Application Support Files

Files at the top level of src/ support running your application. Subfolders contain the application source and application-specific configuration.

### Application

### support files

### Purpose

app/

Contains the component files in which your application logic and data are defined.

favicon.ico

An icon to use for this application in the bookmark bar.

index.html

The main HTML page that is served when someone visits your site. The CLI automatically adds all JavaScript and CSS files when building your app, so you typically don't need to add any `<script>` or `<link>` tags here manually.

main.ts

The main entry point for your application.

styles.css

Global CSS styles applied to the entire application.

Inside the src folder, the app folder contains your project's logic and data. Angular components, templates, and styles go here.

### src/app/ files

### Purpose

app.config.ts

Defines the application configuration that tells Angular how to assemble the application. As you add more providers to the app, they should be declared here.

*Only generated when using the --standalone option.*

app.component.ts

Defines the application's root component, named AppComponent. The view associated with this root component becomes the root of the view hierarchy as you add components and services to your application.

app.component.html

Defines the HTML template associated with AppComponent.

app.component.css

Defines the CSS stylesheet for AppComponent.

## src/app/ files

## Purpose

app.component.spec.ts Defines a unit test for AppComponent.

app.module.ts Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here.

*Only generated when using the --standalone false option.*

app.routes.ts Defines the application's routing configuration.

## Application Source Files

Files at the top level of src/ support running your application. Subfolders contain the application source and application-specific configuration.

## Application support files

## Purpose

app/ Contains the component files in which your application logic and data are defined.

favicon.ico An icon to use for this application in the bookmark bar.

index.html The main HTML page that is served when someone visits your site. The CLI automatically adds all JavaScript and CSS files when building your app, so you typically don't need to add any <script> or<link> tags here manually.

main.ts The main entry point for your application.

styles.css Global CSS styles applied to the entire application.

Inside the src folder, the app folder contains your project's logic and data. Angular components, templates, and styles go here.

## src/app/ files

## Purpose

app.config.ts Defines the application configuration that tells Angular how to assemble the application. As you add more providers to the app, they should be declared here.

*Only generated when using the --standalone option.*

**src/app/ files****Purpose**

app.component.ts	Defines the application's root component, named App Component. The view associated with this root component becomes the root of the view hierarchy as you add components and services to your application.
app.component.html	Defines the HTML template associated with App Component.
app.component.css	Defines the CSS stylesheet for App Component.
app.component.spec.ts	Defines a unit test for App Component.
app.module.ts	Defines the root module, named App Module, that tells Angular how to assemble the application. Initially declares only the App Component. As you add more components to the app, they must be declared here.
	Only generated when using the --standalone false option.
app.routes.ts	Defines the application's routing configuration.

**Advantages of Angular**

- Modular structure for large applications
- Two-way data binding
- Real-time updates and dynamic UI
- Large ecosystem and community support
- Backed by Google
- Built-in form validation and routing

### **3.6 FIREBASE**

Firebase is a platform developed by Google that provides a set of cloud-based tools and services to help developers build, improve, and grow web and mobile applications. It eliminates the need to manage traditional backend infrastructure and allows developers to focus more on building rich, user-friendly applications.

It is especially popular for its ease of use, real-time capabilities, and seamless integration with other Google services.

**It provides tools for:**

- ❖ Authentication
- ❖ Realtime and Cloud Databases

- ❖ Hosting
- ❖ Cloud Functions
- ❖ Analytics
- ❖ Push Notifications
- ❖ and more.

## Core Features of Firebase

Service	Description
Authentication	Login with email/password, Google, Facebook, etc.
Firestore / Realtime DB	Cloud-hosted NoSQL databases for storing and syncing data in real-time.
Hosting	Fast and secure static web hosting with free SSL.
Cloud Functions	Serverless backend logic written in JavaScript or TypeScript.
Cloud Messaging (FCM)	Send push notifications to Android, iOS, and web users.
Analytics	Track user engagement, retention, and events using Google Analytics.
Security Rules	Control access to your data based on user roles or data values.
Cloud Storage	Store and serve user-generated files like images and videos.
App Distribution & Testing	Share early versions of apps for testing.

## Core Services Offered by Firebase

### ❖ Authentication

Simplifies user login with email/password, Google, Facebook, GitHub, and phone number.  
Secure and easy to implement.

### ❖ Cloud Firestore & Realtime Database

Cloud Firestore: A modern, scalable NoSQL database with strong querying and real-time sync.  
Realtime Database: JSON-based database with real-time data syncing.

### ❖ Hosting

Fast, secure static and dynamic web hosting.  
Built-in SSL and global CDN for better performance.

### ❖ Cloud Functions

Write backend logic (APIs, triggers) in JavaScript/TypeScript without managing a server.  
Used for handling events (e.g., file uploads, user signups).

### ❖ Cloud Messaging (FCM)

Send push notifications to Android, iOS, and web users.

### ❖ Cloud Storage

Store and serve large files like images, videos, and documents.

### ❖ Analytics

Track user behavior, engagement, and retention using Google Analytics for Firebase.

### ❖ Technical Overview

Firebase uses NoSQL databases for storing data.

Offers real-time data synchronization between clients.

Has a powerful serverless architecture using Cloud Functions.

Works well with front-end frameworks like React, Angular, and Vue.

Also supports mobile platforms like Android and iOS.

## 3.7 CLOUD FIRE STORE

Cloud Firestore is a NoSQL cloud database from Firebase (Google) designed to store, sync, and query data for web, mobile, and server apps. It offers real-time updates, offline support, and scalable infrastructure across the globe. It's the modern alternative to Firebase's original Realtime Database, with more powerful features and structured data storage. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

- Simple data is easy to store in documents, which are very similar to JSON.
- Complex, hierarchical data is easier to organize at scale, using subcollections within documents.
- Requires less denormalization and data flattening.
- Offline support for Apple, Android, and web clients.
- Write data operations through set and update operations as well as advanced transformations such as array and numeric operators.
- Transactions can atomically read and write data from any part of the database

### Firestore Structure

Firestore uses a document-based model, which is structured as follows:

- **Collections**

A collection is a container for documents.

Each collection can contain multiple documents.

Collections are schema-less, meaning documents within a collection don't have to follow the same structure.

- **Documents**

Documents are individual records inside a collection.

A document can store key-value pairs, where each key is a field name and each value is a field value (such as a string, number, map, array, etc.).

Every document has a unique ID, and it can optionally contain subcollections.

- **Fields**

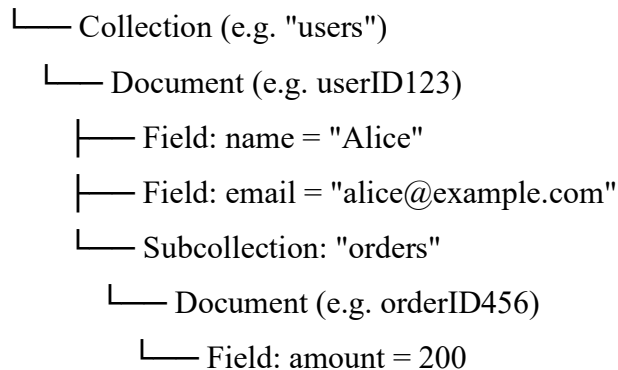
Fields represent the data within a document. These fields can be of various types: strings, numbers, booleans, arrays, maps, etc.

Firestore supports data types like timestamps, references to other documents, geographic locations, and more.

### **Basic Data Structure**

Firestore stores data in a **hierarchical structure**:

Firestore



- **Collections** contain **documents**.
- **Documents** contain **fields** (key-value pairs).
- Documents can also contain **subcollections**.

## **3.8 FIREBASE AUTHENTICATION**

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Firebase Authentication is a service that allows you to easily add secure user authentication to your web and mobile apps. It supports multiple sign-in methods and handles many security tasks like password recovery, account linking, and email verification.

It's simple to set up, secure, and integrates directly with other Firebase services like Firestore and Cloud Functions.



When we upgrade to Firebase Authentication with Identity Platform, we unlock additional features, such as multi-factor authentication, blocking functions, user activity and audit logging, SAML and generic OpenID Connect support, multi-tenancy, and enterprise-level support.

Firebase Authentication is a service that helps you authenticate users to our app. It provides a variety of authentication methods, including email/password, social logins (Google, Facebook, Twitter, etc.), and phone number authentication. Firebase Authentication integrates easily with other Firebase services, like Firestore, Realtime Database, and Cloud Functions, and provides backend services to help secure your app's data.

### Key Features

Feature	Description
Multiple Login Methods	Supports Email/Password, Google, Facebook, Twitter, GitHub, Phone, and anonymous login.
Secure & Managed	Google handles backend security and session management.
Email Verification	Send verification emails automatically.
Password Reset	Built-in support for resetting user passwords.
Account Linking	Combine multiple sign-in providers into one account.
User Management	Easily handle users from Firebase Console or Admin SDK.

### Supported Authentication Methods

- Email and Password
- Phone Number (with SMS)
- OAuth Providers:
  - Google
  - Facebook
  - Twitter
  - GitHub
  - Microsoft, Apple, Yahoo (via generic OAuth)
- Anonymous Authentication (great for guest access)

Firebase Authentication is a powerful tool for integrating secure authentication into our web or mobile apps. It supports a wide range of sign-in methods, from email/password to third-party logins and phone

authentication, all while handling the heavy lifting of security, session management, and integration with Firebase services.

### **3.9 CLOUDINARY**

Cloudinary is a cloud-based service that provides tools for managing, optimizing, and delivering images and videos across web and mobile applications. It's commonly used by developers and designers to handle media workflows, from uploading to transformation and delivery.

#### **Key Features of Cloudinary:**

##### **Image & Video Upload**

**Multiple upload sources:** Upload from local devices, URLs, social media, or directly from browsers/apps.

**Automatic format detection:** Cloudinary auto-detects file types and supports common image/video formats.

##### **Media Management**

**Media Library:** A visual UI to organize assets with folders, tags, and metadata.

**Asset organization:** Use public IDs, folders, and structured naming for better management.

**Search and filters:** Quickly find assets using advanced search and filters.

##### **Image & Video Transformation**

On-the-fly transformations via URL parameters (e.g., crop, resize, rotate, watermark).

Smart cropping & face detection: Automatically crop around key subjects like faces or objects.

**Text overlays:** Add dynamic or branded text overlays.

**Image effects:** Apply filters like grayscale, blur, pixelate, etc.

**Video transformations:** Trim, transcode, change bitrate, overlay subtitles, extract thumbnails.

##### **Performance Optimization**

**Auto quality and format:** Automatically optimize image size and format (e.g., convert to WebP, AVIF).

**Lazy loading:** Load assets only when needed to improve site performance.

**Responsive images:** Serve different sizes based on the user's device and screen size.

**CDN delivery:** Fast global delivery via multiple CDNs (Akamai, Fastly).

## **Security & Access Control**

**Signed URLs:** Prevent unauthorized access or manipulation of URLs.

**Upload presets & restrictions:** Control how assets are uploaded and by whom.

**Access control rules:** Limit visibility and transformations of assets.

## **Analytics & Monitoring**

**Usage statistics:** Monitor bandwidth, transformations, and storage.

**Performance tracking:** Track delivery speeds and optimization success rates.

## **AI & Smart Features**

**Auto-tagging:** Uses AI to detect and tag image content.

**Background removal (add-on):** Automatically remove backgrounds from images.

**Content-aware cropping:** Smartly crops based on subject importance, not just dimensions.

**User Management:** Implemented secure login, registration, and logout functionality using Firebase Authentication.

**Database Operations:** Performed CRUD (Create, Read, Update, Delete) operations on Firestore, including data structuring and indexing.

**Version Control:** Used Git and GitHub for source code management, branching, and collaboration.

**Deployment:** Deployed the live web application using Firebase Hosting.

## **3.10 Tools and Technologies Used:**

Angular CLI, Visual Studio Code, Firebase Console, GitHub, Postman, and Google Chrome.

## **Challenges faced and how those were tackled**

Several technical and project management challenges were encountered:

- Firebase security rules configuration initially blocked data access; resolved by carefully defining custom rules and testing them with different user roles.
- Data binding issues in Angular due to component state mismanagement; tackled by applying reactive forms and using proper lifecycle hooks.
- Deployment errors on Firebase Hosting; resolved by troubleshooting build errors and ensuring

proper configuration in angular.json and firebase.json.

### **Learning outcomes**

This project enhanced both my frontend development and cloud integration skills. Key outcomes include:

- Learned effective use of Angular services, routing, and component-based architecture.
- Understood Firebase's suite of tools for real-time databases, authentication, and hosting.
- Improved debugging, version control, and collaboration skills.
- Learned how to integrate Angular (frontend) with Firebase services like Firestore, Authentication, Storage, and Hosting.
- Understood client-server communication using Observables and Firebase's real-time capabilities.

## CHAPTER 4

### My Project

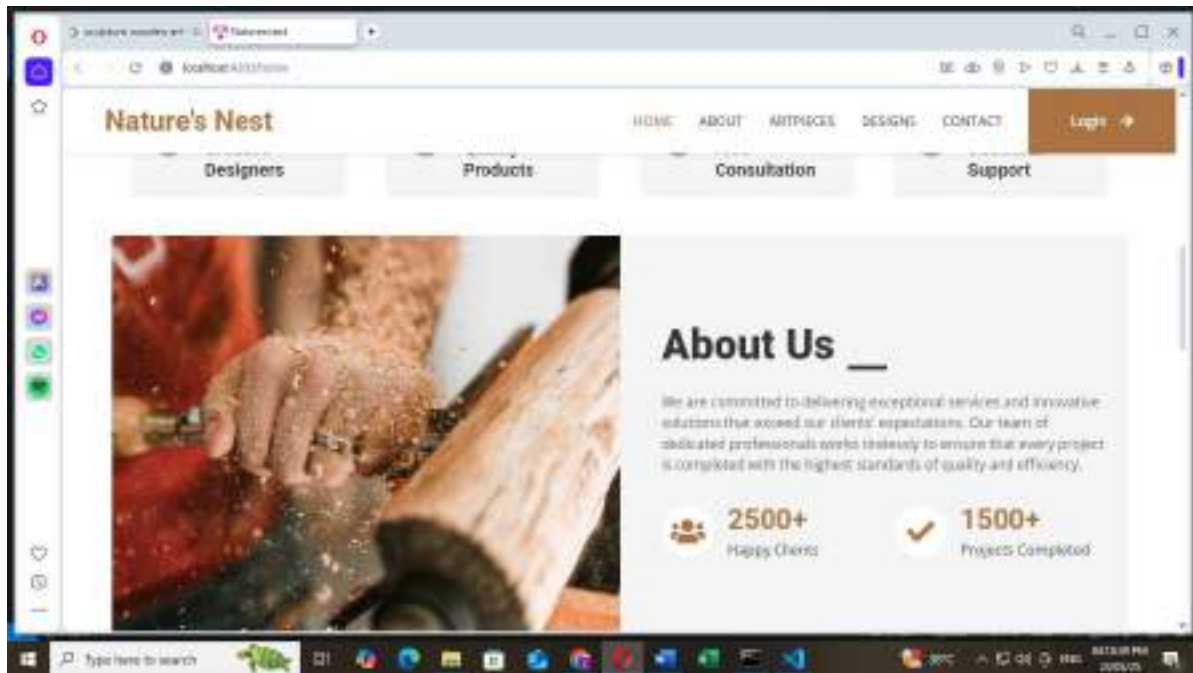


fig 4.1

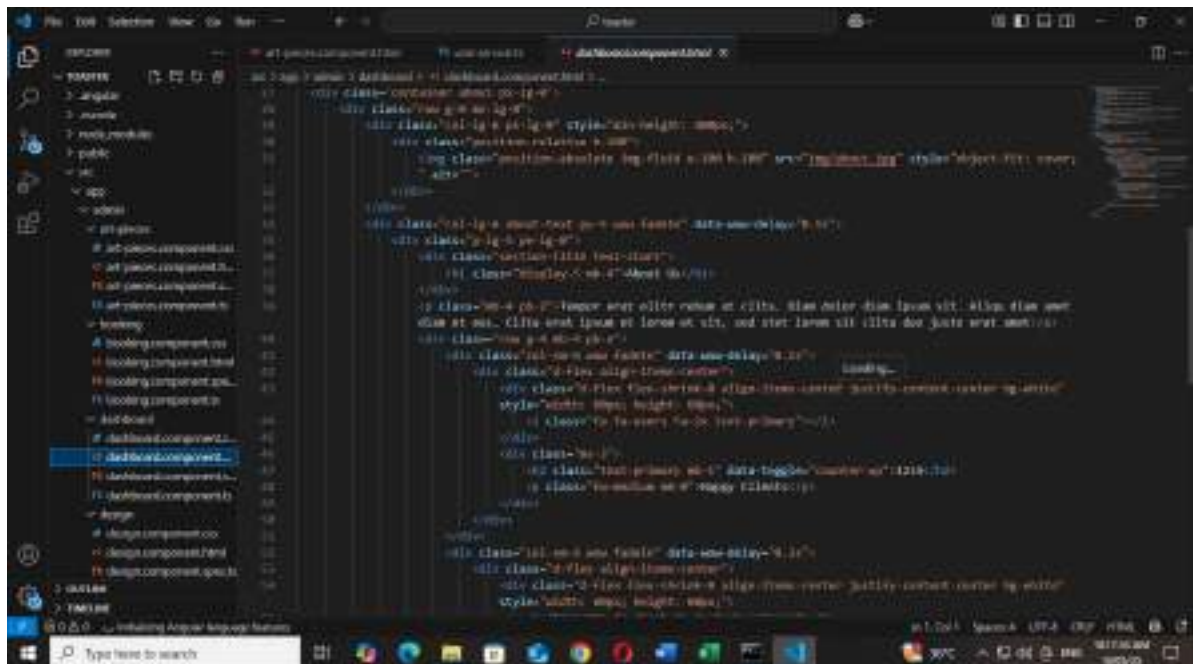


fig 4.2



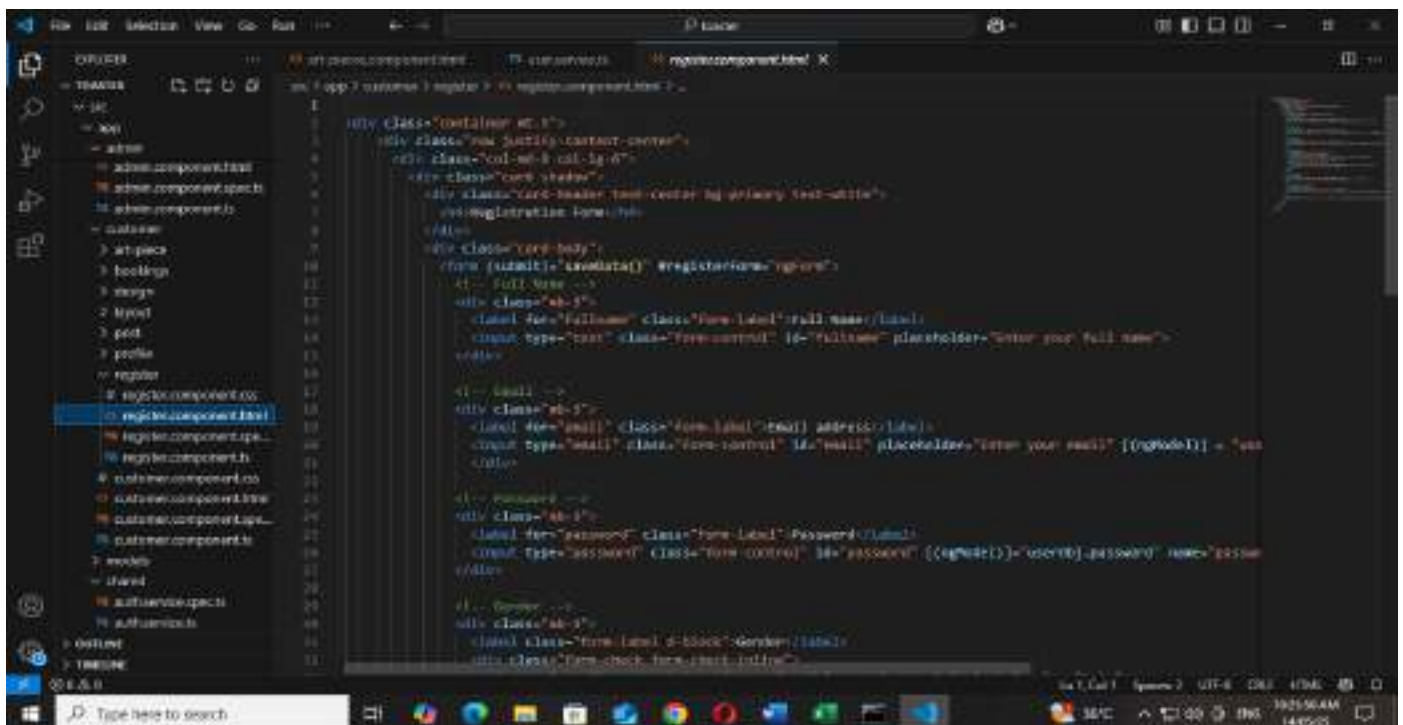
## Register Page



The screenshot shows a web browser displaying the 'Nature's Nest' website. The page has a navigation bar with links: DASHBOARD, ARTICLES, DESIGN, PAGES, POST, LOGIN, BOOKING, and a 'Data Table' button. The main content area features a 'Registration Form' with the following fields:

- Full Name:** Enter your full name
- Email address:** Enter your email
- Password:** Create a password
- Gender:** Radio buttons for Male and Female
- Phone Number:** e.g. 4021007000
- Agreement:** I agree to the Terms and Conditions
- Register:** A button to submit the form

fig 4.5



```

1  <!-- Full Name -->
2  <div class="form-label">Full Name</div>
3  <input type="text" class="form-control" id="Fullname" placeholder="Enter your full name">
4  </div>
5  <!-- Email -->
6  <div class="form-label">Email</div>
7  <input type="email" class="form-control" id="Email" placeholder="Enter your email" [(ngModel)] = "user.email">
8  </div>
9  <!-- Password -->
10 <div class="form-label">Password</div>
11 <input type="password" class="form-control" id="password" [(ngModel)] = "user.password" name="password">
12 </div>
13 <!-- Gender -->
14 <div class="form-label">Gender</div>
15 <div class="form-check form-check-inline">
16   <input type="radio"/> Male
17 </div>
18 <div class="form-check form-check-inline">
19   <input type="radio"/> Female
20 </div>
21 <div class="form-label">Phone Number</div>
22 <input type="text" class="form-control" id="Phone" placeholder="Enter your phone number">
23 </div>
24 <div class="form-check">
25   <input type="checkbox"/> I agree to the Terms and Conditions
26 </div>
27 <button type="submit" class="btn btn-primary">Register</button>
28 </div>

```

fig 4.6



## Register and Login Page Service.ts File

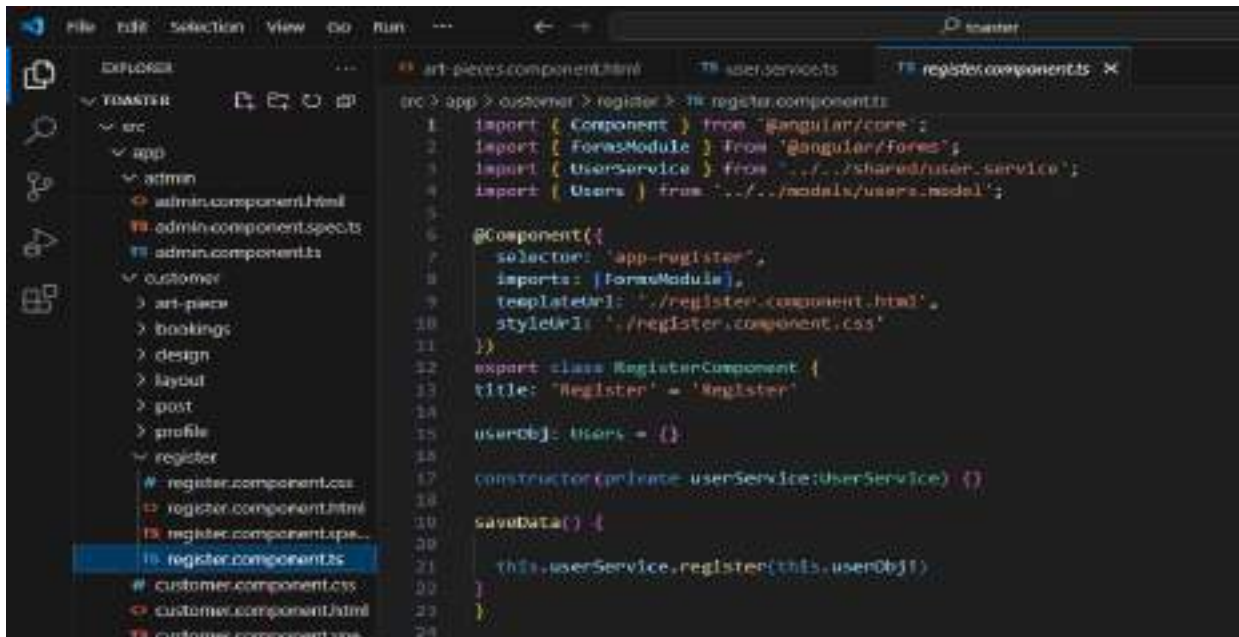


fig 4.7

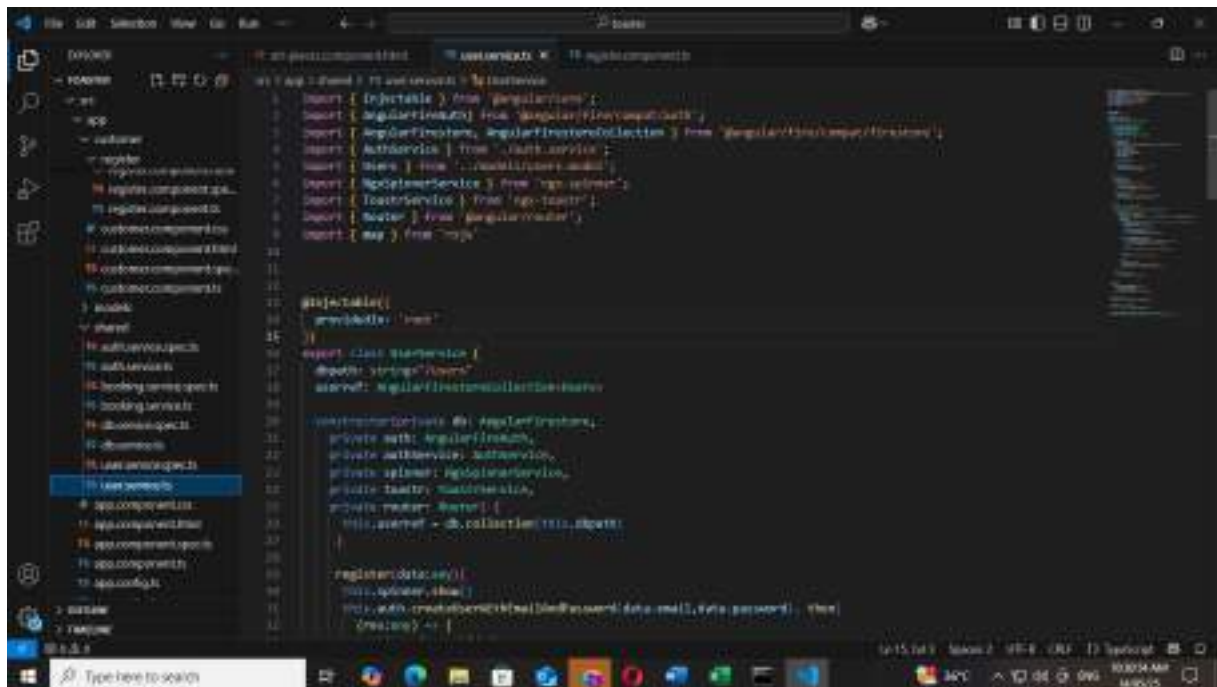
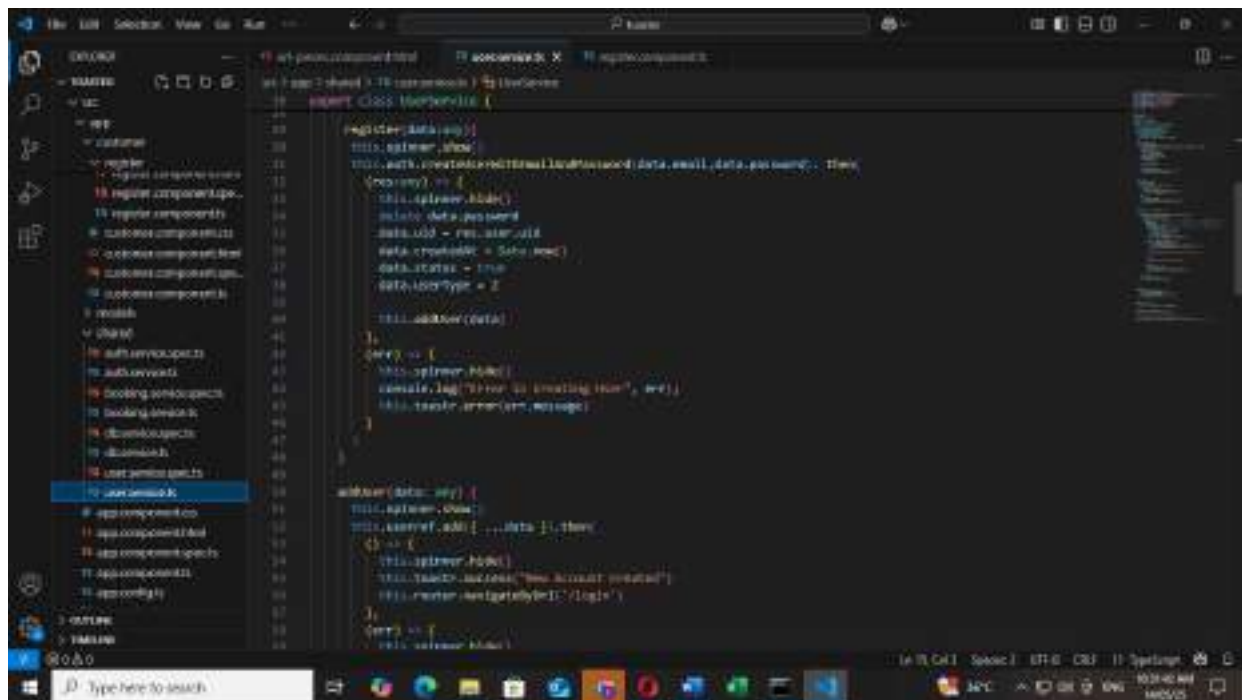


fig 4.8



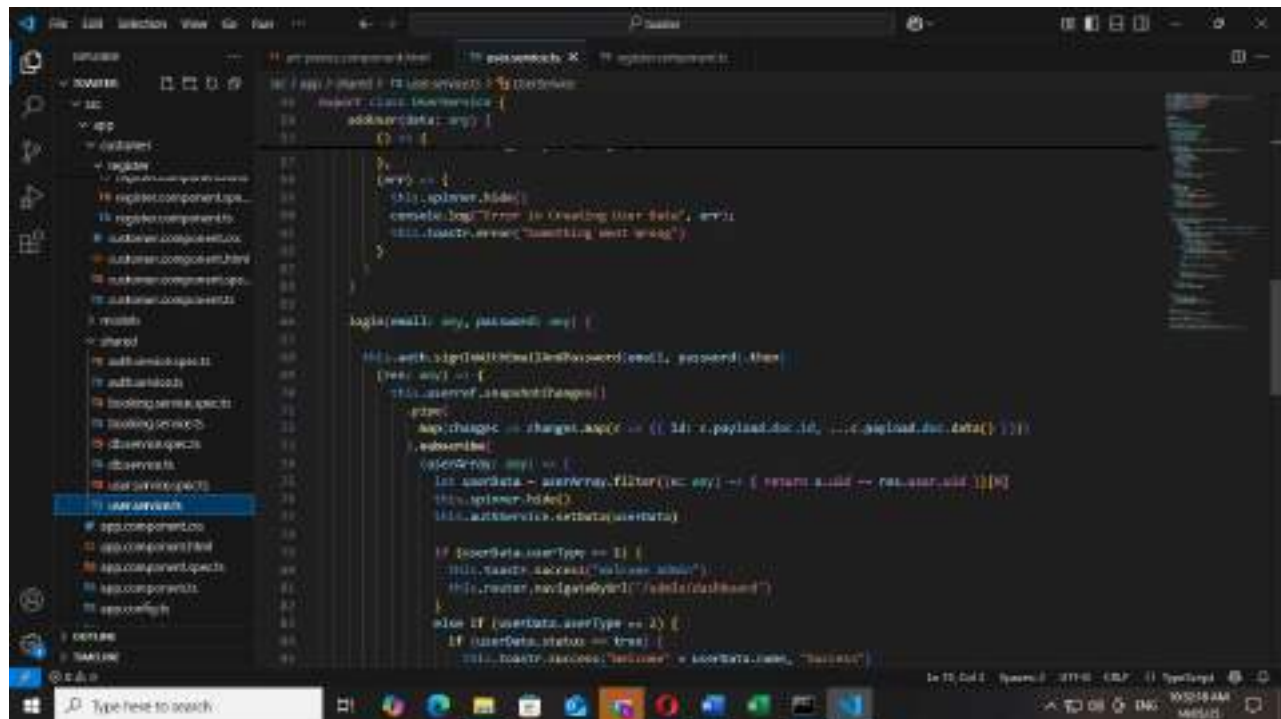


```

16 // app > shared > ts user-service > ts AuthService
17 export class AuthService {
18
19   register(data: any) {
20     this.spinner.show()
21     this.authService.createAndSendEmailAndPassword(data.email, data.password).then(
22       (res: any) => {
23         this.spinner.hide()
24         const data: any = {
25           email: res.email,
26           password: res.password,
27           status: true,
28           userType: 1
29         }
30         this.authService.addUser(data)
31       },
32       (err) => {
33         this.spinner.hide()
34         console.log('Error in creating user', err)
35         this.toastr.error(err.message)
36       }
37     )
38   }
39
40   addUser(data: any) {
41     this.spinner.show()
42     this.authService.addUser(data).then(
43       (res: any) => {
44         this.spinner.hide()
45         this.toastr.success('User Account created')
46         this.router.navigateByURL('/login')
47       },
48       (err) => {
49         this.spinner.hide()
50       }
51     )
52   }
53 }

```

fig 4.9

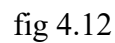


```

16 // app > shared > ts user-service > ts AuthService
17 export class AuthService {
18   addUser(data: any) {
19     (res: any) => {
20       this.spinner.hide()
21       console.log('Error in creating user data', err)
22       this.toastr.error('Something went wrong')
23     }
24   }
25
26   login(email: any, password: any) {
27     this.authService.signInWithEmailAndPassword(email, password).then(
28       (res: any) => {
29         this.authService.requestTokens()
30         const { token } = res
31         const { user } = res
32         const { email } = user
33         const { password } = user
34         const { status } = user
35         const { userType } = user
36         const { email } = user
37         const { password } = user
38         const { status } = user
39         const { userType } = user
40         const { email } = user
41         const { password } = user
42         const { status } = user
43         const { userType } = user
44         const { email } = user
45         const { password } = user
46         const { status } = user
47         const { userType } = user
48         const { email } = user
49         const { password } = user
50         const { status } = user
51         const { userType } = user
52         const { email } = user
53         const { password } = user
54         const { status } = user
55         const { userType } = user
56         const { email } = user
57         const { password } = user
58         const { status } = user
59         const { userType } = user
60         const { email } = user
61         const { password } = user
62         const { status } = user
63         const { userType } = user
64         const { email } = user
65         const { password } = user
66         const { status } = user
67         const { userType } = user
68         const { email } = user
69         const { password } = user
70         const { status } = user
71         const { userType } = user
72         const { email } = user
73         const { password } = user
74         const { status } = user
75         const { userType } = user
76         const { email } = user
77         const { password } = user
78         const { status } = user
79         const { userType } = user
80         const { email } = user
81         const { password } = user
82         const { status } = user
83         const { userType } = user
84         const { email } = user
85         const { password } = user
86         const { status } = user
87         const { userType } = user
88         const { email } = user
89         const { password } = user
90         const { status } = user
91         const { userType } = user
92         const { email } = user
93         const { password } = user
94         const { status } = user
95         const { userType } = user
96         const { email } = user
97         const { password } = user
98         const { status } = user
99         const { userType } = user
100        const { email } = user

```

## Routing



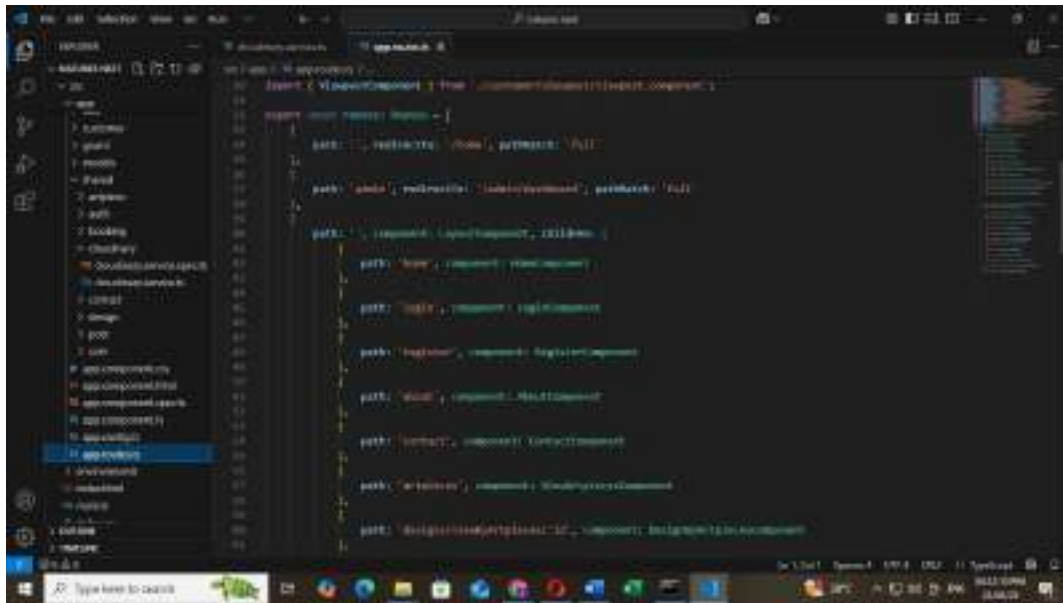
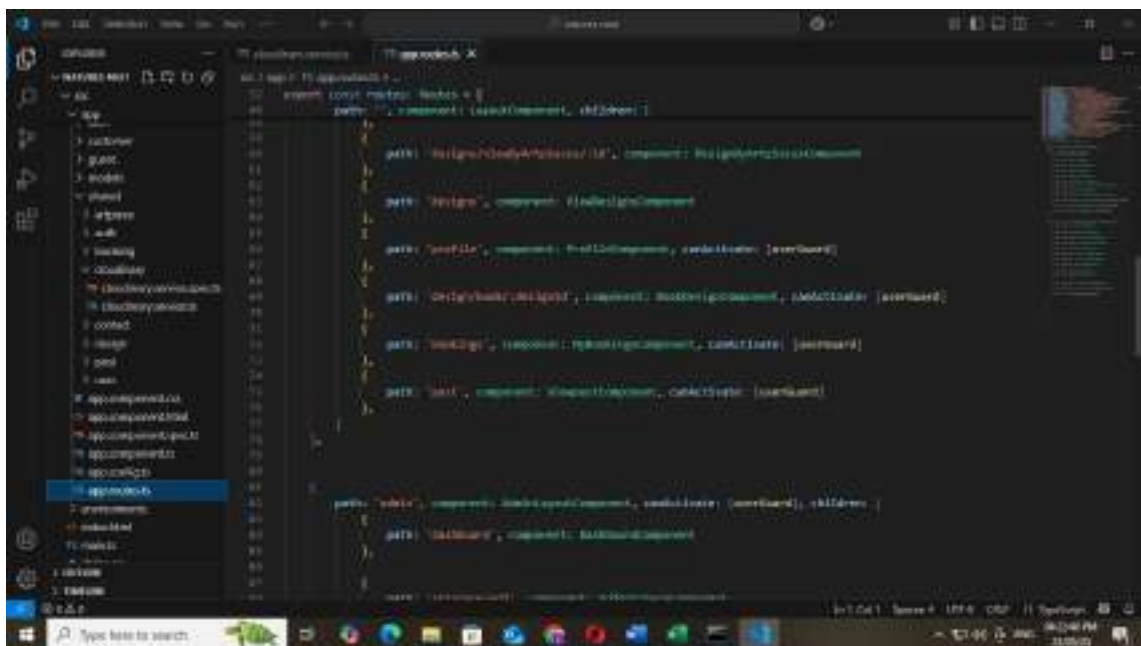


fig 4.13





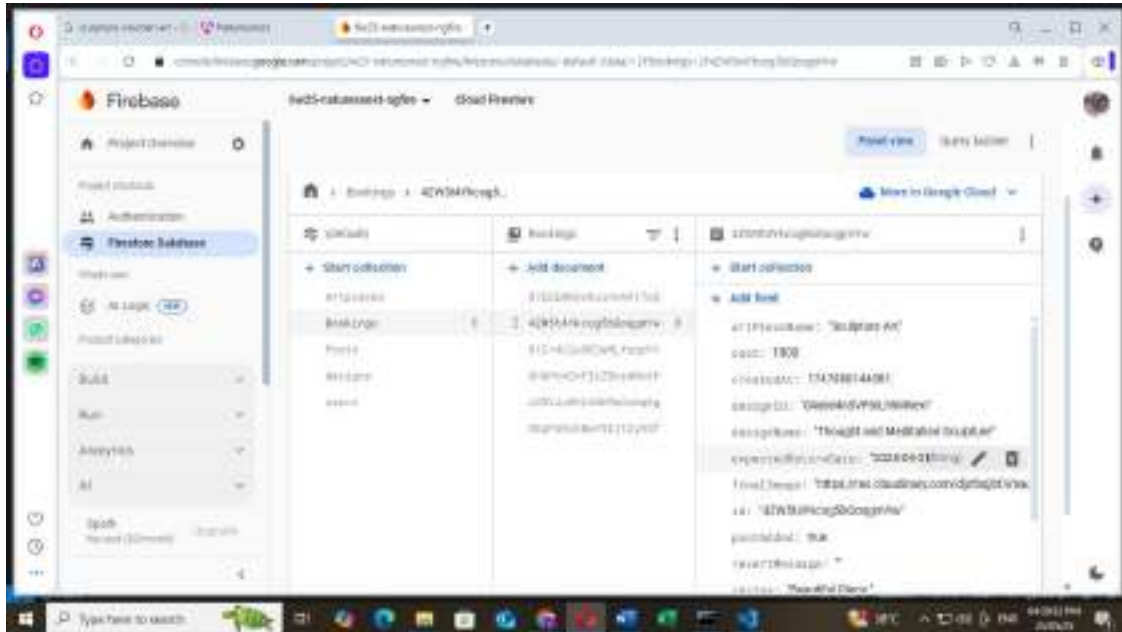


fig 4.16

## DFD's

Level 1



fig 4.17



## Level 2

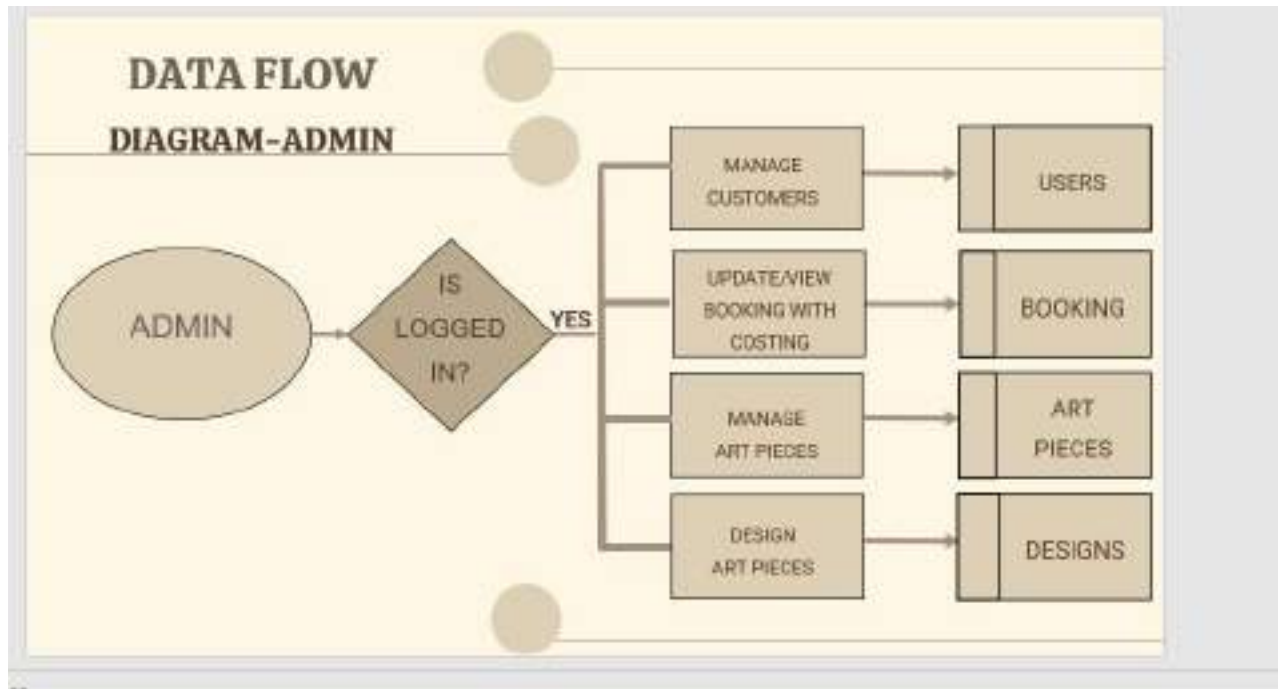


fig 4.18

## Level 2 customer

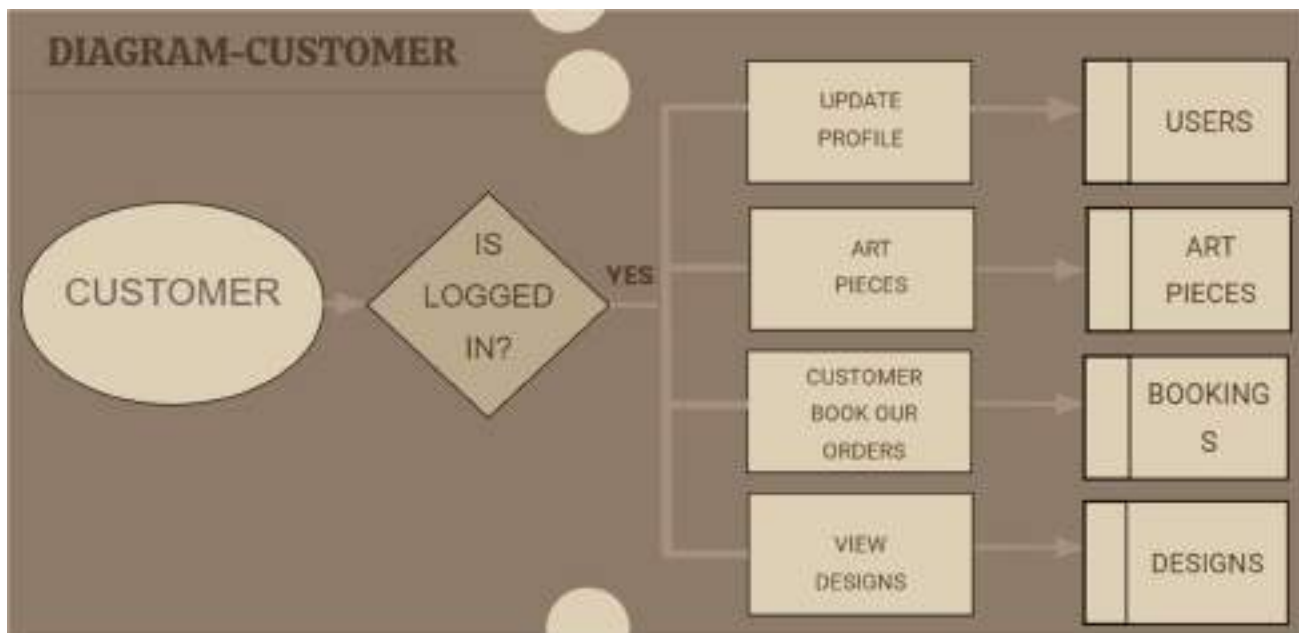


fig 4.19

## CHAPTER 5

### CONCLUSION

The “Nature’s Nest” Wooden Art Piece Gallery and Online Website using Angular + Firebase offers a powerful and modern solution for an interactive and scalable platform. The combination of Angular's dynamic, component-based frontend and Firebase's real-time, serverless backend provides a robust foundation to build and expand our online business, offering a seamless user experience and easy deployment.

#### **Key Highlights:**

- 🌿 **Elegant & Responsive UI:** Angular enables a dynamic and intuitive user experience across devices, reflecting the natural aesthetic of the wooden art-pieces.
- 🔥 **Real-time Backend:** Firebase ensures fast and synchronized product updates, inventory management, and customer interactions.
- 🛡️ **Secure User Authentication:** Firebase Authentication provides secure sign-in/sign-up features, protecting user data.
- ☁️ **Scalable Hosting & Database:** Firebase Hosting and Firestore allow the application to scale seamlessly with increased traffic or product listings.

By merging the warmth of handcrafted art with the efficiency of cloud-based technology, **Nature’s Nest** not only supports local artisans but also provides users with an enjoyable and trustworthy shopping experience.

This conclusion effectively summarizes the purpose of the website, encourages exploration, and emphasizes the appreciation for wooden art pieces.

#### **Future Scope**

The future of our Nature’s Nest website looks promising as we continue to evolve with the art and design landscape. With a growing global appreciation for sustainable materials and artisanal craftsmanship, we foresee expanding our collection to include more diverse styles and techniques from artists around the world. Embracing digital innovation, we aim to enhance user experience through virtual galleries, interactive features, and possibly augmented reality to provide a richer engagement with the artworks.

Furthermore, we plan to foster a community of artists and enthusiasts through forums, workshops, and

collaborations, promoting knowledge sharing and creativity. As environmental consciousness grows, we are committed to highlighting the sustainability of wood as a renewable resource, partnering with eco-friendly initiatives, and promoting responsible practices.

- Innovation in design
- AI-Powered system
- Craftsmanship



## **REFERENCES**

- ❖ Frontend Web UI Frameworks and Technology Tools  
edited by Dr. Ajay Bansal from Lovely Professional University.
- ❖ Fundamentals of Web Programming  
By Dr. Amit Sharma from Lovely Professional University.
- ❖ Weblink: <https://angular.dev/reference/configs/file-structure> (Accessed on 12<sup>th</sup> May 2025)
- ❖ Weblink: <https://getbootstrap.com/> (assessed on 12<sup>th</sup> May 2025)
- ❖ Weblink: <https://firebase.google.com/docs/build> (Accessed on 13<sup>th</sup> May 2025)