

**PROJECT REPORT**  
Submitted to  
**DEPARTMENT OF COMPUTER SCIENCE**



**VIVEKANAND EDUCATION SOCIETY'S  
COLLEGE OF ARTS, SCIENCE AND COMMERCE,  
SINDHI SOCIETY, CHEMBUR, MUMBAI 400071.**

**Personal Portfolio**

For Partial Fulfilment for Degree of  
**Bachelor of Science(Computer Science)**  
**Academic Year(2023-24)**

COORDINATOR OF DEPARTMENT

**Mr. Kamlakar Bhopatkar**

COLLEGE GUIDE

**Dr. Madhavi Vaidya**

**Mr. Kamlakar Bhopatkar**

SUBMITTED BY

**Krishna R. Kushwaha**



**VIVEKANAND EDUCATION SOCIETY'S  
COLLEGE OF ARTS, SCIENCE AND COMMERCE,  
SINDHI SOCIETY, CHEMBUR, MUMBAI 400071.**

**NAAC Re-Accredited 'A' Grade**

## **CERTIFICATE**

This is to certify that **Mr. Krishna R. Kushwaha** of T.Y.B.Sc(Computer Science) affiliated to University of Mumbai has successfully completed a project work entitled.

### **Personal Portfolio**

As partial fulfilment of the requirement for the degree of B.Sc.(Computer Science) for the Academic Year 2023-24.

**COORDINATOR OF DEPARTMENT**

**Mr. Kamlakar Bhopatkar**

**COLLEGE GUIDE**

**Mr. Kamlakar Bhopatkar**

**Dr. Madhavi Vaidya**

Date:

Examiner:

College Seal

## **Acknowledgement:**

With great pleasure, I present my project, '**Personal Portfolio**,' and extend sincere gratitude to all those whose invaluable contributions have been pivotal to its success.

This project report stands as a testament to collective efforts, and I express heartfelt thanks to **V.E.S. College** and our esteemed Principal, **Dr. Anita Kanwar**, for their unwavering support.

A special acknowledgment is reserved for our dedicated Project Guides, **Mr. Kamlakar Bhopatkar** and **Dr. Madhavi Vaidya**, whose timely and invaluable guidance played a crucial role in each phase of the project's development.

The synergy among the project team and the mentorship provided were integral to overcoming challenges and achieving the milestones outlined in the 'Personal Portfolio.'

This endeavor has been a true collaboration, and I extend my gratitude to every individual who has contributed to its success. Your collective efforts have not only shaped this project but have also enriched the overall learning experience.

In closing, I express my deepest appreciation to all involved in making 'Personal Portfolio' a reality. Your support and guidance have been indispensable, and I am genuinely thankful for the privilege of undertaking this project within such a supportive community.

As I reflect on this journey, I am inspired by the collective commitment to excellence and the pursuit of knowledge. This experience has not only honed my technical skills but has also instilled in me a deep sense of gratitude for the collaborative spirit that defines our academic community.

I look forward to applying the knowledge gained from this project as I embark on future endeavours, and I am confident that the lessons learned will serve as a solid foundation for my continued growth in the field.

**INDEX**

<b><u>SR.No</u></b>	<b><u>TOPIC</u></b>	<b><u>PAGE NO</u></b>
	<b>PRELIMINARY INVESTIGATION</b>	
1	Organizational Overview.	6
2	Description of Existing System.	6
3	Limitations of Existing System.	6
4	Description Proposed system	7
5	Advantages of Proposed system and Modules	7
6	Technologies Used	9
	<b>SYSTEM ANALYSIS</b>	
7	Use Case Diagram.	11
8	Activity Diagram.	13
9	ER Diagram	15
10	Class Diagram	17
	<b>SYSTEM IMPLEMENTATION</b>	
11	Test Cases	20
12	Sequence Diagram	21
13	Component Diagram	23
14	Deployment Diagram	25
15	Images of application	27
16	Future Enhancements	30
17	References	30

## **Personal Portfolio**

KRISHNA

## **Preliminary Investigation**

## **Personal Portfolio**

### **Organizational overview:**

A portfolio website is a digital space where we may display all our efforts, talents, and experiences. Creating a portfolio website is a professional method to display our efforts, experiences, or brand. Employers and job seekers can view a sample of our work on this website. It aids in giving the employer an opportunity to see our work during their downtime and enables them to learn about us in a relaxed situation. When a candidate applies for a job, they can create a website called a portfolio that showcases their educational background, their abilities, and the projects they have worked on. An individual's career background is briefly described on their personal portfolio webpage. They can include a link to it in their CV.

### **Description of Existing System:**

The majority of portfolios attempt to explain how a graduating bachelor's design student with a focus on online design and development developed and built a web portfolio. It will teach the core philosophy and components of the design process for an online portfolio in addition to defining what a portfolio website is. Additionally, the portfolios outline various avenues and methods via which a design student might establish and grow a personal online portfolio. By going through topics like how to incorporate one's own visual identity and what is needed to create a successful portfolio, awareness the numerous methods and techniques employed today is crucial for carrying out this process, as is developing a thorough awareness of contemporary tools and trends.

### **Limitations of Existing System:**

Website builder platforms include templates for standardised websites. To put it another way, every website created using these templates will have the exact same navigation, layout, and design. The amount of design customization available to business owners to create distinctive websites is restricted.

Here is a list of the limitations.

#### **1. Usability problems**

Even with the aid of website builders, creating a personal website may take a significant amount of time and effort. With these programmes, you may simply create a website, but selecting from hundreds of themes can be daunting for beginners, which can result in severe design errors. Furthermore, you might need to code it yourself if you want a special theme or a few extra features.

#### **2. Responsiveness**

The portfolio website must be flexible. Considerable screen sizes include mobile view and desktop view.

### **3. Over-customisation**

Any technique you want may be used to customise a website. In order to assist you actualize your branding, website builders also provide you a variety of ready-to-use themes. Despite all of that, you would still need to think about the beauty of the design because a confusing layout is one issue that frequently plagues personal websites.

### **4. Accessibility issues**

Typically, you would need a laptop or computer to add or delete a project from your portfolio website, especially if the modifications require coding. Website modification can also change the content; for instance, a section of blank space can be omitted, or the mobile view might not be as user-friendly as it appears.

## **Description Proposed system:**

The IT industry was effectively expanding across the market. Many individuals are attempting to present themselves using internet resources like websites, blogs, etc. Many businesses and organisations compete for survival in the cutthroat IT industry by running effective official websites online. This prompts us to create my idea using current market trends that are widely discussed and have a broad reach. My project's objective is to create a personal website for myself where I can display my talents, recent and prior efforts, and experiences. A job seeker can set up a portfolio website where he can showcase his efforts, past projects, key competencies, services provided, and any certificates or honours.

One can effectively improve their designing and development talents by creating a portfolio website. If we include a website link in my resumes, the hiring manager will probably open it and look at my work. They will be able to trust us and our work since they will know what we are capable of. There are several areas on our website, including:

- **Title & Footer Section**
- **Navigation Bar.**
- **About Me section.**
- **Projects Section**
- **Technology Stack**

## **Advantages of Proposed system:**

### **1. User Interface**

A website that is easy to grasp thanks to its user-friendly design.

### **2. Responsiveness**

The portfolio website must be flexible. Considerable screen sizes include mobile view (less than 600 pixels) and desktop view (more than 600 pixels)

### **3. Customisation**

Offers a lot of creative freedom and is versatile.

### **4. Design**

The website was not constructed using a pre-made template.

### **5. Data Customisation**

Real time changes of data`

## **Modules:**

The Portfolio module was created just for me as a custom module. The website enables you to present relevant information that will provide my prospects.

The Portfolio Module combines numerous content kinds into a single listing. You may add portfolios to sites to produce more strong content across numerous pages because each listing is self-contained.

### **1. Home**

Your website's main page serves as each visitor's initial introduction to your website. They'll go over your site before deciding whether to become a client to get a sense of what you do, why that matters to them, and how they may gain from what you have to offer.

### **2. Service**

A service page is crucial to create for a website since it enables people to see what I can create and provides them with a summary of what I can offer.

### **3. Contact**

Building a website must have a Contact Us page since it enables users to get in touch with you quickly.

### **4. Project section**

This section features our completed projects and works that we have produced.

### **5. About me**

We can briefly describe ourselves, our technological talents, and academic background in this.



## **Technologies Used:**

### **1. HTML**

HTML is the standard markup language for creating webpages.

### **2. SCSS**

Sassy Cascading Style Sheets. It is basically a more advanced and evolved variant of the CSS language

### **3. Node JS**

Node js is simple to use as a server-side proxy since it can manage several connections at once in a nonblocking fashion. It is helpful for gathering data from several source points or proxying various services with diverse response times.

### **4. Vite**

Vite is a next-generation, front-end tool that focuses on speed and performance. It consists of two major parts: A development server that provides rich feature enhancements over native ES modules: fast Hot Module Replacement (HMR), pre-bundling, support for typescript, jsx, and dynamic import.

### **5. React JS**

A Java-script library called React JS is used to build front-end websites with dynamic features. React is a component-based method where we build pages by disassembling them into separate components. Reacts main objectives are component reuse and dynamic functionality between different pages of a website or application.

### **6. Sanity**

This is a schema less backend that lets you store and query JSON documents, and subscribe to real-time changes. It comes with a query API that uses the query language GROQ to let you quickly filter down to the documents you want, and project exactly the data structure you want your content in.

## **SYSTEM ANALYSIS**

## Use Case:

Use case diagrams are used to depict the context of the system to be built and the functionality provided by that system. They depict who (or what) interacts with the system. They show what the outside world wants the system to do

### ELEMENTS OF USE CASE DIAGRAM:

#### Actors:

Any entity (or entities) that play specific roles in a particular system are portrayed by actors. The many roles that the actor plays correspond to the users' actual job functions inside a particular system. In a use case diagram, an actor engages in interaction with a use case. All of the stakeholders, who are the target users of the app, are involved. It makes no sense to represent an entity as an actor if it has no impact on the functionality you are simulating. In a use case diagram that is portrayed "outside" the system boundaries, an actor is represented by a stick figure.

#### Use Cases:

A use case is a graphic depiction of a specific piece of business functionality in a system. You should make a list of the discrete business functions in the form of simple scenarios as the first step in determining use cases. These business operations may all be categorised as possible use cases. In a use case diagram, a use case is represented by an ellipse.

#### System Boundary:

A system boundary defines the scope of what a system will be. The system boundary is shown as a rectangle spanning all the use cases in the system.

### Relationships: The following relationships can be established among use cases:

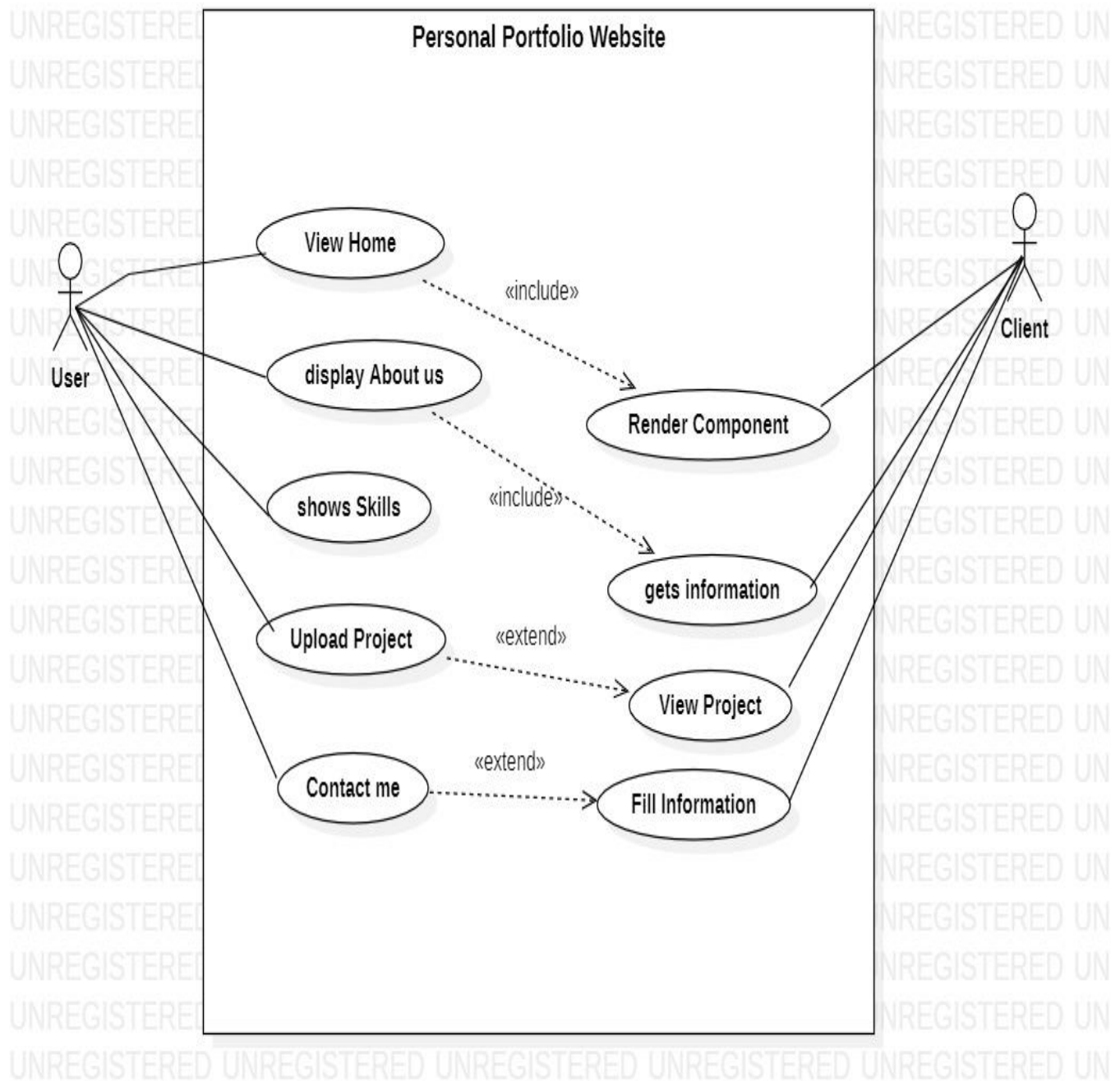
#### Extends:

A use case may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. It is used for showing optional use cases after a particular use case. The notation is a dashed arrow from the optional case to the base use case, with the label "«extend»"

#### Includes:

A use case may include another. Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The first use case often depends on the outcome of the included use case. The notation is a dashed arrow from the base case to the included use case, with the label "«include»".

### Use Case Diagram:



## **Activity Diagram:**

An activity diagram visually displays a series of actions or the flow of control in a system. Using activity diagrams in business process modelling is common. Additionally, they can outline the procedures in a use case diagram. Activities might be sequential or concurrent while being modelled. An activity diagram will always have a start (the starting state) and a finish (the final state) in both scenarios.

There are several methods to represent activities, flows, decisions, guards, merging events, and more in between.

### **Initial State or Start Point:**

Any activity diagram's first action state or starting point is shown as a tiny filled circle and an arrow.

### **Activity or Action State:**

An action state is a representation of an object's continuously acting state.

### **Action Flow:**

Action flows, also known as edges and routes, show how one action state changes into another. They are frequently represented with an arrowed line.

### **Synchronisation:**

To divide a single incoming flow into numerous concurrent flows, a fork node is employed. In an activity diagram, it is shown as a straight, somewhat thicker line.

### **Swimlanes:**

Swim lanes group related activities into one column.

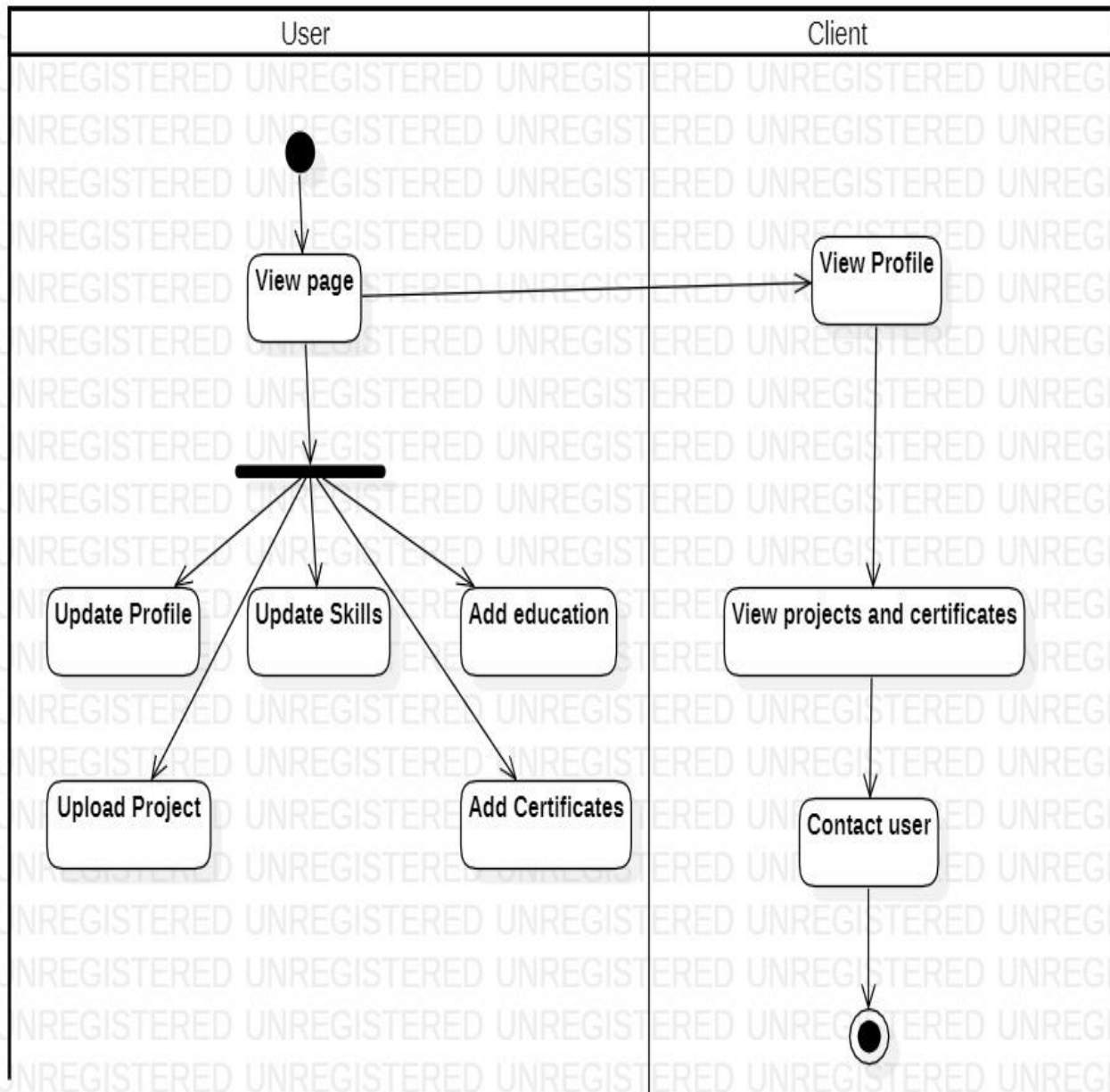
### **Final State or End Point:**

The ultimate action state is represented by an arrow pointing to a filled circle enclosed inside another circle.

### **Fork and Join Nodes:**

Forks and joins have the same notation: either a horizontal or vertical bar (the orientation is dependent on whether the control flow is running left to right or top to bottom). They indicate the start and end of concurrent threads of control. The following diagram shows an example of their use.

A join is different from a merge in that the join synchronizes two inflows and produces a single outflow. The outflow from a join cannot execute until all inflows have been received. A merge passes any control flows straight through it. If two or more inflows are received by a merge symbol, the action pointed to by its outflow is executed two or more times.

**Activity Diagram:**

## **Entity Relationship Diagram:**

An Entity-Relationship Diagram (ERD) is a graphical representation used in database design and modelling to illustrate the relationships between entities (such as tables in a relational database) and their attributes. ERDs help visualize the structure of a database and how different pieces of information are related to each other.

### **ELEMENTS OF ENTITY RELATIONSHIP DIAGRAM:**

#### **Entities:**

Entities represent objects or concepts in the real world, which are stored in a database. Each entity has attributes that describe the properties or characteristics of the entity.

#### **Attributes:**

Attributes are properties or characteristics of an entity that hold data. They are typically represented as ovals or ellipses in the diagram and are connected to their respective entities. Attributes help define what kind of data is stored for each entity.

#### **Relationships:**

Relationships define how entities are connected or associated with each other in the database. Relationships show how data from one entity is related to data in another entity. They are typically represented as lines connecting entities with diamond shapes that indicate the type of relationship.

## Entity Relationship Diagram:

Project { }
Name:
Image :
Projectlink :
Description :
Role :
imageTitle [ ]
title :
image :
timestamp :

Skills{ }
Languages :
Database :

Certifications { }
Title
Image

About { }
Title
Description
Image

EducationDetails { }
Organization Name :
Course Name :
Grade:
Duration :

Testimonial { }
Name
Company
image
Feedback

Contact { }
Name
email
message



## **Class Diagram:**

A class diagram is a type of static structure diagram in the Unified Modelling Language (UML) that represents the structure and relationships of classes in a system or software application. Class diagrams are commonly used in software engineering for modelling the classes, attributes, methods, and associations within a system's object-oriented design

### **ELEMENTS OF CLASS DIAGRAM:**

#### **Class**

A class represents a blueprint or template for creating objects. It defines the attributes (data members or fields) and methods (functions or operations) that objects of the class will have.

#### **Attributes**

Attributes are properties or data members of a class that represent the characteristics or state of objects. They are typically shown as variables or fields inside the class rectangle.

#### **Methods**

Methods are functions or operations that can be performed on objects of the class. They are usually shown as functions inside the class rectangle.

#### **Associations**

Associations represent relationships or connections between classes.

#### **Inheritance (Generalization)**

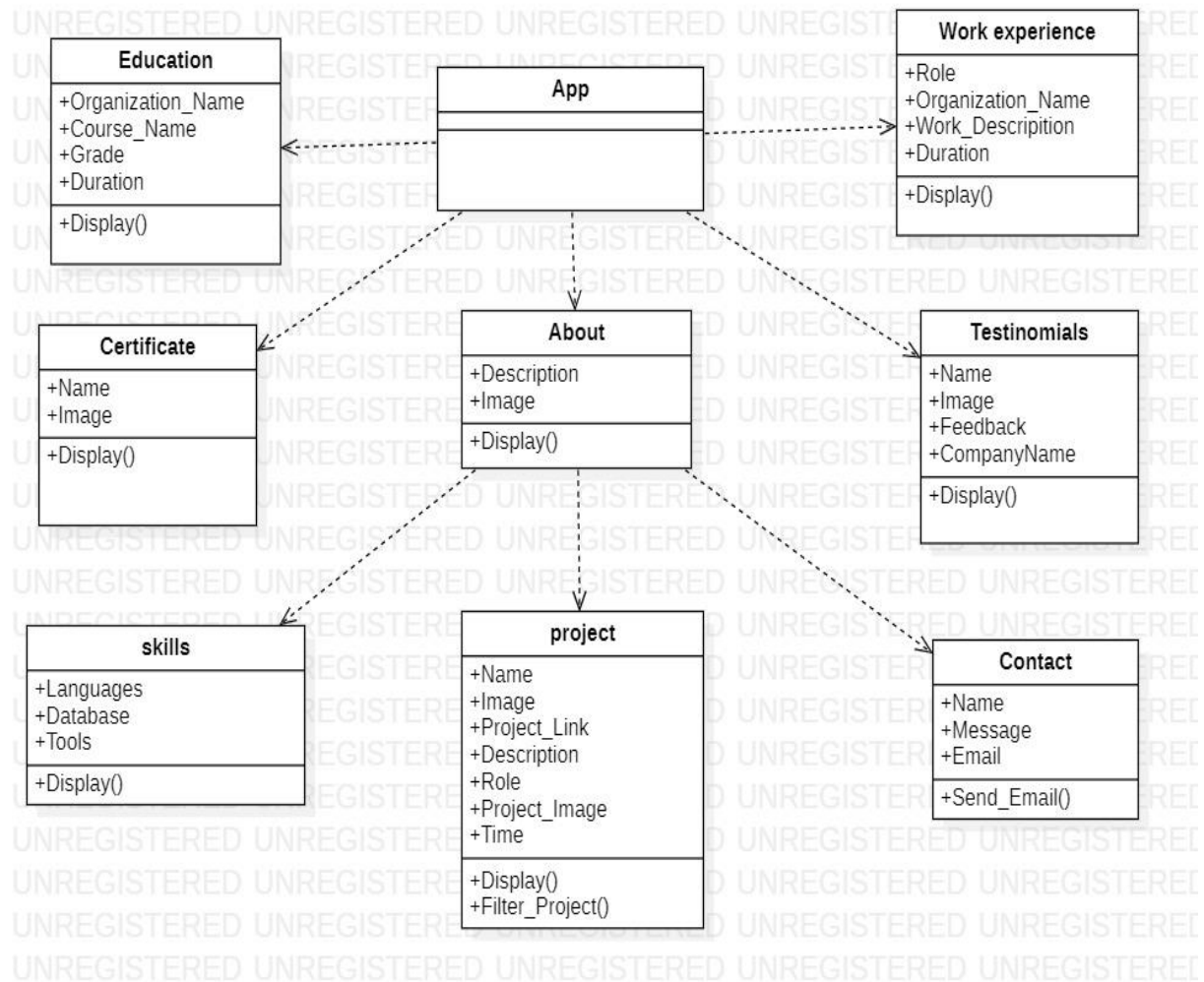
Inheritance represents the "is-a" relationship between classes. It shows that one class (the subclass or derived class) inherits the attributes and methods of another class (the superclass or base class).

#### **Aggregation and Composition**

Aggregation indicates a weaker relationship, where one class contains or is associated with other classes, but the contained objects can exist independently.

Composition indicates a stronger relationship, where the whole (composite) class owns and manages its parts

### Class Diagram:



## **SYSTEM IMPLEMENTATION**

### Test Case:

Test case Id	Module	Form	Test Condition	Steps to execute	Input test data	Expected Output	Actual Result	Status
TC_1	Project Section	Project Section Form	Filter out the project from the database and give result	1)visit website 2)click on project component	click on category	shows filter results	shows filter results	Pass
TC_2	Contact	Contact Form	To send mail with the help of Email Address	1)Enter valid name 2)Enter valid email address 3)Enter valid phone number 4) Click on send button	1)Invalid name 2)Invalid phone number 3)invalid Email	Invalid email,please try again!!	Invalid email, please try again!!	Pass

## **Sequence Diagram:**

sequence diagram is a type of Unified Modeling Language (UML) diagram that focuses on illustrating the interactions and chronological order of messages exchanged between objects or components within a system or between different systems. Sequence diagrams are commonly used in software engineering and system design to visualize the dynamic behavior of a system, particularly during the execution of a use case or scenario

### **ELEMENTS OF A SEQUENCE DIAGRAM:**

#### **Lifeline:**

A vertical dashed line representing an object or participant involved in the interaction. Each lifeline corresponds to an instance of a class or component and spans the duration of its involvement in the sequence.

#### **Activation Bar**

An activation bar is a horizontal line that extends from a lifeline to represent the period during which an object is actively processing or executing a specific action or operation.

#### **Message**

Messages are arrows that show the flow of communication between objects or components. They indicate the exchange of information or the invocation of operations between lifelines.

#### **Return Message**

A return message (indicated by a dashed arrow with a label) shows the response from the receiver back to the sender after an operation is executed.

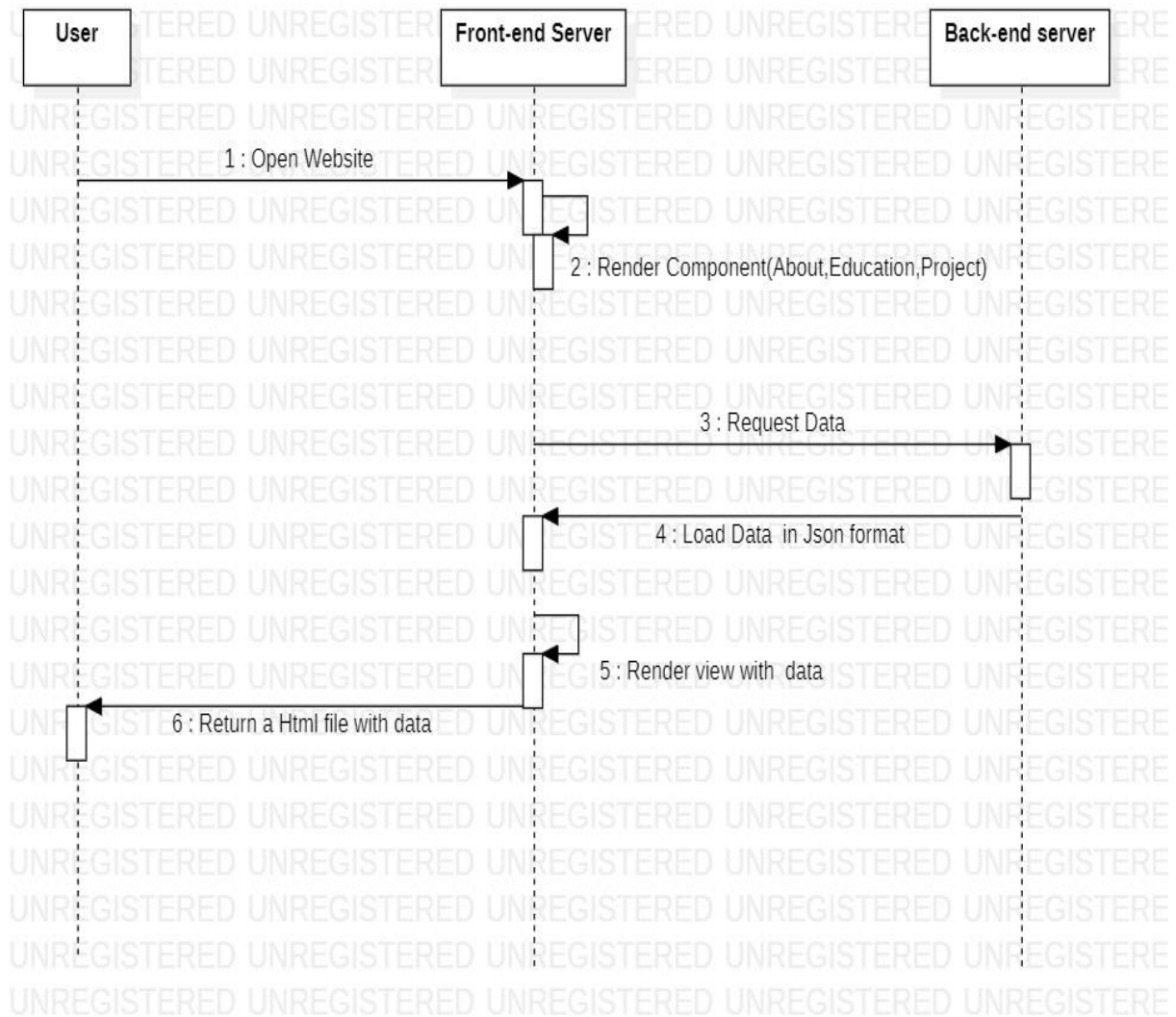
#### **Synchronous Message**

Denoted by a solid arrow, it represents a call where the sender waits for a response from the receiver before proceeding.

#### **Asynchronous Message**

Denoted by a dashed arrow, it represents a call where the sender doesn't wait for an immediate response and continues its execution

## Sequence Diagram:



## **Component Diagram:**

A component diagram is a type of Unified Modeling Language (UML) diagram used to represent the structural organization of a system or application into components, their relationships, and how they interact to form a complete system. Component diagrams are primarily used to model the high-level architecture of software systems and the dependencies between different parts of the system.

### ELEMENTS OF A SEQUENCE DIAGRAM:

#### Components

Components are modular, reusable, and replaceable parts of a system. They can represent classes, packages, libraries, subsystems, or physical modules, depending on the context. Components are typically depicted as rectangles with the name of the component inside.

#### Interfaces

Interfaces represent the contract or set of services that a component provides or requires from other components. Interfaces are shown as circles on the boundaries of the component, and they can be labelled to indicate the specific services or methods exposed.

#### Dependencies

Dependencies between components are depicted as arrows pointing from one component to another. These arrows show that one component depends on another, which implies that changes in the depended-upon component can affect the dependent component.

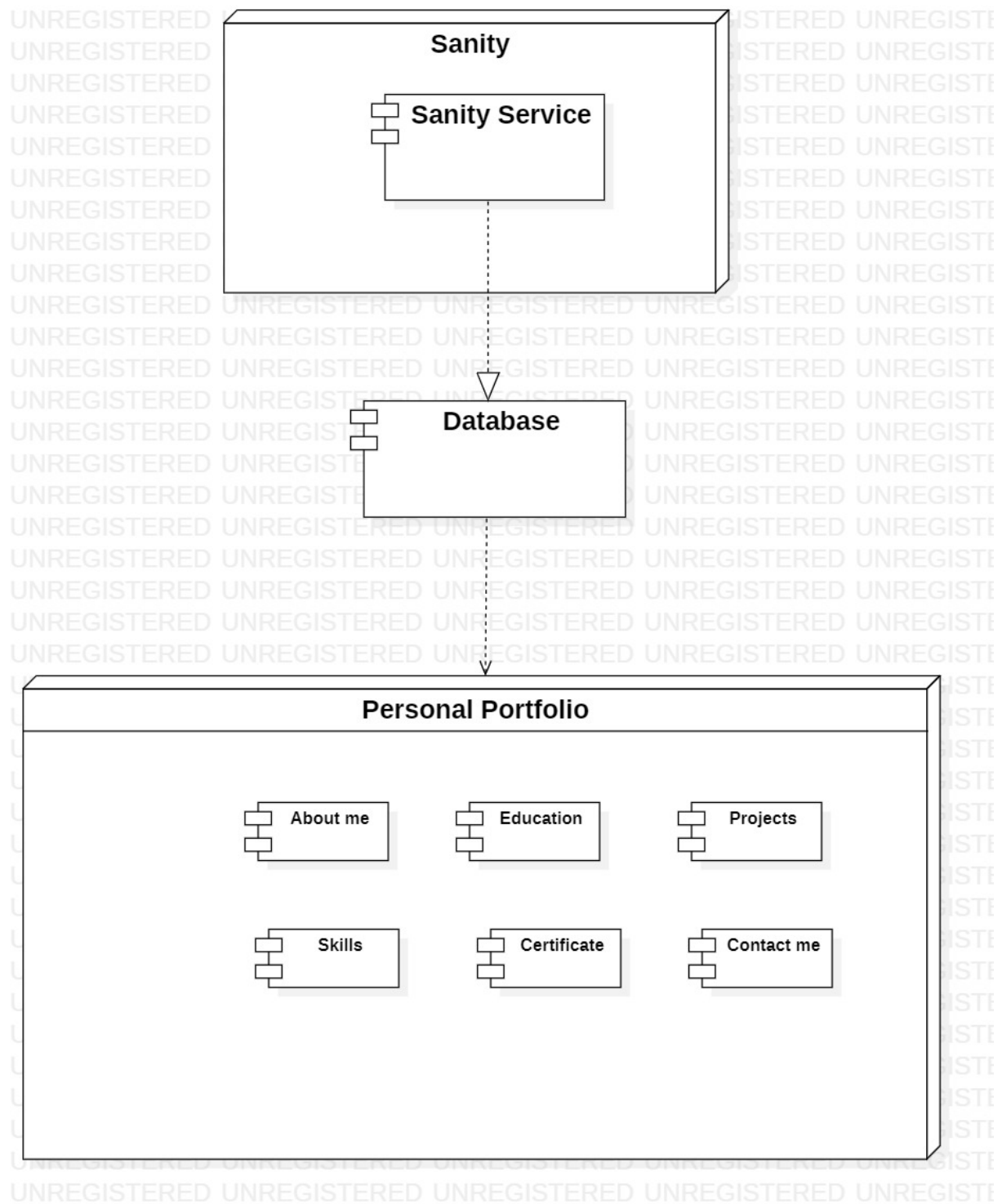
#### Associations

Associations between components can represent more flexible relationships between them, such as associations or connections between objects in object-oriented design. These are typically shown as lines connecting components without arrows.

#### Provided and Required Interfaces

Each component may have a set of provided interfaces (the services it offers) and required interfaces (the services it depends on from other components). Provided interfaces are usually shown inside the component, while required interfaces are shown as part of the component's boundary.

### Component Diagram:





## **Deployment Diagram:**

A deployment diagram is a type of Unified Modeling Language (UML) diagram that focuses on visualizing the physical deployment of software components and hardware nodes in a system or application. Deployment diagrams are used to illustrate how software artifacts (such as components, executables, and libraries) are distributed across hardware nodes (such as servers, computers, or devices) in a networked environment. These diagrams help software architects and developers understand how a system is deployed in the real-world infrastructure.

### **ELEMENTS OF A SEQUENCE DIAGRAM:**

#### **Nodes**

Nodes represent hardware devices or execution environments in a deployment, such as servers, computers, routers, or physical devices. Nodes are typically depicted as rectangles with the name of the node inside.

#### **Artifacts**

Artifacts represent software components, executable files, or libraries that are deployed on nodes. They are shown as rectangles with the name of the artifact inside. Artifacts can be associated with nodes to indicate where they are deployed.

#### **Communication Paths**

Communication paths (also known as communication links or connectors) represent the connections and communication channels between nodes. Communication paths are represented by lines connecting nodes.

#### **Associations**

Associations between nodes and artifacts indicate that an artifact is deployed on a particular node. These associations are typically represented by a dashed line with an arrow pointing from the node to the artifact.

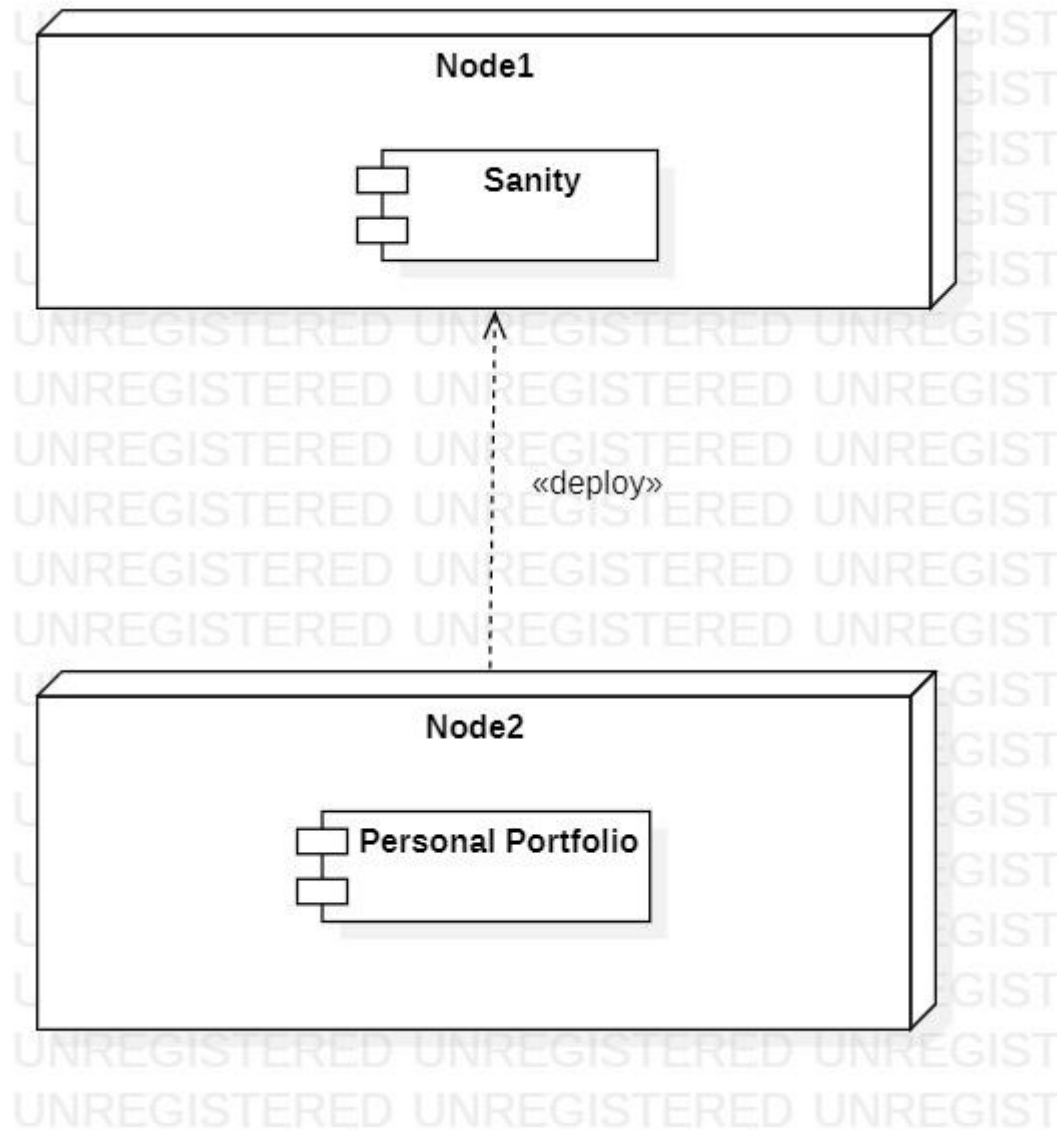
#### **Deployment Relationships**

Deployment relationships are used to specify how artifacts are deployed on nodes. There are two main types of deployment relationships:

#### **Deployment**

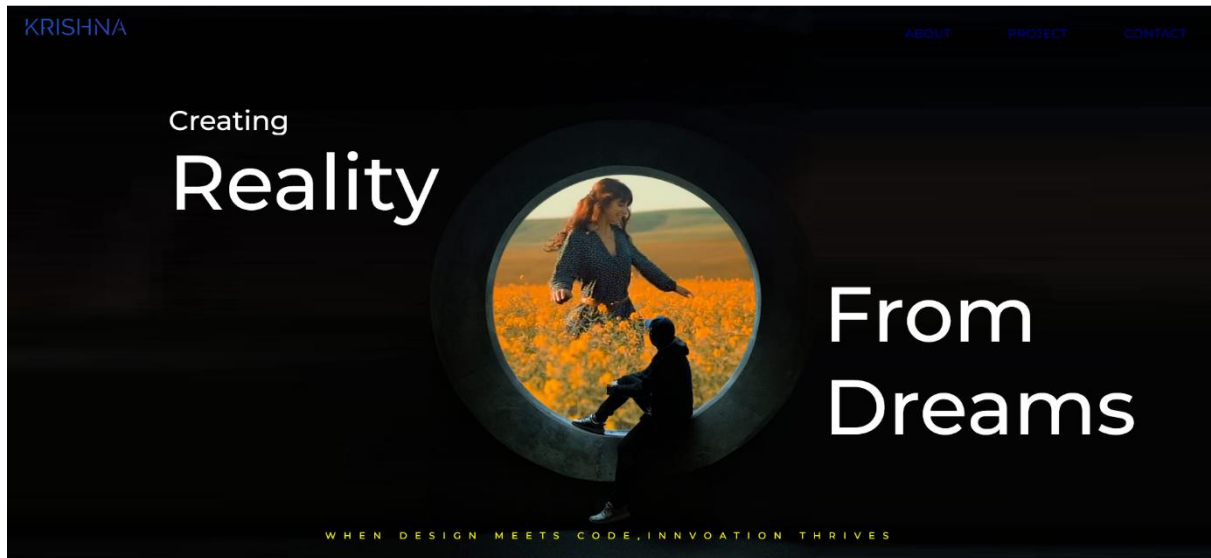
Represents a one-to-one relationship between an artifact and a node, indicating that a specific artifact is deployed on a specific node.

### Deployment Diagram:



## Images of Application:

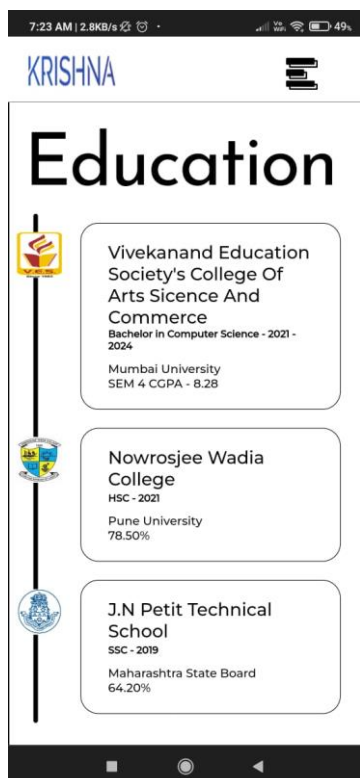
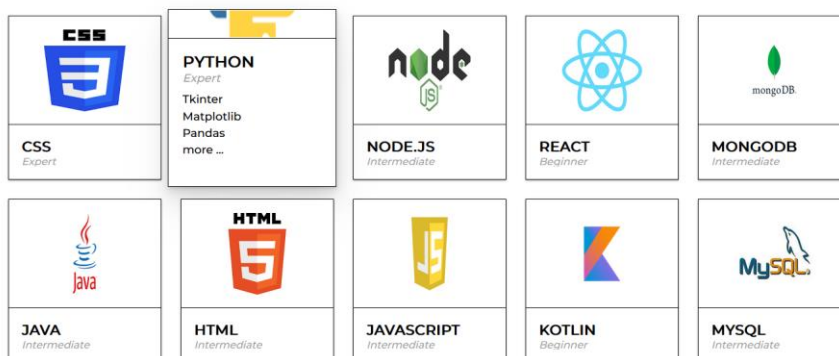
### Landing Page:



## Home Page:



# SKILLS

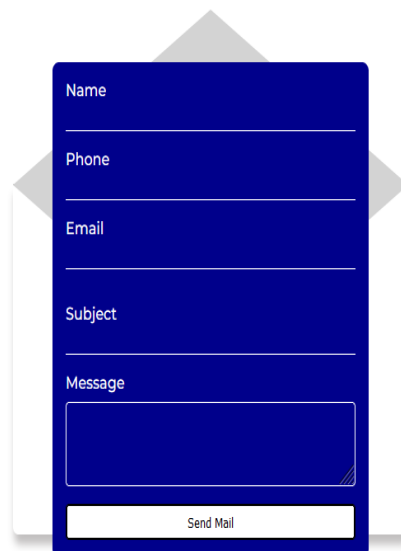


## Contact Page:

[ABOUT](#)[PROJECT](#)[CONTACT](#)

# Get In Touch

Let's Talk



Name

Phone

Email

Subject

Message

Send Mail

## Future Enhancements:

Builder: Portfolio Builder for everyone

Customize: AI customization Layout

## References:

Stack overflow:-

<https://stackoverflow.com/>

Design:-

<https://dennissnellenberg.com/>

Sanity: -

[https://www.youtube.com/@sanity\\_io](https://www.youtube.com/@sanity_io)

Email.js: -

<https://www.youtube.com/@ChaoCharles>

React.js: -

<https://legacy.reactjs.org/docs/getting-started.html>

SCSS Design: -

<https://www.youtube.com/@TheCoderCoder>