**Lecture Management System**

**“Lecture Stream”**

*A Project Report Submitted*

­­­­in Partial Fulfillment of the Requirements

for the Degree of

**Master of Computer Application**

*Submitted By*

**Name: Suraj Kumar**

**Roll No: 21MCA008**

****

**Faculty of Computing and Information Technology**

**Usha Martin University, Ranchi**

**January, 2023**

**BONAFIDE CERTIFCATE**

Certified that this project report titled Lecture Management named “Skill - up” is the bonafide work of suraj Kumar who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

(Signature of the concerned Supervisor of the Organization with Organization Seal)

(Certificate to be countersigned by the HOD.)

**DECLARATION**

I do hereby declare that this report entitled Lecture Management “Skill-up”, submitted by Mr. Suraj Kumar, bearing Roll No: 21MCA008 in the fulfillment of the requirement for the degree of Master of Computer Application to Usha Martin University, Ranchi, is my own and it is not submitted to any other institute.

**Name: Suraj Kumar**

**Roll No: 21MCA008**

**CERTIFICATE**

This is to certify that entitled E-Learning Application “Lecture Management System” being submitted by Suraj Kumar, bearing Roll No- 21MCA008, in the fulfillment of the requirement for the degree of Master of Computer Application to Usha Martin University, Ranchi, is a bonafide work carried out under my/our supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

**(Dr. Mohan Prakash) (Dr. Sharmistha Roy)**

**Co-Guide Guide**

**(Dr. Sharmistha Roy)**

**HOD External Examiner**

**ACKNOWLEDGEMENTS**

I express my deepest sense of gratitude to my guide **Dr. Sharmistha Roy** and co-guide **Dr. Mohan Prakash** Faculty of Computing and Information Technology, Usha Martin University, Ranchi, for suggesting the subject of work and constant supervision throughout this work. His/her co-operation and timely suggestions have been unparalleled stimuli for me to travel eventually towards the completion of this project report. Indeed his/her continuous involvement has helped me in bringing of this project work which otherwise would have remained a distant dream.

I am indeed thankful to **Dr. Sharmistha Roy**, HOD, Faculty of Computing and Information Technology, Usha Martin University, Ranchi, for giving me permission to carry out my project work. I would like to express my gratitude to all teaching and non-teaching staff members of Faculty of Computing and Information Technology, Usha Martin University, Ranchi, for their co-operation in my work.

**Name: Suraj Kumar**

**Roll No: 21MCA008**

**Abstract**

Online communication is ruling the world today. Especially during the COVID times we have seen a drastic closing of educational institutions, offices and other places of employment and exchange. There was an increased challenge of conducting discussions and communicating with people of all sections, increased problems of discourse like taking classes in schools, colleges and Universities due to lack of interactive methods accessible to the masses. Of course we need to say that there were some online tools for meeting with each other like Skype, Google duo and some other paid tools. However the use of these tools were restricted to a few people and it was difficult to organize big lengthy sessions using such methods. The first lock downs promoted software developers throughout the world to organize and develop better tools and methods for online communication among a large number of people. This led to the development of many user friendly software applications and the research to improve these continue till date. In this project we have tried to build a software that aims to provide streaming video for and users. This streaming videos maybe weed life or even maybe recorded for viewing later. The development work has mainly three parts the first one which consists of the front end of the software and has been built upon the very modern and popular JavaScript framework React. The second part is basically JSON server which stores the data about the different video-streams and which allows the admin to add, delete or modify streams as per the requirement. The third part consists of a video streaming server which actually implements the streaming part and enables the admin to stream a live video or recorded one for future requirements. The other users can view the videos as per the requirement but only the admin is allowed to make changes. The authentication system that has been used for this software is the Google auth API which allows authentication uniquely using email ID and Google password. The objective is to provide a simple and lightweight environment for one to many communication which may be suitable for purposes like taking classes or addressing gatherings

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Bonafied Certificate** | **……………………………………………………………** | **2** |
| **Declaration** | **………………………………………..............................** | **3** |
| **Certificate** | **……………………………………………………………** | **4** |
| **Acknowledgement**  **Abstract** | **…………………………………………………………………………………………………………………………** | **5**  **6** |

[Chapter 1 9](#_Toc125032718)

[INTRODUCTION 9](#_Toc125032719)

[1.1 Background 9](#_Toc125032720)

[1.2 Objectives 10](#_Toc125032721)

[1.2.1 Loyal Customers 10](#_Toc125032722)

[1.2.2. Brand Recognition 10](#_Toc125032723)

[1.2.3. Better Efficiency and Increased Revenues 10](#_Toc125032724)

[1.2.4. Detailed Analytics 11](#_Toc125032725)

[1.3 Nature and Scope 12](#_Toc125032726)

[Chapter 2 19](#_Toc125032727)

[TECHNOLOGIES 19](#_Toc125032728)

[2.1 Software Technologies: 19](#_Toc125032729)

[2.2 Hardware technologies: 19](#_Toc125032730)

[2.3 DFD Diagram 20](#_Toc125032731)

[2.4 ER Diagram 22](#_Toc125032732)

[Chapter 3 23](#_Toc125032733)

[REQUIREMENTS AND ANALYSIS 23](#_Toc125032734)

[3.1 SYSTEM ANALYSIS 23](#_Toc125032735)

[3.2 SOFTWARE REQUIREMENT SPECIFICATION 25](#_Toc125032736)

[3.2.1. Definition 26](#_Toc125032737)

[3.2.2. Purpose 26](#_Toc125032738)

[3.2.3. Scope 26](#_Toc125032739)

[Chapter 4 27](#_Toc125032740)

[System Design 27](#_Toc125032741)

[4.1 Basic Modules 27](#_Toc125032742)

[4.2 Data Design 27](#_Toc125032743)

[4.3 Security 28](#_Toc125032744)

[4.4 Testing Strategies 29](#_Toc125032745)

[CHAPTER 5 33](#_Toc125032746)

[IMPLEMENTATION AND TESTING 33](#_Toc125032747)

[5.1 Implementation Approach 33](#_Toc125032748)

[5.2 SYSTEM TESTING 38](#_Toc125032749)

[Chapter 6 45](#_Toc125032750)

[SNAPSHOT OF THE RUNNING MODEL 45](#_Toc125032751)

[Chapter 7 50](#_Toc125032752)

[LIMITATIONS, CONCLUSIONS AND FUTURE SCOPE 50](#_Toc125032753)

[7.1 LIMITATIONS 50](#_Toc125032754)

[7.2 FUTURE SCOPE 51](#_Toc125032755)

[7.3 CONCLUSION 51](#_Toc125032756)

[Chapter 8 54](#_Toc125032757)

[SOURCE CODE 54](#_Toc125032758)

[REFERENCES 85](#_Toc125032759)

# 

# Chapter 1

# INTRODUCTION

## 1.1 Background

An e-learning history timeline

Long before the internet was launched, distance courses were being offered to provide students with education on particular subjects or skills. In the 1840′s Isaac Pitman taught his pupils shorthand via correspondence. This form of symbolic writing was designed to improve writing speed and was popular amongst secretaries, journalists, and other individuals who did a great deal of note taking or writing. Pitman, who was a qualified teacher, was sent completed assignments by mail and he would then send his students more work to be finished using the same system.

In 1924, the first testing machine was invented. This device allowed students to test themselves. Then, in 1954, BF Skinner, a Harvard Professor, invented the “teaching machine”, which enabled schools to administer programmed instruction to their students. It wasn’t until 1960 however that the first computer based training program was introduced to the world. This computer-based training program (or CBT program) was known as PLATO-Programmed Logic for Automated Teaching Operations. It was originally designed for students attending the University of Illinois, but ended up being used in schools throughout the area.

The first online learning systems were really only set up to deliver information to students but as we entered the 70s online learning started to become more interactive. In Britain, the Open University was keen to take advantage of e-learning. Their system of education has always been primarily focused on learning at a distance. In the past, course materials were delivered by post and correspondence with tutors was via mail. With the internet, the Open University began to offer a wider range of interactive educational experiences as well as faster correspondence with students via email etc.

Online learning today

With the introduction of the computer and internet in the late 20th century, e-learning tools and delivery methods expanded. The first MAC in the 1980′s enabled individuals to have computers in their homes, making it easier for them to learn about particular subjects and develop certain skill sets. Then, in the following decade, virtual learning environments began to truly thrive, with people gaining access to a wealth of online information and e-learning opportunities.

By the early 90s, several schools had been set up that delivered courses online only, making the most of the internet and bringing education to people who wouldn’t previously have been able to attend a college due to geographical or time constraints. Technological advancements also helped educational establishments reduce the costs of distance learning, a saving that would also be passed on to the students – helping bring education to a wider audience.

In the 2000′s, businesses began using e-learning to train their employees. New and experienced workers alike now had the opportunity to improve upon their industry knowledge base and expand their skill sets. At home, individuals were granted access to programs that offered them the ability to earn online degrees and enrich their lives through expanded knowledge.

## 1.2 Objectives

There are certain goals when it comes to eLearning and some of these are to:

Enhance the quality of learning and teaching

Meet the learning style or needs of students

Improve the efficiency and effectiveness

Improve user-accessibility and time flexibility to engage learners in the learning process

eLearning is vast and an expanding platform with huge prospective in higher education. Since there are many challenges in making eLearning effective, it is important to know how to manage it and access to the resources. Take a minute and just imagine if one is not having the roadmap to guide from start to finish is actually like plunging into eLearning without an effective strategy because learners would be lost in the learning content..

**The Objectives of the App:**

1. The first, and main objective of the app is to get lecture and with related notes.
2. To offer students a flexible learning schedule without any time limit to complete the course.
3. To allows instructors to build online courses on their preferred topics.
4. To provide access to thousands of courses produced by independent instructors to millions of students..

**The Benefits of the App:**

1. Any new jobs that are posted on your Website will automatically update to the app so there is no duplication of work and admin time.
2. You can send push messages to all your users free of charge unlike texting software which has become expensive. So, there is literally no limit to the number you can send out.
3. You will save a massive amount of time making candidates aware of new jobs leaving you free to concentrate on other areas of your work. The app will improve and enhance your candidate resourcing ability.
4. You will be seen in the market as a company that is doing something new and is embracing technology.
5. As well as permanent positions, this is a great tool to utilize when filling temporary bookings, contract and interim assignments.

**How to Promote the App:**

Here are some ways in which you can promote our app. This is not an exhaustive list, but is designed to give you some initial ideas to start the process:

1. Include downloading the app as part of the candidate registration process.
2. Put details of the app on your candidate registration forms.
3. Promote the app on your web homepage, and on your jobs page on your website.
4. Promote the app on your email signatures. (This will be seen by Candidates and Clients)
5. Put details of the app on your paper pay slips or e-pay slips
6. Promote the app on all your job board advertising
7. Promote the app on your next business card re-print.
8. Include the app details on any marketing collateral you may have.
9. Run refer a friend campaigns to all existing users.
10. Create videos and start using them to promote your app and your business on Social Media.

## 1.3 Nature and Scope

* **Affordable:** There are many ways to start a new e-learning system with little to no overhead
* **Easy**: Sometimes you don’t even have to have your own product; some e-commerce shops work thanks to drop shipping
* **Options**: Whether you’re a tech expert or total newbie, there’s an e-commerce platform out there that caters to you

If you’re in the initial planning stages of putting up your e-commerce site, temper yourself to avoid picking a design before deciding on the essential types of e-commerce features you’ll need to incorporate. Some of these features are already built into certain e-commerce platforms, while others may need to be custom built.

Regardless of how you plan to incorporate them, here are the essential types of e-commerce features you need to be successful online.

**Ease of Use**

Simplicity is the mark of a good design, and 76% of consumers agree. They say that the most important characteristic of a good website is ease of use.

By ease of use, this means helping shoppers get what want they want faster and without the unnecessary complexity that could deter their eventual purchase. You can implement this concept by:

* Creating shopping categories
* Adding filters
* Incorporating comparison capabilities

Additionally, make it easy to find customer reviews, FAQs, and information about the types of e-commerce policies you abide by (like shipping and returns) to help buyers make decisions more quickly.

Invest in a website with a simple, clutter-free design and make sure that your website loads as quickly as people are browsing through it.

**Site Speed**

Site speed is an indicator of an easy to use website and draws a lot of parallels with the ability to convert visitors. Online sellers literally have seconds to make a sale as shrinking attention spans (8 seconds, smaller than a goldfish’s) influences e-commerce purchasing behavior.

51% of U.S. online shoppers cite slow site loading times as the top reason they abandon a purchase, and these slow loading websites cost the U.S. e-commerce market more than $500 billion annually. Based on data monitoring real user activity from 33 major retailers, an increase in site speed from 8 to 2 seconds can boost your conversion rate by 74%.

**Mobile Friendly**

Another of the major types of e-commerce features you’ll need to make your site easier to use is by optimizing for mobile.

People spend more than half their time on mobile phones (compared to computers), a trend that also applies to online shopping. If you still think that optimizing for mobile is optional—*don’t*. Google will penalize sites that aren’t optimized for mobile. Mobile websites fare better for SEO; a mobile-first user experience is one of the four major ranking signals.

If you have the resources, consider creating a mobile experience contained within an app. Of the reported 5 hours per day that people spend on their mobile phones, 92% of that is spent within apps.

Here are some tips for how to design for mobile if you’re just getting started.

**Related Items**

As an online entrepreneur, your primary motivation for starting an e-commerce store might have something to do with earning money. Even if it’s #2 or #3 on your list of motivations, earning money is still important to keeping your business afloat.

Help increase order totals by connecting visitors to more of your products through a related items function. You can do this through similar product categories, “people who bought this item also searched for”, and comparison functions.

To put this into context, say for example that you’re selling laptops. Increase sales by suggesting related accessories: laptop skins, bags, and so on.

**Variety of Payment Getaways**

Depending on what you sell, where, an e-commerce site should also provide a myriad of options for payment methods. Some trust certain options over others and some countries don’t support certain payment methods, so it’s important to know your audience before settling on your options.

As an online store owner, you must make the whole transaction (payment especially!) as easy as possible for your customers. Not having a customer’s preferred payment method could spell the difference between a purchase and an abandoned cart.

While on the topic, make sure that your site is PCI (payment card industry) compliant as an extra security measure to build trust with your customers. If you haven’t yet decided on an e-commerce platform, make sure that the one you decide on is capable of supporting multiple payment options.

**User Reviews**

The major pitfall of online shopping is the fact that you cannot physically touch or experience a product.

Because of this, including functionality for user reviews is important. Reviews may be the only thing online shoppers have to rely on (besides information provided by you, which can be seen as biased) to make a purchase decision.

92% of shoppers read reviews. In fact, the star rating is the first factor customers consider when judging a website. 71% of consumers say they use online reviews to decide whether or not to make a purchase, and consider online reviews more credible than they would a salesperson (which is completely understandable).

Reviews don’t just influence purchase decisions, but also have up to 10% impact on businesses search rankings, so don’t be afraid to ask for one as part of a post-purchase email sequence. Luckily, 70% of customers are willing to leave an online review when asked.

And before you start worrying about the possibilities of public negative feedback, you’ll be happy to hear this: 52% of customers trust a product more when it has bad reviews.

**High-Resolution Photo and/or Video**

A study by Invodo found that 92.6% of consumers claim that visuals are the most influential factor affecting purchase decisions. Though most people feel a need to physically touch items before buying them, great product photos on an e-commerce product page are the next best thing (besides the aforementioned reviews).

The photos you display should be high-resolution but optimized so that they do not affect site load speed. Additionally, photos should also be zoomable so people can look at details.

But photos shouldn’t be the only types of e-commerce media you share; video is beneficial as well. According to Invodo:

* Video viewers are 1.7x more likely to buy
* Up to 4x as many shoppers prefer product videos to the text description
* Sites with video on most product pages had a 68% higher average order value than sites with video on a few product pages
* When researching a purchase decision, 80% of millennials consider video content
* 7 out of 10 millennials are likely to watch a company video when online shopping

Video also contributes to the page’s SEO ranking since there’s low competition in regard to this type of media.

**Security Features**

E-commerce sites are prime targets for cybercriminals, who can steal your sensitive information as well as your customers’.

Start with a secure e-commerce platform that supports HTTPS and SSL, which allows secure payment and checkout. This is an especially important e-commerce feature, since Google Chrome now requires HTTPS for sites that collect sensitive information—marking those that don’t as non-secure.

Other must-have security features include:

* **Two-factor authentication:** Adds an extra layer of security by requiring username/password and a system-generated code sent via email or text
* **Firewall:** A wall that blocks malicious traffic and permits authorized traffic

HubSpot shares other basic considerations when it comes to types of e-commerce security features.

**SEO**

You can affect your e-commerce website’s SEO without it being a built-in feature, though some platforms operate off of a better baseline than others.

SEO is very broad and multifaceted, involving site architecture, on-page SEO, technical SEO, link building, content and more. There are plenty of SEO-friendly e-commerce options but E-commerce Platforms provides an in-depth guide to the top picks.

**Social Media Integration**

Social media integration hasn’t always been a priority when it came to e-commerce. It started with social share buttons, then grew to using social as a way for users to log in to their accounts. These social components can contribute to off-page SEO.

With new social features like Pinterest buyable pins and Instagram’s shoppable feed, it becomes easier to push product outside of your e-commerce store. When those browsing their social feeds see your pin or Instagram post and click on it, they will be directed to your site to complete the purchase.

**Membership Site Capabilities**

The ability to add customer accounts helps your e-commerce shop by:

* Collecting customer information (so you can send them promotional materials advertising sales and other special deals)
* Ease of access for the customer the next time they purchase something (encouraging repeat business)
* Provide account security for transactions

Some e-commerce platforms already have this functionality built-in, but if you’re building a website from scratch, it’s important to include this functionality.

**Customer Support**

Customer support is one of the most important types of e-commerce features to consider for online selling success. Some e-commerce platforms offer related functionality as part of a package (which you pay monthly or annually), but if you’re building your website from scratch, you’ll want to look into live chat solutions.

Live chat is an incredibly convenient solution for providing customer service, as it provides customers instant access to assistance. Wait times are also usually shorter than a call center, and both customers and live chat agents can multitask while participating in a live chat.

42% of customers said live chat is their preferred support method. Similarly, 63% of customers said that they were most likely to return to a website that offered live chat.

Live chat also helps companies cut down costs on their customer service expenses. Unlike voice calls, which require at least one dedicated customer service representative per customer call, live chat operators can answer the queries of multiple customers at once—without sacrificing the quality of their interactions.

Additionally, live chat also helps advance customer purchases. 38% of customers say that they made a purchase as a result of a live chat session.

**Clear Set of Policies**

Being transparent with policies helps improve trust in your online shop. It informs the customers of the set of guidelines you adhere to, and ensures customers that if they ever need something, they can always ask you about it.

Consider including these policies on your e-commerce store:

* **FAQ:** More complex purchases need detailed information, and instead of answering the same queries over again, try to collate common questions on a public page.
* **Return & Exchange Policies:** There is always a risk when ordering online, especially when purchasing products that come in different sizes. A returns page is important because:
  + At least 30% of all products ordered online are returned
  + 34% of shoppers are put off by difficult return policies
  + 67% of shoppers check the returns page before making a purchase
  + 92% of consumers will buy again if product return process is easy
* **Privacy Policies:** A privacy policy lets customers know that you won’t be sharing their data. This is for their peace of mind. Furthermore, these policies are required by law for e-commerce websites.
* **Shipping Charge and Options:** Unexpected shipping costs are the #1 cause of abandoned shopping carts, so it’s critical to include shipping information where your customers can see it (usually a banner at the top of the page). Websites that advertise free shipping attract customers that spend 30% more per order. On a related note, your shipping information shouldn’t be located in just one place, so repeat relevant information at checkout. Having a shipping charge calculator based on zip code at checkout will help prevent shock.

On a final note, make these policies easy to access and easy to read.

**Contact Page**

When it comes to running an online shop, nothing gives customers more reassurance that the website can be trusted than a professional-looking contact page. This is most true when selling technical or expensive, high-end products.

**Don’t hide your contact information.** Make it easy for customers to get in touch and include several channels to do so by sharing relevant phone numbers, emails, and an address. Don’t forget to add social links to your pages and Google Maps for directions if there’s a physical store location (this will help with local SEO).

**Analytics and Reporting**

Analytics and reporting features are what gives you an idea of which of your products are selling, which strategies are effective, and where traffic is coming from. Using reports from Google Analytics, you can improve your e-commerce conversion rate.

**Afterword**

It’s easier than ever to start an online business—the e-commerce route is accessible to anyone with a computer and internet connection.

But running an e-commerce store takes a lot of work. There’s a lot to think about and a lot of features you’ll need to successfully make sales. This list covers the basics but the nature of your business will determine additional components necessary for driving sales.

# Chapter 2

# TECHNOLOGIES

The following are the system requirements to run our project.

## 2.1 Software Technologies:

**Front End:** Expo Go, Web Browser (Internet explorer, Mozilla Firefox or Chrome), Command Prompt, Visual Studio Code, Android Studio.

**Back End Database server :** Google Firebase

**Platform**  : Android.

## 2.2 Hardware technologies:

**For Computers:**

**RAM :** 4 GB or more

**PC :** **Intel**® **Core**™ **i3**

**HDD : 150GB or more**

**Monitor :** 15” at least VGA/SVGA/PGA (color preferred)

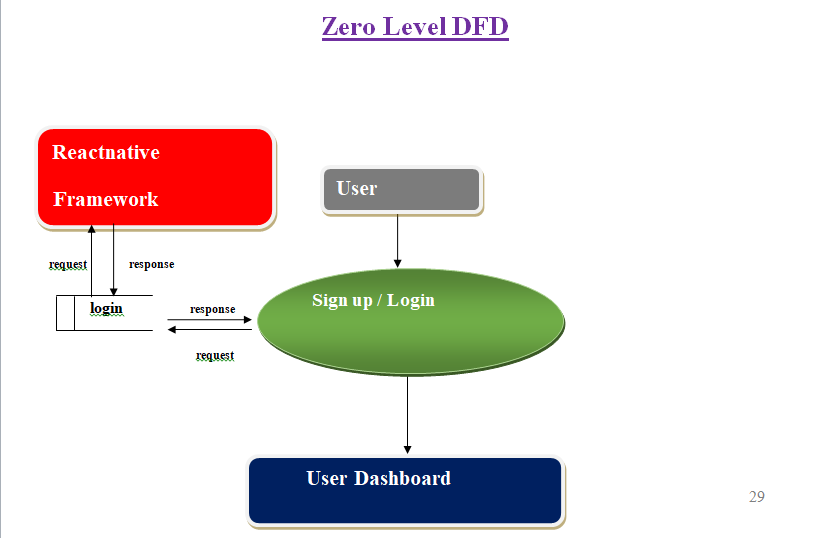
**For Android:**

Ram: 2GB or more

Processor: MediaTek P22 or above(recommended)

Operating System: Above Android 4.0

## 2.3 DFD Diagram



#### Fig 2: Zero Level DFD

LEVEL 1 DFD

**User Dashboard**

#### Fig 3: First Level DFD

## 2.4 ER Diagram

Add Product to cart

Product List

Buy Now

**E-Commerce App**

Purchase

User Login

Login

Stop

#### Fig 4:ER Diagram

# Chapter 3

# REQUIREMENTS AND ANALYSIS

## 3.1 SYSTEM ANALYSIS

System analysis is the reduction of the entire system by studying the various operations performed and their relationships within the system.

It consists of following terminology: -

1. system planning and the initial investigation
2. information gathering
3. the tools of structural analysis
4. feasibility study
5. cost analysis

*1. System planning and the initial investigation: -*

The most critical phase of managing system project is planning. To launch a system investigation. We need a master plan detailing the steps to be taken and the outcome expected. The first step is to determine project requirements. This is an important step, and one cannot ignore it because this is a solid base as requirement specifications, and will help for future reference when the product is ready that it has been developed as per requirement specified by the user.

*2.Information gathering: -*

The key part of the system analysis is gathering information about the present system. The system analyst must know what information to gather. Where to find it, how to collect it and what to make of it. The proper use of the tools for gathering information is the key to successful analysis. The tools are traditional interview, questionnaire and on-site observation. These tools when learned help analyse and assess the effectiveness of the present system and provide the ground work for recommending a candidate system.

*3. Structural Analysis: -*

Structural analysis is set of technique and graphical tools that allow the analyst to develop a new kind of system specification that easily understandable to the user.

Structured tools such as the dataflow diagram, E-R Diagram, flow chart provide alternative ways of designing a candidate system. In real life application a combination of the traditional and structural tools is used.

*4.Feasibility Analysis: -*

An important objective of conducting the system analysis is the determination of the feasibility. All projects are feasible if given unlimited resources and infinite time. But our systems have scarcity of resources and difficult delivery dates. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. Feasibility and risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. During system engineering, however, we concentrate our attention on three primary areas of interest:

*a). Economic Feasibility*

It is an evaluation of development of cost weighed against the ultimate income or benefit derived from the developed system. Economic justification includes a broad range of concerns that include cost–benefit analysis, long-term corporate income strategies, cost of resources needed for development.

Proposed system is developed by utilizing the available resources, it requires no much extra manpower, and SSI is a research and training center who’s objective is to provide more information on the Internet. By utilizing the proposed system, the organization will be more benefited compared to its costs. So, this is economically feasible.

Since the cost involved for the proposed system is less because the organization had already equipped with required resources like operating system software, Application software and Hardware.

*b). Technical Feasibility*

It is a study of function, performance and constraints that may affect the ability to achieve an acceptable system. The proposed system is Web based system and all the technical requirements are available with the organization and the Internet technology is available for most of the people in this World. So, the proposed system will definitely work with the current equipment, existing software technology, and available personnel.

The technologies required to accomplish system function are Web technologies and browsers. The methods, Algorithms, or processes required are normal and the development risk is less. The proposed system is developed in such a way that, it is simple enough to understand and manipulate.

*c). Operational Feasibility*

Will the system be used if it is developed and implemented? Will there be resistance from users that will undermine the possible application benefits?

By considering the various factors, the proposed system gives high performance, it is reliable, maintainable, and its productivity is high. Hence the proposed system is feasible.

*5. Cost analysis: -*

Data gathering, traditional or structure is only one part of system analysis.

The next steps are examining the data gathered, accessing the situations, looking at the alternative and recommending solutions the cost and benefits of each alternatives guides the selections of the best system for the job. Cost and benefits may be tangible or intangible, direct or indirect, fixed or variable. Cost estimates also takes into consideration hardware, personal, facility and supply cost for trial evaluation. This identifies the cost and categories them for analysis.

## 3.2 SOFTWARE REQUIREMENT SPECIFICATION

Requirement documentation is very important activity after the requirements elicitation and analysis. This is the way to represent requirements in a consistent format. Requirements document is called Software Requirements (SRS).

This SRS is a specification for a particular software product, program or set of programs that performs certain functions in a specific environment. It serves a number of purposes depending on who is writing it. First, the SRS could be written by a system. Second, the SRS could be written by a developer of a system. The two scenarios create entirely different purpose for the document. First case, SRS is used to define the needs and expectations of the users. The second case, SRS is written for different purpose and serve as a contract document between customer and developer.

This reduces the probability of the customer being disappointed with the final product.

**Nature of the SRS: -**

The basic issues that SRS writer(s) shall address are the following.

1. Functionality: - what the software is supposed to do?
2. External interfaces: - How does the software interact with people, the systems hardware, and other hardware and other software?
3. Performance: - what is the speed availability, response time, recovery time etc. of various software functions?
4. Attributes: -what are considerations for portability, correctness, maintainability, security, reliability etc?

3.2.1. Definition:

The app is focuses to serve the local small businesses and encourage them to use this simple app so that they can take their business online. The app is extremely handy and less educated person can also be able to use this app with ease. Anyone can add their any type of product to sell them and anyone will be able to buy that product easily through this app.

3.2.2. Purpose:

a) *Interface:* The interface of our e learning application is very simple to use and the user interface is handy. Look and feel of the app is attractive. Any new person can easily use this app with very little or no training.

c) *Hardware*: A simple smartphone is enough to run this application. In other words, low specification hardware can be used for running this app.

d) *Performance*: The application is very fast and it runs in a simple smartphone with ease. No extra local device data is needed to run this application since all the products will be stored online in a google firebase database. Products will load as user scroll through the pages.

3.2.3. Scope:

* 1. *Proposed System description*: The seller will be able to add products as there wish and set a price for individual products and edit their prices and product description according to their needs. The user(buyer) will be able to purchase these products according to there needs whenever they want. Finally, this is a great e learning application for buyers and seller to satisfy their needs.
  2. *Functional requirements of the software*: In order to develop such a kind of software and to use it, the developer and the user has to use the most modern technology that can be made available. And these technologies are made possible using React Native to build this application. React native is a great JavaScript framework to build native apps.
  3. *Assumptions and dependency:* Assumptions are that the logic and data supplied to the system are correct and the internet connections are suitably fast enough for the required interface. At the same time, it is assumed that the users are accessible to the interface and have sufficient technical skills to use them.

# Chapter 4

# System Design

## Basic Modules

Native Module and Native Components are our stable technologies used by the legacy architecture. They will be deprecated in the future when the New Architecture will be stable. The New Architecture uses Turbo Native Module and Fabric Native Components to achieve similar result.

React Native does not expose a JavaScript API to communicate with the native calendar libraries. However, through native modules, you can write native code that communicates with native calendar APIs. Then you can invoke that native code through JavaScript in your React Native application.

Native Modules Setup

There are two ways to module for your React Native Application:

1. Directly within your React Native application’s iOS/Android projects
2. As a NPM package that can be installed as a dependency by your/other React Native applications.

## 4.2 Data Design

**Composition**

The key feature of React is composition of components. Components written by different people should work well together. It is important to us that you can add functionality to a component without causing rippling changes throughout the codebase.

**Common Abstraction**

In general we resist adding features that can be implemented in userland. We don’t want to bloat your apps with useless library code. if React didn’t provide support for local state or lifecycle methods, people would create custom abstractions for them. When there are multiple abstractions competing, React can’t enforce or take advantage of the properties of either of them. It has to work with the lowest common denominator.

**Escape Hatches**

React is pragmatic. It is driven by the needs of the products written at Facebook. While it is influenced by some paradigms that are not yet fully mainstream such as functional programming, staying accessible to a wide range of developers with different skills and experience levels is an explicit goal of the project.

**Interoperability**

We place high value in interoperability with existing systems and gradual adoption. Facebook has a massive non-React codebase. Its website uses a mix of a server-side component system called XHP, internal UI libraries that came before React, and React itself. It is important to us that any product team can start using React for a small feature rather than rewrite their code to bet on it.

**Configuration**

 It is occasionally requested that we implement a function like React. Configure(options) or React. Register(component). However, this poses multiple problems, and we are not aware of good solutions to them.

**Implementation**

 We are much less concerned with the implementation being elegant. The real world is far from perfect, and to a reasonable extent we prefer to put the ugly code into the library if it means the user does not have to write it. When we evaluate new code, we are looking for an implementation that is correct, performant and affords a good developer experience. Elegance is secondary.

## 4.3 Security

Security is often overlooked when building apps. It is true that it is impossible to build software that is completely impenetrable—we’ve yet to invent a completely impenetrable lock (bank vaults do, after all, still get broken into). However, the probability of falling victim to a malicious attack or being exposed for a security vulnerability is inversely proportional to the effort you’re willing to put in to protecting your application against any such eventuality. Although an ordinary padlock is pickable, it is still much harder to get past than a cabinet hook!

**Android - Secure Shared Preferences**[**​**](https://reactnative.dev/docs/security#android---secure-shared-preferences)

Shared Preferences is the Android equivalent for a persistent key-value data store. **Data in Shared Preferences is not encrypted by default**, but Encrypted Shared Preferences wraps the Shared Preferences class for Android, and automatically encrypts keys and values.

**Android - KeyStore**[**​**](https://reactnative.dev/docs/security#android---keystore)

The Android KeyStore system lets you store cryptographic keys in a container to make it more difficult to extract from the device.

In order to use iOS Keychain services or Android Secure Shared Preferences, you can either write a bridge yourself or use a library which wraps them for you and provides a unified API at your own risk. Some libraries to consider:

* expo-secure-store
* react-native-encrypted-storage - uses Keychain on iOS and Encrypted Shared Preferences on Android.
* react-native-keychain
* react-native-sensitive-info - secure for iOS, but uses Android Shared Preferences for Android (which is not secure by default). There is however a branch that uses Android KeyStore.
  + redux-persist-sensitive-storage - wraps react-native-sensitive-info for Redux.

## 4.4 Testing Strategies

As your codebase expands, small errors and edge cases you don’t expect can cascade into larger failures. Bugs lead to bad user experience and ultimately, business losses. One way to prevent fragile programming is to test your code before releasing it into the wild.

In this guide, we will cover different, automated ways to ensure your app works as expected, ranging from static analysis to end-to-end tests.

**Static Test Analysis**

* **Linters** analyze code to catch common errors such as unused code and to help avoid pitfalls, to flag style guide no-nos like using tabs instead of spaces (or vice versa, depending on your configuration).
* **Type checking** ensures that the construct you’re passing to a function matches what the function was designed to accept, preventing passing a string to a counting function that expects a number, for instance.

**Writing Tests**

After writing testable code, it’s time to write some actual tests! The default template of React Native ships with Jest testing framework. It includes a preset that's tailored to this environment so you can get productive without tweaking the configuration and mocks straight away—more on mocks shortly. You can use Jest to write all types of tests featured in this guide.

**Structuring Tests**

The test is described by the string passed to the it functions. Take good care writing the description so that it’s clear what is being tested. Do your best to cover the following:

1. **Given** - some precondition
2. **When** - some action executed by the function that you’re testing
3. **Then** - the expected outcome

This is also known as AAA (Arrange, Act, Assert).

**Unit tests**

Unit tests cover the smallest parts of code, like individual functions or classes.

When the object being tested has any dependencies, you’ll often need to mock them out, as described in the next paragraph.

The great thing about unit tests is that they are quick to write and run. Therefore, as you work, you get fast feedback about whether your tests are passing. Jest even has an option to continuously run tests that are related to code you’re editing.

**Mocking**

Sometimes, when your tested objects have external dependencies, you’ll want to “mock them out.” “Mocking” is when you replace some dependency of your code with your own implementation.

* It could make the tests slow and unstable (because of the network requests involved)
* The service may return different data every time you run the test
* Third party services can go offline when you really need to run tests!

**Integration Tests**

When writing larger software systems, individual pieces of it need to interact with each other. In unit testing, if your unit depends on another one, you’ll sometimes end up mocking the dependency, replacing it with a fake one.

In integration testing, real individual units are combined (same as in your app) and tested together to ensure that their cooperation works as expected. This is not to say that mocking does not happen here: you’ll still need mocks (for example, to mock communication with a weather service), but you'll need them much less than in unit testing.

**Component Tests**

React components are responsible for rendering your app, and users will directly interact with their output. Even if your app's business logic has high testing coverage and is correct, without component tests you may still deliver a broken UI to your users. Component tests could fall into both unit and integration testing, but because they are such a core part of React Native, we'll cover them separately.

For testing React components, there are two things you may want to test:

* Interaction: to ensure the component behaves correctly when interacted with by a user (eg. when user presses a button)
* Rendering: to ensure the component render output used by React is correct (eg. the button's appearance and placement in the UI)

**End-to-End Tests**

In end-to-end (E2E) tests, you verify your app is working as expected on a device (or a simulator / emulator) from the user perspective.

This is done by building your app in the release configuration and running the tests against it. In E2E tests, you no longer think about React components, React Native APIs, Redux stores or any business logic. That is not the purpose of E2E tests and those are not even accessible to you during E2E testing.

Instead, E2E testing libraries allow you to find and control elements in the screen of your app: for example, you can *actually* tap buttons or insert text into Text Inputs the same way a real user would. Then you can make assertions about whether or not a certain element exists in the app’s screen, whether or not it’s visible, what text it contains, and so on.

E2E tests give you the highest possible confidence that part of your app is working. The tradeoffs include:

* writing them is more time consuming compared to the other types of tests
* they are slower to run
* they are more prone to flakiness (a "flaky" test is a test which randomly passes and fails without any change to code).

# CHAPTER 5

# IMPLEMENTATION AND TESTING

## 5.1 Implementation Approach

1. Installing Node.js that enable us to work with React Native.

**Setting up the Node Development Environment**

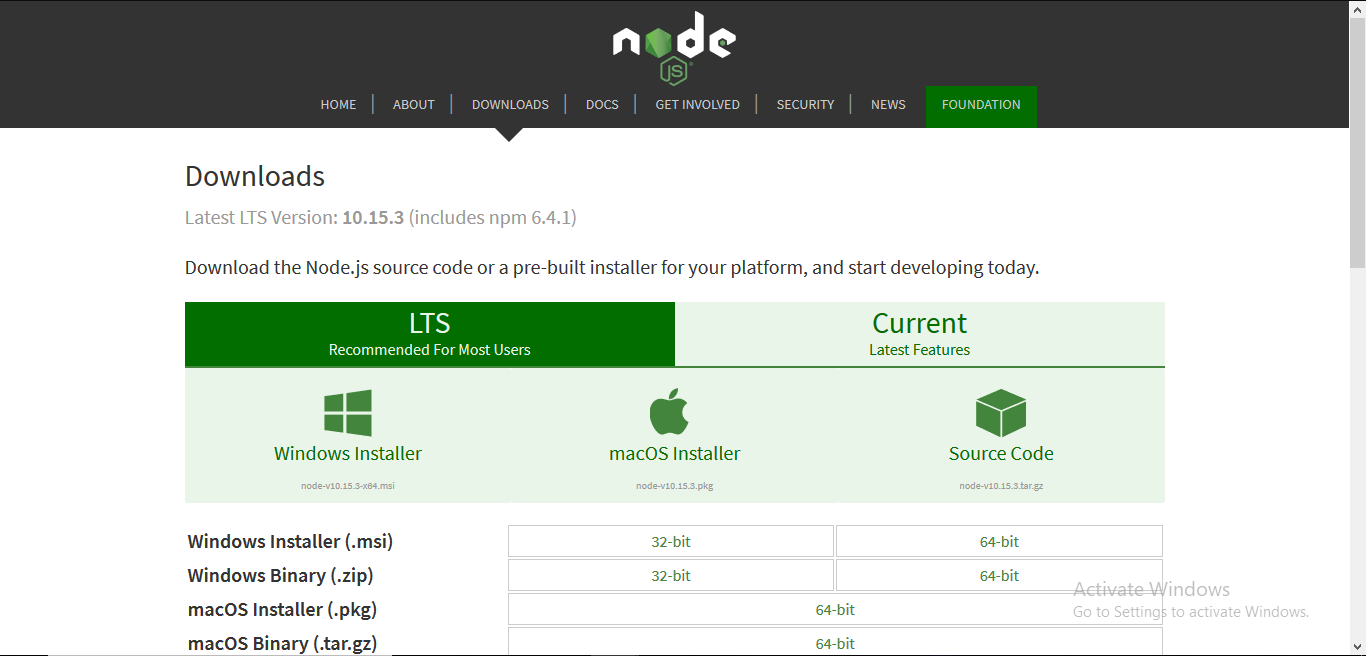
The Node can be installed in multiple ways on a computer. The approach used by you depends on the existing development environment in the system. There are different package installer for different environments. You can install Node by grabbing a copy of the source code and compiling the application. Another way of installing Node is by cloning the GIT repository in all the three environments and then installing it on the system.

**Installing Node On Windows (WINDOWS 10):**

You have to follow the following steps to install the Node.js on your Windows :

**Step-1:** Downloading the Node.js ‘.msi’ installer.

The first step to install Node.js on windows is to download the installer. Visit the official Node.js website i.e) <https://nodejs.org/en/download/> and download the .msi file according to your system environment (32-bit & 64-bit). An MSI installer will be downloaded on your system.



#### Fig 5 : Node

**Step-2:** Running the Node.js installer.

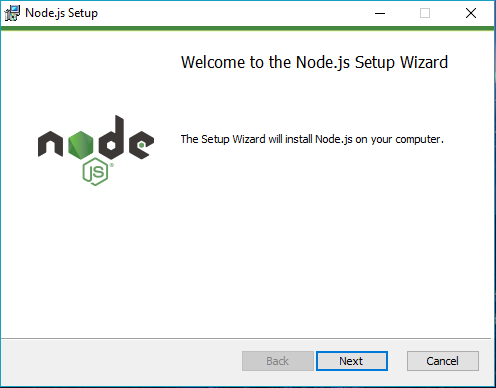
Now you need to install the node.js installer on your PC. You need to follow the following steps for the Node.js to be installed:-

* Double click on the .msi installer.

***The Node.js Setup wizard will open.***

* Welcome To Node.js Setup Wizard.

***Select “Next”***

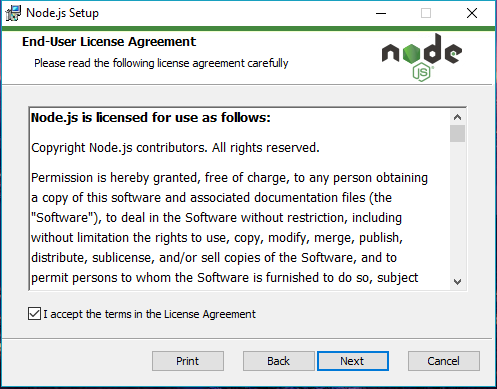


#### Fig 6: Node Installation

* After clicking “Next”, End-User License Agreement (EULA) will open.

***Check “I accept the terms in the License Agreement”***

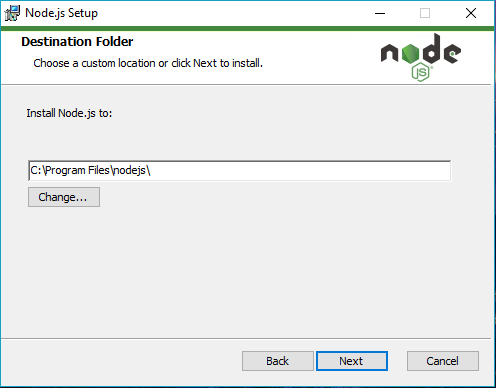
***Select “Next”***



#### Fig 7 : Node Setup

* Destination Folder

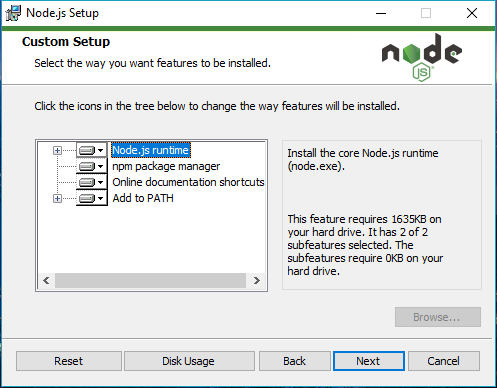
***Set the Destination Folder where you want to install Node.js & Select “Next”***



#### Fig 8 : Node Destination folder

* Custom Setup

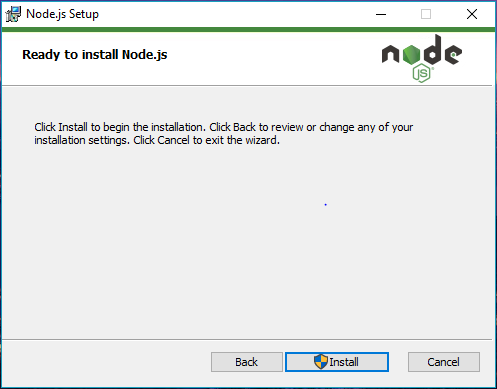
***Select “Next”***



#### Fig 9 : Custom Setup

* Ready to Install Node.js.

***Select “Install”***



#### Fig 10: Node Install

**NOTE :**

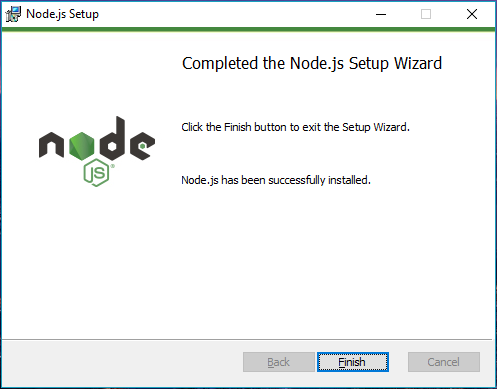
A prompt saying – “This step requires administrative privileges” will appear.

Authenticate the prompt as an **“Administrator”**

* Installing Node.js.

Do not close or cancel the installer until the install is complete

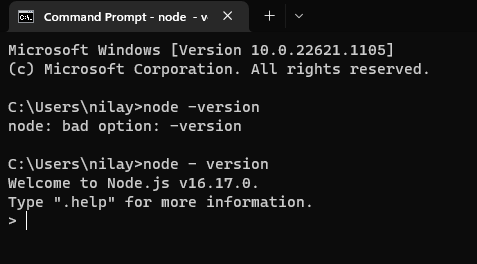
* Complete the Node.js Setup Wizard.  
  ***Click “Finish”***



#### Fig 11: Node Installation Finished

**Step 3: Verify that Node.js was properly installed or not.**

To check that node.js was completely installed on your system or not, you can run the following command in your command prompt or Windows PowerShell and test it:-



#### Fig 12: Installation Verification

1. Installing expo CLI.

Now we need to install a tool called expo CLI.

Expo CLI

Expo CLI is a command-line tool that is the primary interface between a developer and other Expo tools. You are going to use it for different tasks in the development life cycle of your project such as serving the project in development, viewing logs, opening the app on an emulator or a physical device, and so on.

For installing we can use :

npm i expo-cli

This will install expo CLI.

1. **Create the Application:**

Now you can create an application. The command:

For Admin:

- npx create-expo-app admin

For Client:

-npx create-expo-app client

## 5.2 SYSTEM TESTING

**Testing Process**

Software testing is a specialized discipline requiring unique skills. Software testing should not be intuitive as far as possible and we must learn how to do it systematically.

**What is Testing?**

**“*Testing is the process of executing a program with the intent of finding errors.*”**

The whole effort of software engineering activities is to design methods and tools to eliminate errors at the source. Software testing is becoming increasingly important in the earlier part of the software life cycle, aiming to discover errors before they are deeply embedded within systems. I

In software testing we are facing a major dilemma. On one hand, we wish to design the software product that has zero errors, while on the other hand we must remain firm in our belief that any software product under testing certainly has errors, which needs to be unearthed.

**Why should we test?**

Although software testing is itself an expensive activity, yet launching of software without testing may lead to cost potentiality much higher than that of testing, especially in systems in which human safety is involved. No one would think of allowing automatic pilot software into service without the most rigorous testing. In so-called life’s critical systems, economics must not be the prime consideration while deciding whether a product should be released to the customer.

In most systems, however, it is the cost factor, which plays a major role. It is both the driving force and the limiting factor as well. In the software life cycle the earlier the errors are discovered and removed, the lower is the cost of their removal. The most damaging errors are those, which are not discovered during the testing process and therefore remain when the system ‘goes live’. In commercial systems it is often difficult to estimate the cost of errors. For example, in a banking system, the potential cost of even minor software error could be enormous the consequential cost of lot business (which may never be recovered) can be beyond calculations.

It is not possible to test the software for all possible combinations of input cases. No software would ever be released by its creators if they were asked to certify that it was totally free of all errors. Testing therefore continues to the point where it is considered that the costs of the testing processes significantly outweigh the returns. Hence, when to release the software in the market, is a very important decision.

**What should we test?**

We should test the program’s responses to every possible input. It means we should test for all valid and invalid inputs. Complete testing is not just possible, although, we may wish to do so.

Another dimension is to execute all possible paths of the program. A program path can be traced through the code from the start of the program to program termination. Two paths differ if the program executes different statements in each, or execute the same statements but in different order.

The point which we would to like to highlight is that complete or exhaustive testing is just not possible. Exhaustive testing requires every statement in the program and every possible path combination to be executed at least once. So our objective is not possible to be achieved and we may have to settle for something less than that of complete testing.

We may like to test those areas where probability of getting a fault is maximum. Such critical and sensitive areas are not easy to identify. Organizations should develop strategies and policies for choosing effective testing techniques rather than leaving this to arbitrary judgments of the development team.

**Alpha, Beta and Acceptance Testing**

It is not possible to predict every usage of the software by the customer. Customers may try with strange inputs, combination of inputs and so many other things. Some outputs may be very clearing from the developers’ perspective, but customers may not understand and finally may not appreciate it. In order to avoid or minimize such situations, customer involvement is required before delivering the final product. The above mentioned three terms are related to customer’s involvement in testing but have different meanings.

***Acceptance Testing***

The term is used when the software is developed for a specific customer. A series of test are conducted to enable the customer to validate all requirements. These tests are conducted by the end user / customer and may range from adhoc tests to well-planned systematic series of tests. Acceptance testing may be conducted for few weeks or months. The discovered errors will be fixed and better-quality software will be delivered to the customer.

***Alpha and Beta Testing***

The terms Alpha and Beta testing are used when the software is developed as a product for anonymous customers. Hence, formal Acceptance testing is not possible in such cases. However, some potential customers are identified to get their view about the product. The Alpha tests are conducted at the developer’s site by a customer. These tests are conducted in a controlled environment. Alpha testing may be started when formal testing process is near completion.

The Beta tests are conducted by the customers / end users at their sites. Unlike, Alpha testing, developer is not present here. Beta testing is present in a real environment that cannot be controlled by the developer. Customers are expected to report failures, if any, to the company. After receiving such failure reports, developers modify the code and fix the bug and prepare the product for final release.

Most of the companies are following this practice. Firstly, they sent the Beta release of their product for few months. Many potential customers will use the product and may send their views about the product. Hence, the company gets the feedback of many potential customers. The best part is that the reputation of the company is not at stake even if many failure situations are encountered.

**Functional Testing**

As discussed earlier, complete testing is not at all possible. Thus, we may like to reduce this incompleteness as much as possible. Probably the poorest methodology is random input testing. In random input testing, some subset of all input values is selected randomly. In terms of probability of detecting errors, a randomly selected collection of test cases has little chance of being an optimal, or close to optimal subset. What we are looking for is a set of thought processes that allow us to select a set of data more intelligently.

**Boundary Value Analysis**

Experience shows that test cases that are close to boundary conditions have higher chances of detecting an error. Here boundary condition means, an input value may be on the boundary; just below the boundary (upper side) or just above the boundary (lower side).

***Robustness Testing***

It is nothing but the extension of Boundary Value Analysis. Here, we would like to see, what happens when the extreme values are exceeded with a value slightly greater than the maximum and a value slightly less than the minimum. It means, we want to go outside the legitimate boundary of input domain. This type of testing is quite common in electric and electronic circuits. This extended form of boundary value analysis is called *Robustness Testing* and is shown in (Fig 2.0).

**Worst Case Testing**

If we reject “single faults” assumption theory of reliability and may like to see what happens when more than one variable has an extreme value. In electronic circuit analysis, this is called “*Worst Case Analysis*”. It is more thorough in the sense that boundary value test cases are proper subset of worst-case test cases.

**Unit Testing**

Unit testing is the process of taking a module and running it in isolation from the rest of the software product by using prepared test cases and comparing the actual results with the results predicted by the specifications and design of the module. One purpose of testing is to find (and remove) as many errors in the software as practical. There are number of reasons in support of unit testing than testing the entire product.

1. The size of a single module is small enough that we can locate an error fairly easily.
2. The module is small enough that we can attempt to test it in some demonstrably exhaustive fashion.
3. Confusing interactions of multiple errors in widely different parts of the software are eliminated.

There are problems associated with testing a module in isolation. How do we run a module without anything to call it, to be called by it or, possibly to output intermediate values obtained during execution? One approach is to construct an appropriate driver routine to call it and, simple stubs to be called by it, and to insert output statement in it.

Stubs serve to replace module that are subordinate to (called by) the module to be tested. A stub or dummy subprogram uses the subordinate module’s interface, may do minimal data manipulation, prints verification of entity, and returns.

This overhead code, called scaffolding represents effort that is important to testing, but does not appear in the delivered product. If drivers and stubs are kept simple, actual overhead is relatively low. Unfortunately, many modules cannot be adequately unit tested with simple overhead software. In such cases complete testing can be postponed until the integration test step (where drivers and stubs are also used). A second approach is to generate the scaffolding automatically by means of attest harness, among other things, allows to own a single unit in isolation, while simulating the rest of the software system environment by providing the appropriate input, output, parameters and interaction for the unit. A third and rather ineffective technique is to omit unit testing and simply is to allow incremental addition of module to a partially integrated product, hoping that the integration testing will also provide sufficient coverage of the modules structure.

**Integration Testing**

The purpose of unit testing is to determine that each independent module is correctly implemented. This gives little chance to determine that the interface between module is also correct, and for this reason integration testing must be performed. One specific target of integration testing is the interface: whether parameters match on both sides as to type, permissible ranges, meaning and utilization.

There are several classical integration strategies that really have little bases in a rational methodology Top down \_ integration proceeds down the invocation, higher key, adding one module at a time until an entire tree level is integrated; and thus, it eliminates the need for drivers. The bottom\_ up strategy works similarly from the bottom and has no need of stubs. A sandwich strategy runs from top and bottom concurrently, meeting somewhere in the middle.

**System Testing**

Of the three level of testing, the system level is closet to everyday experience. We test many things; a used car before we buy it; an online cable network service before we subscribe, and so on. A common pattern in these familiar forms is that we evaluate a product in terms of our expectations; not with respect to a specification or a standard. Consequently, goal is not to find faults, but to demonstrate performance. Because of this we tend to approach system testing from a functional standpoint rather than from a structural one. Since it is so intuitively familiar, system testing in practice tense to be less formal then it might be, and is compounded by the reduced testing interval that usually remains before a delivery deadline.

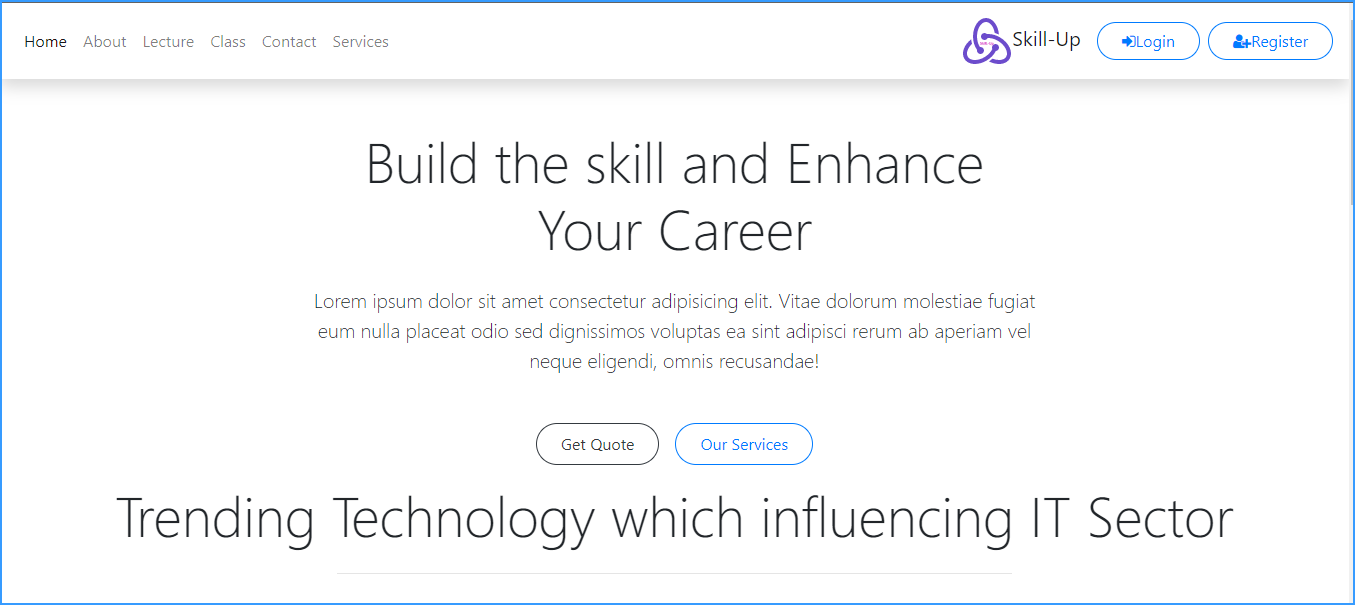
As we know software is one component of a large computer-based system. Ultimately, software is incorporated with other system components (e.g. new hardware, information), and thus, a series of special test are to be conducted. Many times, software products are designed to run on a variety of hardware configuration. The software should actually be tested on many different hardware set-ups, although the full range of memory, processor, operating system, and peripheral possibilities may be too large for complete testing. There are many types of specification, and we should be aware of those as we perform system testing. For instance, there may be a specified level of performance required for the software. This may involve measurement of response time under various loads and operating conditions. It may also require measurements of main and disk memory usage. Software reliability should also be measured during all other test of the integrated products. If minimum and average up time behavior of the product when specified, then this should be made. The time and effort needed to recover from failure should be recorded and compared with specifications. These specifications should represent customer’s wants and needs, and during system testing, we can try to see if the requirements and specification really do coincide.

# 

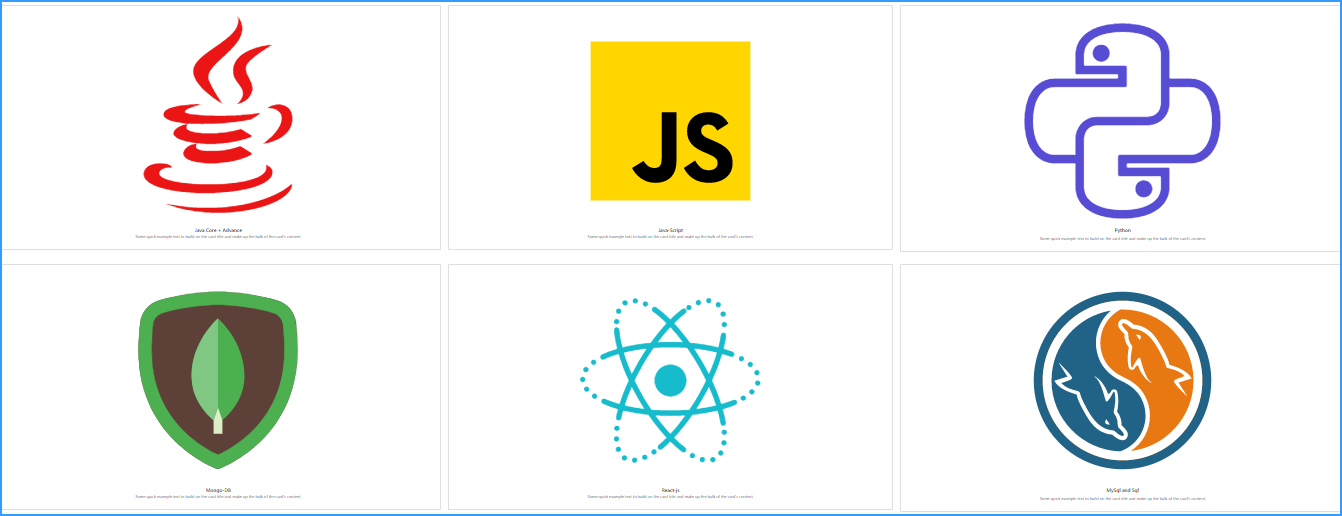
# Chapter 6

# SNAPSHOT OF THE RUNNING MODEL

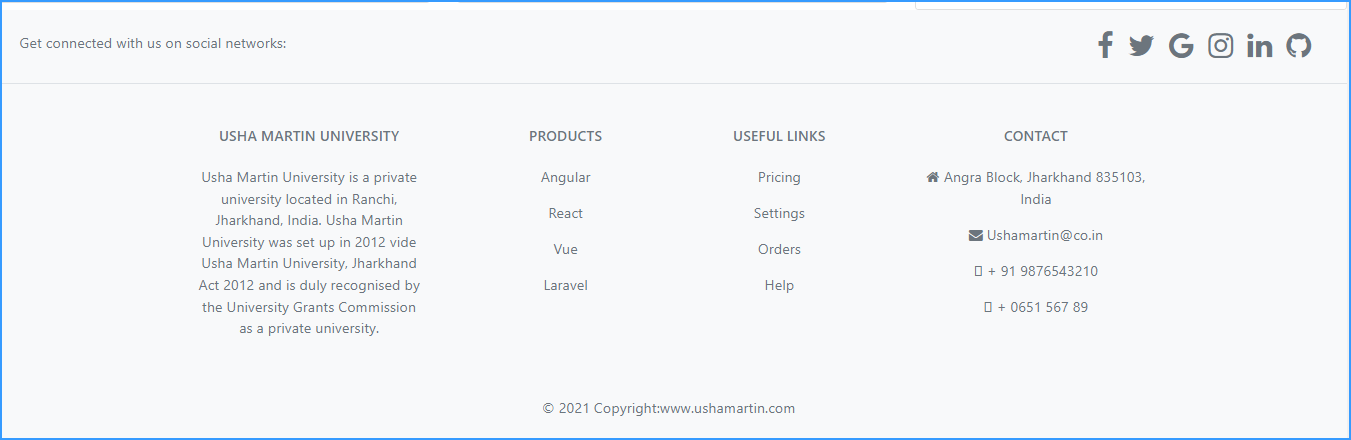
#### Fig 13: Home Page



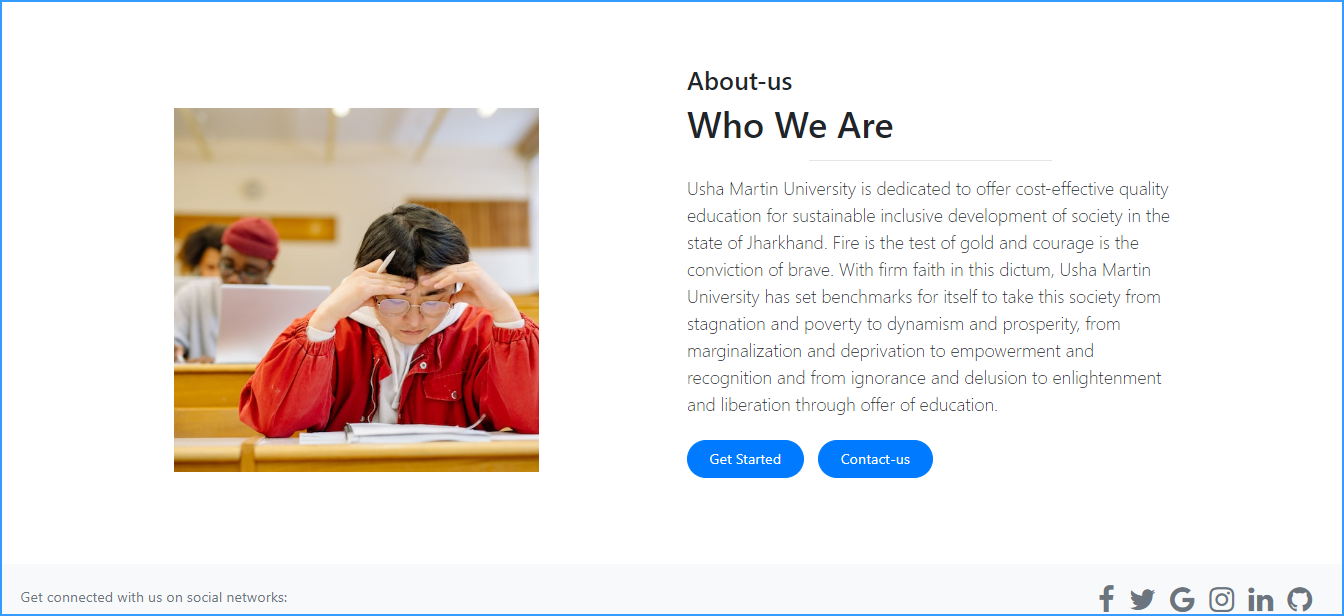
#### Fig 14: Technologies Page



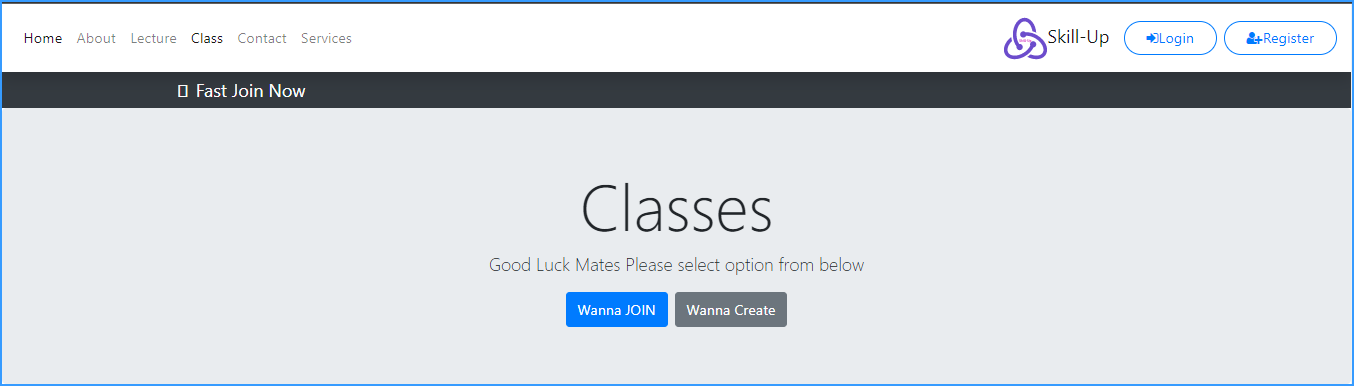
#### Fig 15: Footer page



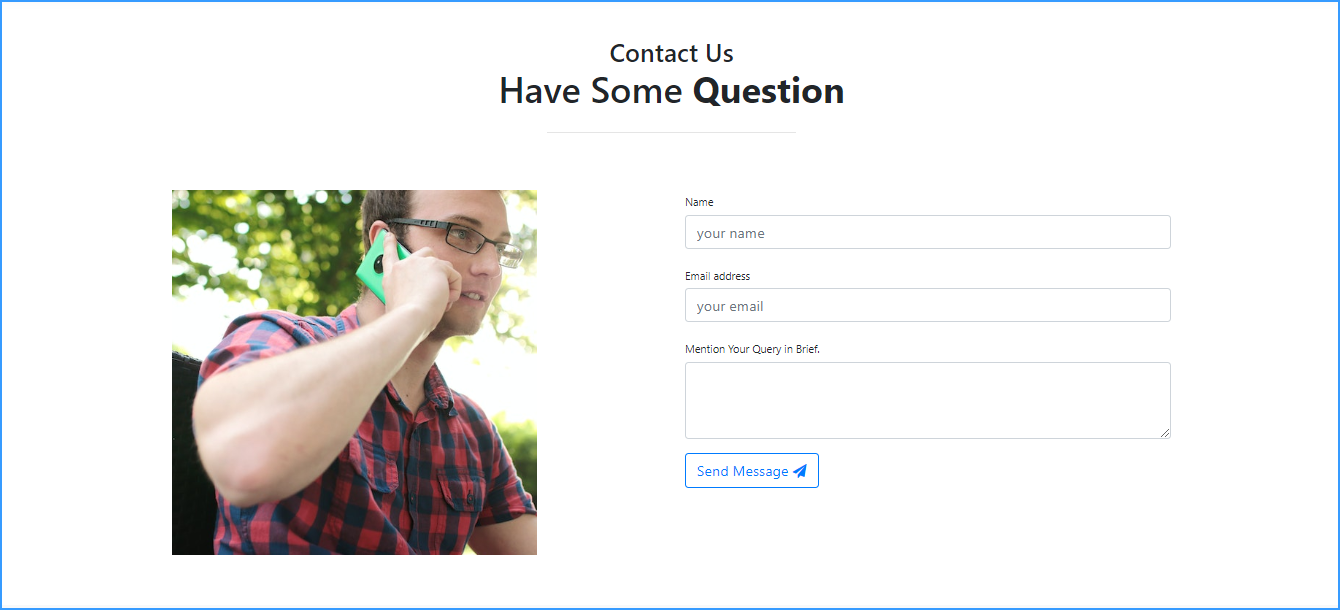
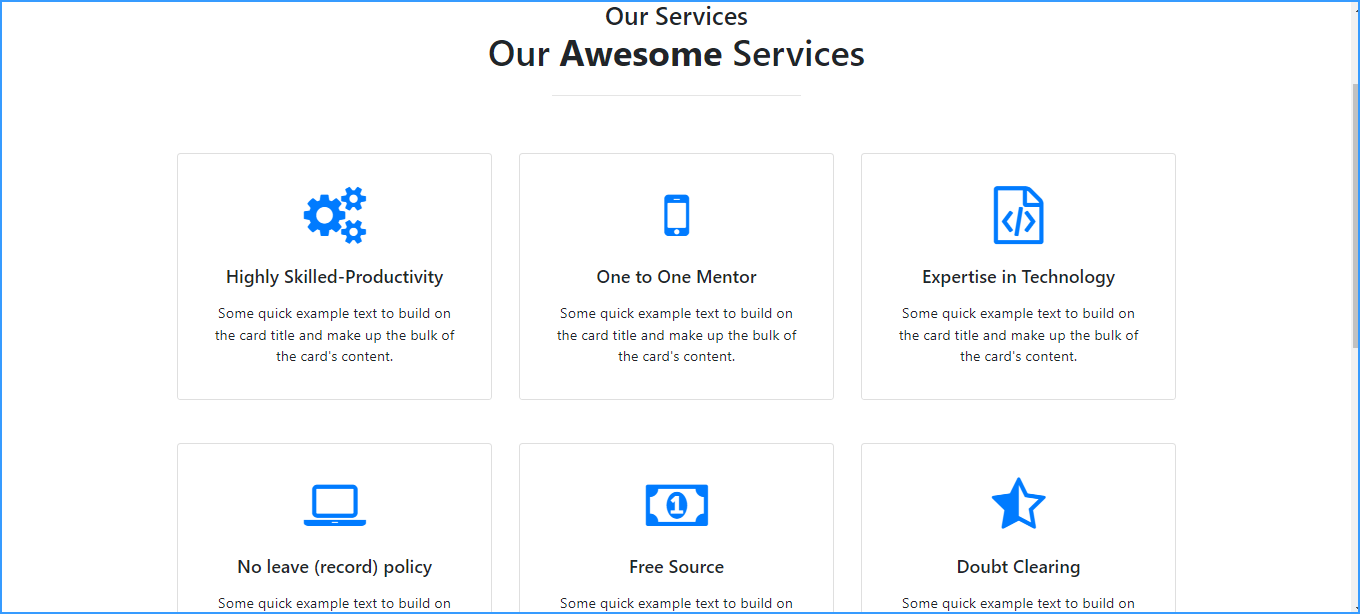
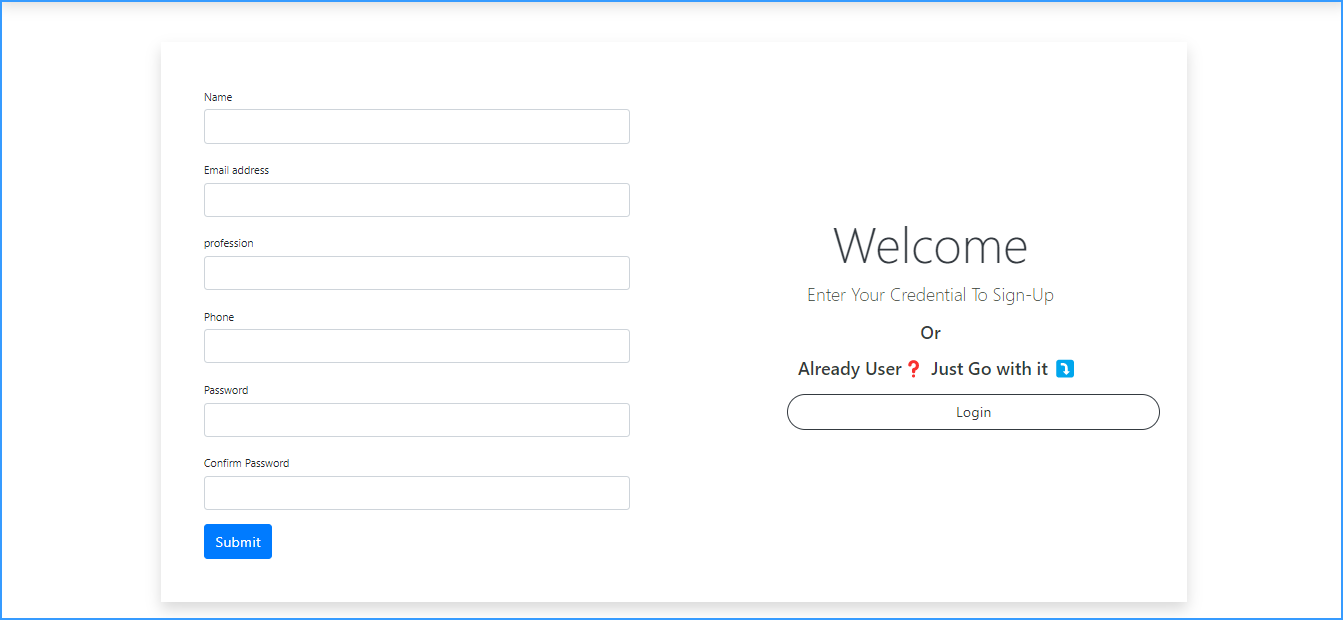
#### Fig 16: About Page

****

#### Fig 17: Class Joining Page

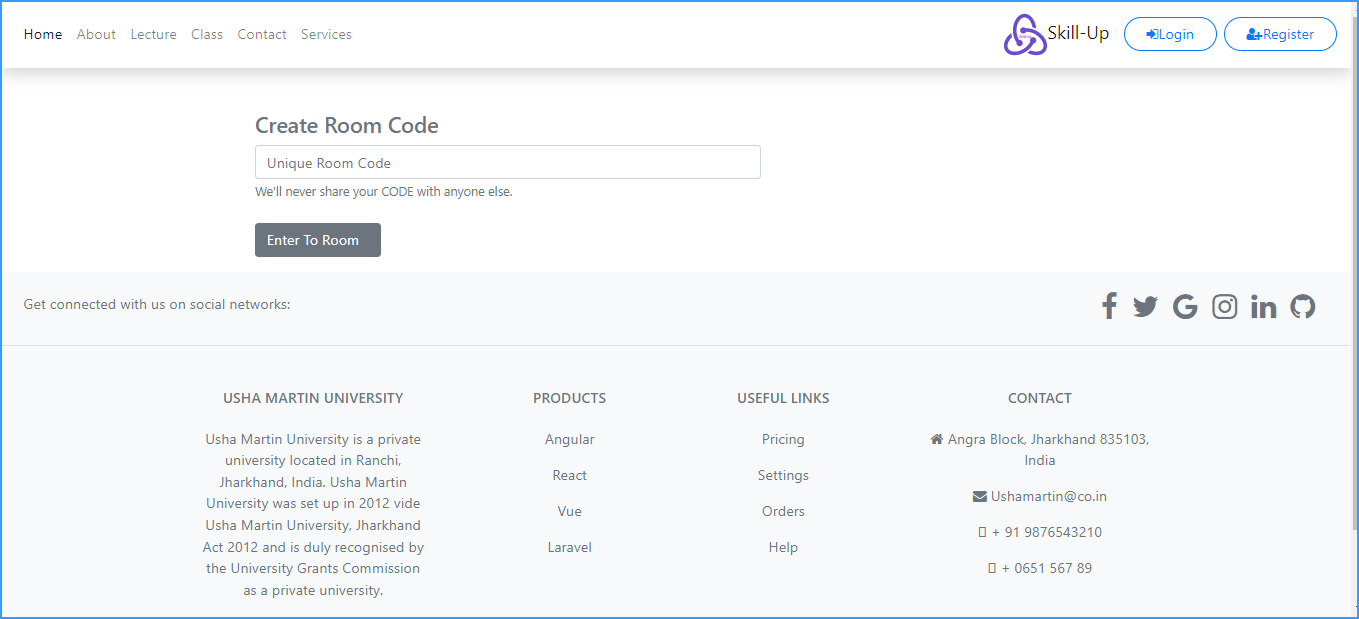


#### Fig 18: Contact Us Page

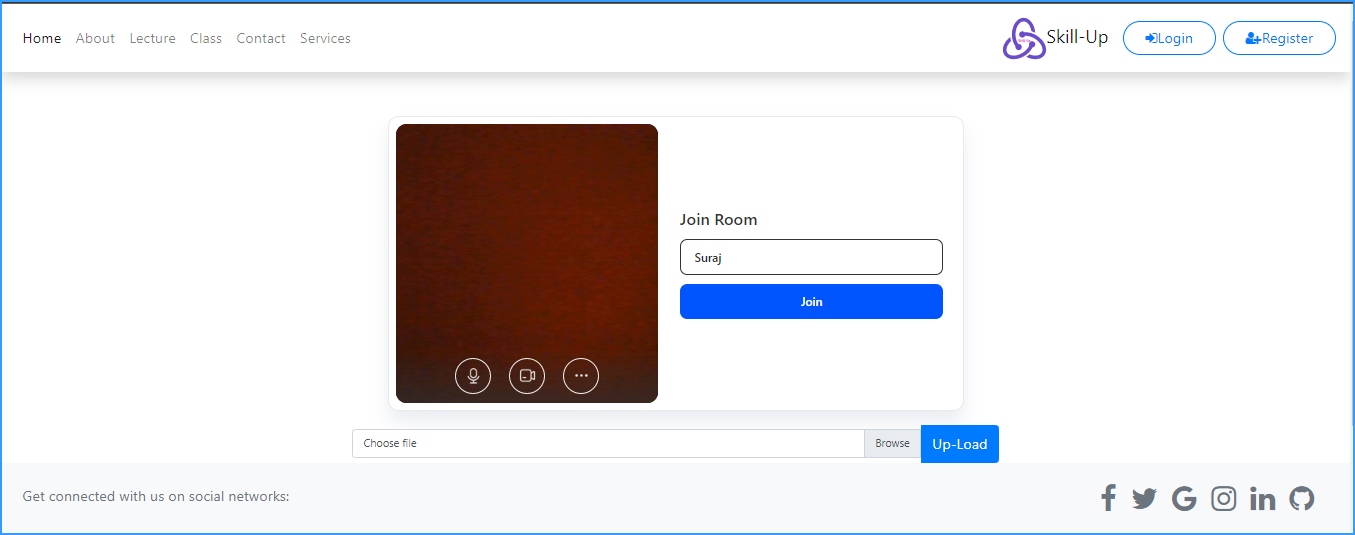
Fig 19: Services Page Fig 19: Register Page 

#### Fig 21: Login Page

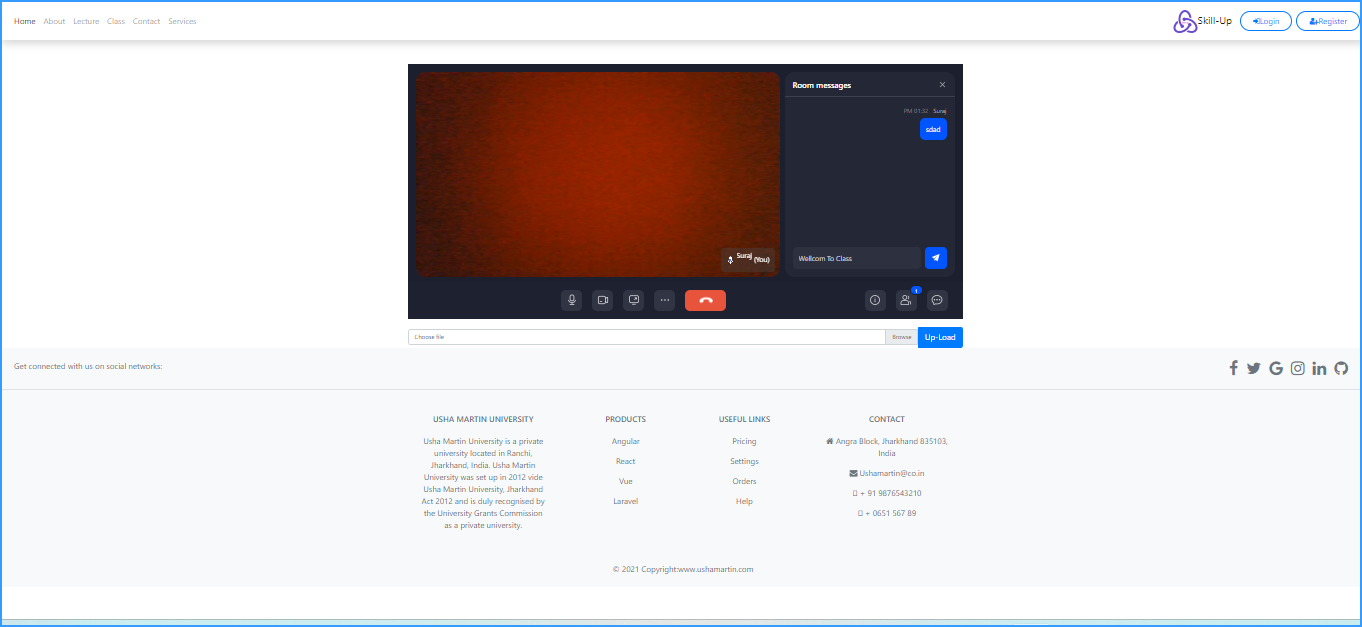
#### Fig 21: Create Room Page



#### Fig 21:Room Joining Page



#### Fig 21: Class Loby



# Chapter 7

# LIMITATIONS, CONCLUSIONS AND FUTURE SCOPE

## 7.1 LIMITATIONS

Nowadays, Mobile application is not much secure. Each and every developed app is having some or the other mistakes followed by third party tool integration. These loopholes in development not only affect smartphone devices but, also damage user's personal data and information on a heavy scale. Defects here can be anything, whether it is relevant to hardware or the software.

Today, mobile apps have become a part of the current culture at fast speed.

Furthermore, the secure mobile app development has not been obvious with the same level of ability. Of course, secure development guidelines exist in the community. Here, we will sum up some drawbacks, as these are the most obvious ones found in mobile apps.

**1. Insufficient Transport Layer Protection:** We should need the TLS/SSL encryption with solid algorithms within communications. The unique blunder is unencrypted connections from the app to a third-party app development company. You must program your apps to showcase any warn messages so that the user is intercommunicated of the configuration of the encrypted connection.

**2. Poor Authorization & Authentication:** These vulnerabilities are established mostly on the server side. The best exercises thatyou should follow are the same with web apps. Especially for app development, device

identifiers ought to be ignored since devices can be removed and interfered with. Finally, out-of-band authentication tokens should not be sent to the related device.

**3. Client-Side Injection:** This category has consisted of a broad diverseness of input strikes against theapplication itself. General best practices for reduction of client-side injection drawbackscover the input validation of the app entry points, on the server side. To avoid this, you

should use parameterized queries, disable file system access for Web views, JavaScript and plugin support for Web views.

**4. Wrong Session Handling:** While session handling mechanisms are largely applied at the server side of apps, securesession management practices can be used in devices themselves. The Confidentiality &

Integrity of session tokens should be preserved via TLS/SSL connections. Like authorization & authentication, device identifiers should be avoided here as well and you should execute safe mechanisms to cancel session on lost devices.

**5. Security Resolutions Through Untrusted Inputs:**

While these issues primarily affect Android-based apps, there has been a case in point for iPhone apps too. Usually and especially, output escaping, authorization controls, input validation, and canonicalization should be carefully analyzed. Also, you should take extra care when to accept and validate URL schemes.

**6. Side Channel Data Leakage:** This comprises of data exchange that normally maximizes the app performance. As withWeak Data Storage, you should develop your app under the premise that the devicemight be taken. The application should be dynamically examined in order to prove that

it does not leak the data while runtime.

**7. Malware:** There are lots of malware that people forced to install from the play store due to hiddenidentity wrapped inside the application. While there is a Google Bouncer open at theplay store to auto detect and block any type of Malware but yet it fails to do. So, toassure a good app, developers must check if the app contains any malware or not. Also,

there are lots of free as well as paid Anti-Malware apps available to preserve you from the hateful application. The app market is constantly growing, we expect to see a step-up in a number of attacks against mobile devices themselves. So, you should develop your next application with

app security in mind.

## 7.2 FUTURE SCOPE

Purchasing and selling products and services over the internet without the need of going physically to the market is what online shopping all about. Online shopping is just like a retail store shopping that we do by going to the market, but it is done through the internet. Online shopping has made shopping painless and added more fun. Online stores offer product description, pictures, comparisons, price and much more. Few examples of these are Amazon.com, ebay.com, framt.com and the benefits of online shopping is that by having direct access to consumer ,the online stores can offer products that cater to the needs of consumer ,cookies can be used for tracking the customer selection over the internet or what is of their interest when they visit the site again . Online shopping makes use of digital technology for managing the flow of information, products, and payment between consumer, site owners and suppliers. Online shopping can be either B2B (business to business) or B2C (business to consumer)  
  
  
Shopping cart is one of the important facilities provided in online shopping, this lets customer to browse different goods and services and once they select an item to purchase they can place the item in shopping cart, and continue browsing till the final selection. Customers can even remove the items from shopping cart that were selected earlier before they place the final order. It reminds us of shopping basket that we carry in departmental store.

## 7.3 CONCLUSION

People need to purchase items such as clothing, and now have the choice to shop online or make purchases in the traditional manner. I. Online shopping has increased and changed the way people shop. 1) There is convince having the ability to shop from home.  
  
\*With the introduction of internet shopping online has become popular. Shopping online allows access to merchandise sold worldwide. It is a growing part of retail. Online shopping is time saving and convenient. There is often no cost for traveling when ordering items online. 2) People do not have to interact.  
  
\*Shopping online an individual does not have face to face transactions. There is limited person to person interaction with online shopping. People do not have to deal with crowds or unwanted socializing. Ordering items online can occur around the clock without the needing another person’s assistance. A person can make returns and purchases with the click of a computer mouse. 3) Security of personal information and merchandise.  
  
\*Reliable shopping websites offers a secure way to make transactions. PayPal is a company (www.paypal.com) that offers a reliable and secure way to make shopping online purchases. Using only secure sites will help secure personal information. Tracking numbers to track packages containing purchased items are often provided or can be requested. When making online purchase a confirmation can be sent to a personal email account if one has been provided. II. Traditional shopping offers things that internet shopping does not. 1) There is the ability to interact face to face with others. \*When people go outside of their homes to shop they are able to socialize and make to acquaintances. With face to face shopping you can make sure that you are charged correctly. If someone is not charged correctly or mistakes have been made, a receipt is often given at the point of sell. It is easy to haggle for deals when shopping in person. Traditional shopping offers families and friends the chance to spend time together. 2) People can physically and visually check the quality of the merchandise. \*With traditional shopping items to be purchased can be examined in advance of purchase. If the quality of an item is not what it should be it can be noticed quickly and easily when shopping traditionally. Some individuals prefer to look before they buy items to ensure they receive what they want. Shopping traditionally allows people the opportunity to do so. 3) It cost money to travel for shopping to make purchases.  
\* Traveling of some sort will happen when shopping the traditional way. The expense of gas or bus tickets should be considered. Factoring in the cost of travel with the amount intended for shopping can help those on a budget. When shopping traditionally quality products are often found various areas. Travel expense will occur. III. Peoples shopping experience varies depending on how they shop via online or traditionally. 1) Some people enjoy socializing when shopping while others do not. \* People shop traditionally to spend time with family and friends. These people often enjoy socializing and are not concerned with crowds. Other people enjoy shopping online because of how convenient it is. People have different needs and desires. 2) Make sure personal information is secure and purchases are correct. \*When shopping making sure personal information such as someone’s social security number is secure is important. Online shoppers and traditional shoppers should receive some proof of purchase once an item is purchased. Reviewing the receipt of any purchased item will aid with corrections if needed. Keeping record and receipts will also help if an item purchased needs to be returned or is defective. 3) The quality of merchandise, who, and how items are purchased should be considered when shopping. \*Shopping online or traditionally the seller’s reputation and credibility should be a consideration. Traditional shopping and online shopping offer the choice of who someone does business with. Quality and hard to find items are found easily on the internet and when shopping the traditional way. III. Conclusion  
4) With the introduction of internet shopping individual can choose to make purchases online or in person. Online shopping has a large consumer market, but people also enjoy face to face transactions when shopping. The quality of merchandise, who, and how items are purchased are things considered when shopping. How someone chooses to shop depends on personal preference. The choices are yours.

# 

# Chapter 8

# SOURCE CODE

Server Part

````````````````````````

Dependencies of Server

{

"name": "MERN\_project",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "nodemon app.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"bcryptjs": "^2.4.3",

"cookie-parser": "^1.4.5",

"dotenv": "^16.0.3",

"express": "^4.18.2",

"jsonwebtoken": "^9.0.0",

"mongoose": "^6.8.1"

},

"watch": [

"src/"

],

"ignore": [

"dist/",

"node\_modules",

".git"

],

"ext": "js,json",

"env": {

"NODE\_ENV": "development"

},

"execMap": {

"js": "node"

},

"devDependencies": {

"nodemon": "^2.0.20"

}

}

***Main Server***

const express = require('express')

const bodyParser = require('body-parser')

const mongoose = require('mongoose')

const cors = require('cors')

const multer =require('multer')

// const {Server} = require('socket.io')

// const articleRouter = require('./Controler/fileControler')

const userController = require('./Controler/userControler')

const fileControler = require('./Controler/fileControler')

const port = 8000

mongoose.set("strictQuery", false);

mongoose.connect('mongodb://127.0.0.1:27017/myapp', async (err)=>{

if(!err){

await console.log("DATABASE CONNECTED SUCCESSFULLY...");

}else{

console.log("OOPS SOMETHING WENT WRONG ==> ",err)

}

});

//io for video server

// const io = new Server({cors:true})

const app = express()

app.use(cors())

//parse application /x-www-font-urlencoded

app.use(bodyParser.urlencoded({extended:false}))

//parse application json

app.use(bodyParser.json())

// Routing part

app.post('/register', userController.signup)

app.post('/signin', userController.signin)

app.get('/vroom/allpost', fileControler.allPost)

app.post('/vroom/addpost', fileControler.addPost)

app.get('/vroom/find/:id', fileControler.findArticleId)

app.put('/vroom/update/:id',fileControler.findArticleIdAndUpdate)

app.delete('/vroom/delete/:id',fileControler.findArticleIdAndDelete)

// app.use('/vroom', articleRouter)

// /////////////////////////////////////////////////////////////////////////////

// const emailToSocketMapping = new Map()

// const socketTOEmailMapping = new Map()

// io.on('connection', (socket)=>{

// console.log("New Connection Joined");

// socket.on('join-room', (data) =>{

// const {roomId, emailId} = data;

// console.log("User", emailId, "Joined Room", roomId)

// emailToSocketMapping.set(emailId, socket.id)

// socketTOEmailMapping.set(socket.id,emailId)

// socket.join(roomId)

// socket.emit('Joined-Room', {roomId})

// socket.broadcast.to(roomId).emit('user-joined', {emailId})

// })

// socket.on("call-user", (data)=>{

// const{emailId, offer} = data;

// const fromEmail = socketTOEmailMapping.get(socket.id)

// const socketId = emailToSocketMapping.get(emailId)

// socket.to(socketId).emit("Incomming call",{from: fromEmail, offer })

// })

// socket.on('call-accepted', (data)=>{

// const {emailId, ans} = data

// const socketId = emailToSocketMapping.get(emailId)

// socket.to(socketId).emit('call-accepted',{ans})

// })

// })

// ////////////////////////////////////////////////////////////////////////////////

app.listen(port, () => {console.log(`Server App Running... on P O R T = ${port}`)})

// io.listen(8001, ()=>{console.log("Video Server Strated in P O R T = 8001");})

Model of Project

const mongoose = require('mongoose')

const Schema = mongoose.Schema

const articleSchema = new Schema({

title: {

type: String,

required: true

},

description: {

type: String,

required: true

},

img: {

type: String,

required: true

},

author: {

type: String,

required: true

},

postDate: { type: Date, default: Date.now }

})

const Articles = mongoose.model('Article', articleSchema)

module.exports = Articles;

***user model***

const mongoose = require('mongoose')

module.exports = mongoose.model('User', {

name: {

type: String,

required: true

},

email: {

type: String,

required: true

},

profession: {

type: String,

required: true

},

phone: {

type: String,

required: true

},

password: {

type: String,

required: true

},

cpassword: {

type: String,

required: true

}

})

***Controller Part***

const Articles = require('../Model/article')

//REQUEST Get all Post

module.exports.allPost = (req, res)=>{

Articles.find().then(article => res.json(article))

.catch(err => res.status(400).json(`Error found => ${err} `))

console.log(req.body)

}

// REQUEST ADD NEW ARTICLE

module.exports.addPost = (req,res)=>{

const newArticle = new Articles({

title:req.body.title,

description:req.body.description,

img:req.body.img,

author:req.body.author

})

newArticle.save()

.then(()=>res.json("The new Article Added SUCCESS"))

.catch(err=>res.status(400).json(`ERROR:=> ${err}`))

}

// REQUEST FIND ARTICLE BY ID

module.exports.findArticleId = (res,req)=>{

Articles.findById(req.params.id)

.then(article=>res.json(article))

.catch(err=>res.status(400).json(`ERROR:=> ${err}`))

}

// FIND ARTICLE BY ID AND UPDATE

module.exports.findArticleIdAndUpdate=(req,res)=>{

Articles.findById(req.params.id)

.then(article=>{

article.title = req.body.title

article.description = req.body.description

article.img = req.body.img

article.author=req.body.author

article.save()

.then(()=>res.json('The Article is Updated Successfully'))

.catch(err=>res.status(400).json(`ERROR:=> ${err}`))

})

.catch(err=>res.status(400).json(`ERROR:=> ${err}`))

}

// REQUEST FIND ARTICLE BY ID AND DELETE

module.exports.findArticleIdAndDelete=(req,res)=>{

Articles.findByIdAndDelete(req.params.id)

.then(()=>res.json("The article is Deleted"))

.catch(err=>res.status(400).json(`ERROR:=> ${err}`))

}

***User Controller***

const userModal = require('../Model/userModel')

module.exports.signup = (req, res) => {

// creating instance of userModal

const newUser = new userModal({

name: req.body.name,

email: req.body.email,

profession: req.body.profession,

phone: req.body.phone,

password: req.body.password,

cpassword: req.body.cpassword,

})

newUser.save().then(() => {

res.send({ code: 200, message: "Sign-Up SUCCESS..." })

}).catch((err) => {

res.send({ code: 500, message: "Something went Wrong on server" })

console.log(err)

})

console.log(req.body);

}

module.exports.signin = (req, res) => {

// matching email.and password

userModal.findOne({ email: req.body.email }).then(result => {

console.log(result);

// matching password

if (result.password !== req.body.password) {

res.send({ code: 404, message: "opps!! Pass Not matched" })

} else {

res.send({ code: 200, message: "user Found", token: "qwertyiop" })

}

}).catch(err => { res.send({ code: 500, message: "User NOT found....." }) })

}

***Client Side***

* ***Classroom-part***

import React, { useState, useEffect, useCallback } from 'react'

import { useNavigate } from 'react-router-dom'

import { useSocket } from '../Providers/Socket'

const Classhome = () => {

const Socket = useSocket();

const navigate = useNavigate();

const [email, setEmail] = useState("")

const [roomId, setRoomId] = useState("")

const handleRoomJoined = useCallback(({ roomId }) => {

console.log("Room joined => ", roomId);

navigate(`/classroom/${roomId}`);

}, [navigate]);

useEffect(() => {

Socket.on("Joined-Room", handleRoomJoined);

return () => {

Socket.off("Joined-Room", handleRoomJoined)

}

}, [handleRoomJoined, Socket]);

const handleJoinRoom = () => {

Socket.emit('join-room', { emailId: email, roomId });

navigate(`/classroom/${roomId}`);

};

return (

<>

<div className="homepage-container">

<div className='input-container'>

<input value={email} onChange={e => setEmail(e.target.value)} type="email" placeholder='Enter your email here' />

<input value={roomId} onChange={e => setRoomId(e.target.value)} type="text" placeholder='Enter Room Code' />

<button onClick={handleJoinRoom}>Enter Room</button>

</div>

</div>

{/\* //////////////////////////////////////////////////////////////////////// \*/}

{/\* <form className="form mt-5 w-75 mt-5">

<div className="form-group w-50 mx-auto ">

<h4 className="text-muted">Create Room Code</h4>

<input type="email" className="form-control mb-2" value={email} onChange={e => setEmail(e.target.value)} required placeholder="Enter Email" />

<input type="text" className="form-control" value={roomId} onChange={e => setRoomId(e.target.value)} required placeholder="Unique Room Code" />

<small id="emailHelp" className="form-text text-muted">We'll never share your CODE with anyone else.</small><br/>

<button type="button" onClick={handleJoinedRoom} className=" btn btn-secondary text-center d-flex w-25 btn-block">Enter To Room</button>

</div>

<div className="">

</div>

</form> \*/}

{/\* ///////////////////////////////////////////////////////////////////////////////////// \*/}

</>

)

}

export default Classhome

………………………………………………………………………………………….

import React, { useEffect, useCallback, useState } from 'react'

import ReactPlayer from 'react-player'

import { useSocket } from '../Providers/Socket'

import { usePeer } from '../Providers/Peer';

const Classroom = () => {

const socket = useSocket();

const { peer, createOffer, createAnswer, setRemoteAns, sendStream, remoteStream } = usePeer();

const [myStream, setMyStream] = useState(null);

const [remoteEmailId, setRemoteEmailId] = useState(null)

const handleNewUserJoined = useCallback(async (data) => {

const { emailId } = data

console.log("New user joined", emailId)

const offer = await createOffer();

socket.emit('call-user', { emailId, offer })

setRemoteEmailId(emailId)

},

[createOffer, socket]

);

const handleIncomingCall = useCallback(async (data) => {

const { from, offer } = data

console.log("Incoming Call From ", from, offer);

const ans = await createAnswer(offer)

socket.emit('call-accepted', { emailId: from, ans })

setRemoteEmailId(from)

}, [createAnswer, socket])

const handleCallAccepted = useCallback(async (data) => {

const { ans } = data

console.log('Call Got Accepted', ans)

await setRemoteAns(ans)

}, [setRemoteAns])

const getUserMediaStream = useCallback(async () => {

const stream = await navigator.mediaDevices.getUserMedia({ audio: true, video: true });

setMyStream(stream)

}, [])

const handleNegosiation = useCallback(() => {

const localOffer = peer.localDescription;

socket.emit('call-user', { emailId: remoteEmailId, offer: localOffer })

}, [peer.localDescription, remoteEmailId, socket])

useEffect(() => {

socket.on('user-joined', handleNewUserJoined)

socket.on('incoming-call', handleIncomingCall)

socket.on('call-accepted', handleCallAccepted)

return () => {

socket.off('user-joined', handleIncomingCall)

socket.off('incoming-call', handleIncomingCall)

socket.off('call-accepted', handleCallAccepted)

}

}, [handleCallAccepted, handleIncomingCall, handleNewUserJoined, socket])

useEffect(() => {

peer.addEventListener('negotiationneeded', handleNegosiation)

return () => {

peer.removeEventListener('negotiationneeded', handleNegosiation)

}

}, [])

useEffect(() => {

getUserMediaStream();

}, [getUserMediaStream])

return (

<>

<div className='room-page-container'>

<h1>Room Page</h1>

<h4>You are connected to {remoteEmailId}</h4>

<button onClick={e => sendStream(myStream)} >Send My Video</button>

<ReactPlayer url={myStream} playing muted />

<ReactPlayer url={remoteStream} playing />

</div>

</>

)

}

export default Classroom

***Lobby of Student***

import React, { useState } from "react";

import { useNavigate } from "react-router-dom";

const VHomePage = () => {

const navigate = useNavigate()

const [roomCode, setRoomCode] = useState('')

const handleFormSubmit = (ev) => {

ev.preventDefault()

navigate(`/vroom/${roomCode}`)

}

return (

<>

<form className="form mt-5 w-75 mt-5">

<div className="form-group w-50 mx-auto ">

<h4 className="text-muted">Create Room Code</h4>

<input type="text" className="form-control" value={roomCode} onChange={e => setRoomCode(e.target.value)} required placeholder="Unique Room Code" />

<small id="emailHelp" className="form-text text-muted">We'll never share your CODE with anyone else.</small><br />

<button type="button" onClick={handleFormSubmit} className=" btn btn-secondary text-center d-flex w-25 btn-block">Enter To Room</button>

</div>

<div className="">

</div>

</form>

</>

)

}

export default VHomePage

…………………………………………………………………………………………….

import React from 'react'

import { useParams } from 'react-router-dom'

import { ZegoUIKitPrebuilt } from '@zegocloud/zego-uikit-prebuilt';

const VRoomPage = () => {

const { roomId } = useParams()

const myMeeting = async (element) => {

const appID = 1154075355;

const sereverSecret = '7b7b6395d8d0f1991f0fc97b4944cf0f';

const kitToken = ZegoUIKitPrebuilt.generateKitTokenForTest(

appID,

sereverSecret,

roomId,

Date.now().toString(),

"Suraj"

)

const zp = ZegoUIKitPrebuilt.create(kitToken);

zp.joinRoom({

container: element,

scenario: {

mode: ZegoUIKitPrebuilt.VideoConference

}

})

}

return (

<>

<div className="room-page mx-auto w-75 mt-5">

<div ref={myMeeting} className="container" />

</div>

<div className="container w-50">

<div className="input-group mb-3">

</div>

<div className="input-group">

<div className="custom-file mt-1">

<input type="file" className="custom-file-input" id="inputGroup-sizing-default" aria-describedby="inputGroupFileAddon04" />

<label className="custom-file-label" for="inputGroupFile04">Choose file</label>

</div>

<div className="input-group-append">

<button className="btn btn-primary " type="button" id="inputGroupFileAddon04">Up-Load</button>

</div>

</div>

</div>

</>

)

}

export default VRoomPage

import React from 'react'

import { useEffect } from 'react'

import pic1 from '../assets/about1.jpg'

function About() {

const callAboutPage = async () => {

try {

await fetch('/about')

} catch (error) {

console.log(error)

}

}

useEffect(() => {

callAboutPage();

}, []);

return (

<>

<section id="about">

<div className="container my-5 py-5">

<div className="row">

<div className="col-md-6">

<img src={pic1} alt="img" className='w-75 me-5 mt-5' />

</div>

<div className="col-md-6">

<h3 className="fs-5">About-us</h3>

<h1 className="display-6 mb-2">Who <b /> We <b />Are </h1>

<hr className='w-50' />

<p className="lead mb-4">Lorem ipsum dolor sit amet, consectetur

adipisicing elit. Labore numquam maxime magnam consequatur nesciunt

accusantium, necessitatibus similique amet blanditiis dicta atque nemo debitis,

fugit quas eum fuga praesentium at non. Assumenda reprehenderit iusto nulla!

Lorem ipsum dolor sit amet consectetur adipisicing elit. Lorem ipsum dolor sit,

amet consectetur adipisicing elit. Provident placeat delectus illo quis quas pariatur

voluptatibus, nostrum, accusamus laboriosam unde, quod distinctio ducimus corrupti eum

veniam fuga hic doloribus neque.

</p>

<button className='btn btn-primary rounded-pill px-4 py-2'>Get Started</button>

<button className='btn btn-primary rounded-pill px-4 py-2 ms-2 ml-3'>Contact-us</button>

</div>

</div>

</div>

</section>

</>

)

}

export default About

……………………………………………………………………………………………..

import React from 'react'

import { Link } from 'react-router-dom'

function Class() {

return (

<>

{/\* ///////////////////////////////////////////////////////// \*/}

<nav className="bg-dark navbar-dark">

<div className="container">

<a href="/" className="navbar-brand"><i className="fas fa-tree mr-2"></i>Fast Join Now</a>

</div>

</nav>

<section id="header" className="jumbotron text-center">

<h1 className="display-3">Classes</h1>

<p className="lead">Good Luck Mates Please select option from below</p>

<Link to="/vroom" className="btn btn-primary">Wanna JOIN</Link>

<Link to="/vhome" className="btn btn-secondary ml-2">Wanna Create</Link>

</section>

<section id="gallery">

<div className="container">

<div className="row">

<div className="col-lg-4 mb-4">

<div className="card">

<img src="https://images.unsplash.com/photo-1477862096227-3a1bb3b08330?ixlib=rb-1.2.1&auto=format&fit=crop&w=700&q=60" alt="" className="card-img-top" />

<div className="card-body">

<h5 className="card-title">Window Programing</h5>

<p className="card-text">Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum similique repellat a laborum, rerum voluptates ipsam eos quo tempore iusto dolore modi dolorum in pariatur. Incidunt repellendus praesentium quae!</p>

<a href="/" className="btn btn-outline-success btn-sm">Read More</a>

<a href="/" className="btn btn-outline-danger btn-sm"><i className="far fa-heart"></i></a>

</div>

</div>

</div>

<div className="col-lg-4 mb-4">

<div className="card">

<img src="https://images.unsplash.com/photo-1516214104703-d870798883c5?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=700&q=60" alt="" className="card-img-top" />

<div className="card-body">

<h5 className="card-title">Complier Design</h5>

<p className="card-text">Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum similique repellat a laborum, rerum voluptates ipsam eos quo tempore iusto dolore modi dolorum in pariatur. Incidunt repellendus praesentium quae!</p>

<a href="/" className="btn btn-outline-success btn-sm">Read More</a>

<a href="/" className="btn btn-outline-danger btn-sm"><i className="far fa-heart"></i></a>

</div>

</div>

</div>

<div className="col-lg-4 mb-4">

<div className="card">

<img src="https://images.unsplash.com/photo-1477862096227-3a1bb3b08330?ixlib=rb-1.2.1&auto=format&fit=crop&w=700&q=60" alt="" className="card-img-top" />

<div className="card-body">

<h5 className="card-title">CryptoGraphy</h5>

<p className="card-text">Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ut eum similique repellat a laborum, rerum voluptates ipsam eos quo tempore iusto dolore modi dolorum in pariatur. Incidunt repellendus praesentium quae!</p>

<a href="/" className="btn btn-outline-success btn-sm">Read More</a>

<a href="/" className="btn btn-outline-danger btn-sm"><i className="far fa-heart"></i></a>

</div>

</div>

</div>

</div>

</div>

</section>

</>

)

}

export default Class

……………………………………………………………………………………………..

import React from 'react'

import pic1 from '../assets/contact1.jpg'

function Contact() {

return (

<>

<section id="contact">

<div className="container my-5 py-2">

<div className="row mb-5">

<div className="col-12">

<h3 className='fs-5 text-center mb-0'>Contact Us</h3>

<h1 className='display-6 text-center mb-4'>Have Some <b>Question</b></h1>

<hr className='w-25 mx-auto' />

</div>

</div>

<div className="row">

<div className="col-md-6">

<img src={pic1} alt="imgs" className='w-75' />

</div>

<div className="col-md-6">

<form action="">

<div className="form-group">

<label for="exampleFormControlInput1">Name </label>

<input type="email" className="form-control" id="name" placeholder="your name" />

</div>

<div className="form-group">

<label for="exampleFormControlInput1">Email address</label>

<input type="email" className="form-control" id="email" placeholder="your email" />

</div>

<div className="form-group">

<label for="exampleFormControlTextarea1">Mention Your Query in Brief.</label>

<textarea className="form-control" id="message" rows="3"></textarea>

</div>

<button type="submit" className='btn btn-outline-primary'>Send Message <i className="fa fa-paper-plane ms-2"></i></button>

</form>

</div>

</div>

</div>

</section>

</>

)

}

export default Contact

………………………………………………………………………………………………

import React, { useEffect } from 'react'

import { useNavigate } from 'react-router-dom'

import Card from 'react-bootstrap/Card';

import pic1 from '../assets/cirkit.jpg'

function Dashbord() {

const history = useNavigate()

useEffect(() => {

const token = localStorage.getItem('TOKEN')

if (!token) {

history('/login')

}

}, [])

return (

<>

<hr>

<nav className="nav nav-pills nav-justified">

<link className="nav-item nav-link active rounded-pill" to="/">className or Course</link>

<link className="nav-item nav-link rounded-pill active" to="/">Offline className</link>

<link className="nav-item nav-link rounded-pill active" to="/">Profile</link>

<button className="nav-item nav-link danger active rounded-pill" to="/">Logout</button>

</nav>

{/\* ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// \*/}

<Card classNameName="bg-dark text-bold mt-3">

<Card.Img src={pic1} alt="Card image" classNameName='fluid' style={{ height: '20rem', Padding: "1rem" }} />

<Card.ImgOverlay>

<Card.Title>Card title</Card.Title>

<Card.Text >

This is a wider card with supporting text below as a natural lead-in

to additional content. This content is a little bit longer.

</Card.Text>

<Card.Text>Last updated 3 mins ago</Card.Text>

</Card.ImgOverlay>

</Card>

{/\* ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// \*/}

<hr className="container">

<hr className="main-body">

{/\* <!-- Breadcrumb --> \*/}

<nav aria-label="breadcrumb" className="main-breadcrumb">

<ol className="breadcrumb">

<li className="breadcrumb-item"><link to="index.html">Home</link></li>

<li className="breadcrumb-item"><link to="javascript:void(0)">User</link></li>

<li className="breadcrumb-item active" aria-current="page">User Profile</li>

</ol>

</nav>

{/\* <!-- /Breadcrumb --> \*/}

<hr className="row gutters-sm">

<div className="col-md-4 mb-3">

<div className="card">

<div className="card-body">

<div className="d-flex flex-column align-items-center text-center">

<img src="https://bootdey.com/img/Content/avatar/avatar7.png" alt="Admin" className="rounded-circle" width="150" />

<div className="mt-3">

<h4>John Doe</h4>

<p className="text-secondary mb-1">Full Stack Developer</p>

<p className="text-muted font-size-sm">Bay Area, San Francisco, CA</p>

<button className="btn btn-primary">Follow</button>

<button className="btn btn-outline-primary">Message</button>

</div>

</div>

</div>

</div>

<div className="card mt-3">

<ul className="list-group list-group-flush">

<li className="list-group-item d-flex justify-content-between align-items-center flex-wrap">

<h6 className="mb-0"><svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" className="feather feather-globe mr-2 icon-inline"><circle cx="12" cy="12" r="10"></circle><line x1="2" y1="12" x2="22" y2="12"></line><path d="M12 2a15.3 15.3 0 0 1 4 10 15.3 15.3 0 0 1-4 10 15.3 15.3 0 0 1-4-10 15.3 15.3 0 0 1 4-10z"></path></svg>Website</h6>

<span className="text-secondary">https://bootdey.com</span>

</li>

<li className="list-group-item d-flex justify-content-between align-items-center flex-wrap">

<h6 className="mb-0"><svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" className="feather feather-github mr-2 icon-inline"><path d="M9 19c-5 1.5-5-2.5-7-3m14 6v-3.87a3.37 3.37 0 0 0-.94-2.61c3.14-.35 6.44-1.54 6.44-7A5.44 5.44 0 0 0 20 4.77 5.07 5.07 0 0 0 19.91 1S18.73.65 16 2.48a13.38 13.38 0 0 0-7 0C6.27.65 5.09 1 5.09 1A5.07 5.07 0 0 0 5 4.77a5.44 5.44 0 0 0-1.5 3.78c0 5.42 3.3 6.61 6.44 7A3.37 3.37 0 0 0 9 18.13V22"></path></svg>Github</h6>

<span className="text-secondary">bootdey</span>

</li>

<li className="list-group-item d-flex justify-content-between align-items-center flex-wrap">

<h6 className="mb-0"><svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" className="feather feather-twitter mr-2 icon-inline text-info"><path d="M23 3a10.9 10.9 0 0 1-3.14 1.53 4.48 4.48 0 0 0-7.86 3v1A10.66 10.66 0 0 1 3 4s-4 9 5 13a11.64 11.64 0 0 1-7 2c9 5 20 0 20-11.5a4.5 4.5 0 0 0-.08-.83A7.72 7.72 0 0 0 23 3z"></path></svg>Twitter</h6>

<span className="text-secondary">@bootdey</span>

</li>

<li className="list-group-item d-flex justify-content-between align-items-center flex-wrap">

<h6 className="mb-0"><svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" className="feather feather-instagram mr-2 icon-inline text-danger"><rect x="2" y="2" width="20" height="20" rx="5" ry="5"></rect><path d="M16 11.37A4 4 0 1 1 12.63 8 4 4 0 0 1 16 11.37z"></path><line x1="17.5" y1="6.5" x2="17.51" y2="6.5"></line></svg>Instagram</h6>

<span className="text-secondary">bootdey</span>

</li>

<li className="list-group-item d-flex justify-content-between align-items-center flex-wrap">

<h6 className="mb-0"><svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" className="feather feather-facebook mr-2 icon-inline text-primary"><path d="M18 2h-3a5 5 0 0 0-5 5v3H7v4h3v8h4v-8h3l1-4h-4V7a1 1 0 0 1 1-1h3z"></path></svg>Facebook</h6>

<span className="text-secondary">bootdey</span>

</li>

</ul>

</div>

</div>

<hr className="col-md-8">

<hr className="card mb-3">

<hr className="card-body">

<div className="row">

<div className="col-sm-3">

<h6 className="mb-0">Full Name</h6>

</div>

<div className="col-sm-9 text-secondary">

Kenneth Valdez

</div>

</div>

<hr>

<div className="row">

<div className="col-sm-3">

<h6 className="mb-0">Email</h6>

</div>

<div className="col-sm-9 text-secondary">

fip@jukmuh.al

</div>

</div>

<hr>

<div className="row">

<div className="col-sm-3">

<h6 className="mb-0">Phone</h6>

</div>

<div className="col-sm-9 text-secondary">

(239) 816-9029

</div>

</div>

<hr>

<div className="row">

<div className="col-sm-3">

<h6 className="mb-0">Mobile</h6>

</div>

<div className="col-sm-9 text-secondary">

(320) 380-4539

</div>

</div>

<hr>

<div className="row">

<div className="col-sm-3">

<h6 className="mb-0">Address</h6>

</div>

<div className="col-sm-9 text-secondary">

Bay Area, San Francisco, CA

</div>

</div>

<hr>

<div className="row">

<div className="col-sm-12">

<link className="btn btn-info " target="\_\_blank" to={"https://www.bootdey.com/snippets/view/profile-edit-data-and-skills"}>Edit</link>

</div>

</div>

</hr>

</hr>

<div className="row gutters-sm">

<div className="col-sm-6 mb-3">

<div className="card h-100">

<div className="card-body">

<h6 className="d-flex align-items-center mb-3"><i className="material-icons text-info mr-2">assignment</i>Project Status</h6>

<small>Web Design</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "80%" }} aria-valuenow="80" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>Website Markup</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "72%" }} aria-valuenow="72" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>One Page</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "89%" }} aria-valuenow="89" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>Mobile Template</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "55%" }} aria-valuenow="55" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>Backend API</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "66%" }} aria-valuenow="66" aria-valuemin="0" aria-valuemax="100"></div>

</div>

</div>

</div>

</div>

<div className="col-sm-6 mb-3">

<div className="card h-100">

<div className="card-body">

<h6 className="d-flex align-items-center mb-3"><i className="material-icons text-info mr-2">assignment</i>Project Status</h6>

<small>Web Design</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "80%" }} aria-valuenow="80" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>Website Markup</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "72%" }} aria-valuenow="72" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>One Page</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "89%" }} aria-valuenow="89" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>Mobile Template</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "55%" }} aria-valuenow="55" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<small>Backend API</small>

<div className="progress mb-3" style={{ height: "5px" }}>

<div className="progress-bar bg-primary" role="progressbar" style={{ width: "66%" }} aria-valuenow="66" aria-valuemin="0" aria-valuemax="100"></div>

</div>

</div>

</div>

</div>

</div>

</hr>

</hr>

</hr>

</hr>

</hr>

</hr>

</hr>

</hr>

</hr>

</hr>

</>

)

}

export default Dashbord

………………………………………………………………………………………………

import React from 'react'

function Footer() {

return (

<>

<footer className="text-center text-lg-start bg-light text-muted">

{/\* <!-- Section: Social media --> \*/}

<section className="d-flex justify-content-center justify-content-lg-between p-4 border-bottom">

{/\* <!-- Left --> \*/}

<div className="me-5 d-none d-lg-block">

<span>Get connected with us on social networks:</span>

</div>

{/\* <!-- Left --> \*/}

{/\* <!-- Right --> \*/}

<div>

<a href="/" className="me-4 text-reset ">

<i className="fa fa-facebook fa-2x mr-3"></i>

</a>

<a href="/" className="me-4 text-reset">

<i className="fa fa-twitter fa-2x mr-3"></i>

</a>

<a href="/" className="me-4 text-reset">

<i className="fa fa-google fa-2x mr-3"></i>

</a>

<a href="/" className="me-4 text-reset">

<i className="fa fa-instagram fa-2x mr-3"></i>

</a>

<a href="/" className="me-4 text-reset">

<i className="fa fa-linkedin fa-2x mr-3"></i>

</a>

<a href="/" className="me-4 text-reset">

<i className="fa fa-github fa-2x mr-3"></i>

</a>

</div>

{/\* <!-- Right --> \*/}

</section>

{/\* <!-- Section: Social media --> \*/}

{/\* <!-- Section: Links --> \*/}

<section className="">

<div className="container text-center text-md-start mt-5">

{/\* <!-- Grid row --> \*/}

<div className="row mt-3">

{/\* <!-- Grid column --> \*/}

<div className="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">

{/\* <!-- Content --> \*/}

<h6 className="text-uppercase fw-bold mb-4">

<i className="fas fa-gem me-3"></i>Usha Martin University

</h6>

<p>

Usha Martin University is a private university located in Ranchi,

Jharkhand, India. Usha Martin University was set up in 2012 vide Usha Martin University,

Jharkhand Act 2012 and is duly recognised by the University Grants Commission as a private

university.

</p>

</div>

{/\* <!-- Grid column --> \*/}

{/\* <!-- Grid column --> \*/}

<div className="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">

{/\* <!-- Links --> \*/}

<h6 className="text-uppercase fw-bold mb-4">

Products

</h6>

<p>

<a href="#!" className="text-reset">Angular</a>

</p>

<p>

<a href="#!" className="text-reset">React</a>

</p>

<p>

<a href="#!" className="text-reset">Vue</a>

</p>

<p>

<a href="#!" className="text-reset">Laravel</a>

</p>

</div>

{/\* <!-- Grid column --> \*/}

{/\* <!-- Grid column --> \*/}

<div className="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">

{/\* <!-- Links --> \*/}

<h6 className="text-uppercase fw-bold mb-4">

Useful links

</h6>

<p>

<a href="#!" className="text-reset">Pricing</a>

</p>

<p>

<a href="#!" className="text-reset">Settings</a>

</p>

<p>

<a href="#!" className="text-reset">Orders</a>

</p>

<p>

<a href="#!" className="text-reset">Help</a>

</p>

</div>

{/\* <!-- Grid column --> \*/}

{/\* <!-- Grid column --> \*/}

<div className="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">

{/\* <!-- Links --> \*/}

<h6 className="text-uppercase fw-bold mb-4">Contact</h6>

<p><i className="fa fa-home me-3"></i> Angra Block, Jharkhand 835103, India</p>

<p>

<i className="fa fa-envelope me-3 mr-1"></i>

Ushamartin@co.in

</p>

<p><i className="fas fa-phone me-3"></i> + 91 9876543210</p>

<p><i className="fas fa-print me-3"></i> + 0651 567 89</p>

</div>

{/\* <!-- Grid column --> \*/}

</div>

{/\* <!-- Grid row --> \*/}

</div>

</section>

{/\* <!-- Section: Links --> \*/}

{/\* <!-- Copyright --> \*/}

<div className="text-center p-4">

© 2021 Copyright:

<a className="text-reset fw-bold mr-2" href="/">www.ushamartin.com</a>

</div>

{/\* <!-- Copyright --> \*/}

</footer>

{/\* <!-- Footer --> \*/}

</>

)

}

export default Footer

………………………………………………………………………………………………

import React from 'react'

import { NavLink } from 'react-router-dom';

import './App.css';

import pic1 from '../assets/icons8-java.svg'

import pic2 from '../assets/icons8-javascript.svg'

import pic3 from '../assets/icons8-python.svg'

import pic4 from '../assets/icons8-mongodb.svg'

import pic5 from '../assets/icons8-react.svg'

import pic6 from '../assets/icons8-my-sql.svg'

function Home() {

return (

<>

<section id='home'>

<div className="container">

<div className="row justify-content-center">

<div className="col-md-8 mt-5">

<h1 className="display-4 fw-border mb-4 text-center">Build the skill and Enhance Your Career</h1>

<p className="lead text-center fs-4 mb-5">

Lorem ipsum dolor sit amet consectetur adipisicing elit. Vitae dolorum molestiae

fugiat eum nulla placeat odio sed dignissimos voluptas ea sint adipisci rerum ab aperiam

vel neque eligendi, omnis recusandae!

</p>

<div className="buttons d-flex justify-content-center">

<NavLink to='/contact' className='btn btn-outline-dark me-4 rounded-pill px-4 py-2'>Get Quote</NavLink>

<NavLink to='/services' className='btn btn-outline-primary ml-3 rounded-pill px-4 py-2'>Our Services</NavLink>

</div>

</div>

</div>

</div>

<h4 className='display-4 fw-border mb-4 text-center mt-3'> Trending Technology which influencing IT Sector</h4>

<hr className='w-50 mx-auto' />

{/\* <div className="container my-5 py-5"> \*/}

<div className="row">

<div className="col-12">

</div>

</div>

<div className="row mt-5">

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<img src={pic1} alt="img" className='w-50' />

<h5 className="card-title mb-1 fs-4 fw-bold">Java Core + Advance </h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<img src={pic2} alt="img" className='w-50' />

<h5 className="card-title mb-1 fs-4 fw-bold"> Java-Script </h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<img src={pic3} alt="img" className='w-50' />

<h5 className="card-title fs-4 fw-bold">Python</h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

</div>

<div className="row mt-5">

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<img src={pic4} alt="img" className='w-50' />

<h5 className="card-title mb-1 fs-4 fw-bold">Mongo-DB </h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<img src={pic5} alt="img" className='w-50' />

<h5 className="card-title mb-1 fs-4 fw-bold">React-js</h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<img src={pic6} alt="img" className='w-50' />

<h5 className="card-title fs-4 fw-bold">MySql and Sql</h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

</div>

{/\* </div> \*/}

</section>

</>

)

}

export default Home

………………………………………………………………………………………………

import React, { useState } from 'react'

import { NavLink, useNavigate } from 'react-router-dom'

import axios from 'axios'

function Login() {

// /////////////////////////////////////////////////////////////////////////////////////

const history = useNavigate()

const [email, setEmail] = useState("")

const [password, setPassword] = useState("")

const loginUser = (e) => {

e.preventDefault();

console.log(email, password,)

axios.post("http://localhost:8000/signin",

{

email: email,

password: password

}

).then(res => {

console.log(res.data)

if (res.data.code === 500) {

alert("User Not Found redirecting to Sign-up")

history('/register')

}

if (res.data.code === 404) {

alert("Password is Wrong")

}

if (res.data.code === 200) {

alert("User Login Successful...")

history('/dashbord')

localStorage.setItem('TOKEN', res.data.token)

}

}).catch(err => { console.log(err) })

}

// ///////////////////////////////////////////////////////////////////////////////////////

return (

<>

<div className="container shadow my-5">

<div className="row">

<div className="ml-5 col-md-5 d-flex flex-column text-dark align-item-center justify-content-center form">

<h1 className="display-4 ml-5 fw-bolder">Welcome Back</h1>

<p className="lead text-center">Enter Your Credential To Login</p>

<h5 className='mb-4 ml-3'>Are You New to This❓ Just Go with it ⤵️</h5>

<div className="container justify-content-center">

<NavLink to='/signup' className='btn btn-outline-dark rounded-pill pb-2 w-100'>Register</NavLink>

</div>

</div>

<div className="col-md-6 p-5">

<div className="display-6 fw-bolder mb-5 text-align-center">Login</div>

<form method='POST'>

<div className="form-group">

<label >Email address</label>

<input type="email" name='email' className="form-control" value={email} onChange={(e) => setEmail(e.target.value)} id="email" aria-describedby="emailHelp" />

<small id="emailHelp" className="form-text text-muted">We'll never share your email with anyone else.</small>

</div>

<div className="form-group">

<label >Password</label>

<input type="password" name='password' className="form-control" value={password} onChange={(e) => setPassword(e.target.value)} id="password" />

</div>

<div className="form-group form-check">

<input type="checkbox" className="form-check-input" id="exampleCheck1" />

<label className="form-check-label" >Remember Me</label>

</div>

<button type="submit" className="btn btn-primary" onClick={loginUser}>Submit</button>

</form>

</div>

</div>

</div>

</>

)

}

export default Login

………………………………………………………………………………………………

import React from 'react'

import { NavLink } from 'react-router-dom'

import Logo from '../assets/icons8-redux.svg'

function Navbar() {

return (

<>

{/\* //////////////////////////////////////////// \*/}

<nav className="navbar navbar-expand-lg navbar-light shadow">

<button className="navbar-toggler" type="button" data-toggle="collapse" data-target="/navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span className="navbar-toggler-icon"></span>

</button>

<div className="collapse navbar-collapse" id="navbarSupportedContent">

<ul className="navbar-nav mr-auto">

<li className="nav-item active">

<NavLink className="nav-link" to="/">Home <span className="sr-only">(current)</span></NavLink>

</li>

<li className="nav-item">

<NavLink className="nav-link" to="/about">About</NavLink>

</li>

<li className="nav-item">

<NavLink className="nav-link" to="/lecture">Lecture</NavLink>

</li>

<li className="nav-item">

<NavLink className="nav-link" to="/class">Class</NavLink>

</li>

<li className="nav-item">

<NavLink className="nav-link" to="/contact">Contact</NavLink>

</li>

<li className="nav-item">

<NavLink className="nav-link" to="/services">Services</NavLink>

</li>

</ul>

<NavLink className="navbar-brand fw-border fs-4 max-auto" to="/"><img src={Logo} alt="" />Skill-Up</NavLink>

<NavLink to='login' className="btn btn-outline-primary mr-2 px-4 rounded-pill">

<i className='fa fa-sign-in me-2'></i>Login</NavLink>

<NavLink to='/signup' className="btn btn-outline-primary ms-2 px-4 rounded-pill">

<i className='fa fa-user-plus me-2'></i>Register</NavLink>

</div>

</nav>

{/\* ///////////////////////////////////////////// \*/}

</>

)

}

export default Navbar

………………………………………………………………………………………………

import React from 'react'

function Services() {

return (

<>

<section id='service'>

<div className="container my-5 py-5">

<div className="row">

<div className="col-12">

<h3 className="fs-5 text-center mb-0">Our Services</h3>

<h1 className="display-6 text-center mb-4">Our <b> Awesome</b> Services</h1>

<hr className='w-25 mx-auto' />

</div>

</div>

<div className="row mt-5">

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<i className='fa fa-cogs fa-4x mb-4 text-primary'></i>

<h5 className="card-title mb-3 fs-4 fw-bold">Highly Skilled-Productivity </h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<i className='fa fa-mobile fa-4x mb-4 text-primary'></i>

<h5 className="card-title mb-3 fs-4 fw-bold">One to One Mentor </h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<i className='fa fa-file-code-o fa-4x mb-4 text-primary'></i>

<h5 className="card-title mb-3 fs-4 fw-bold">Expertise in Technology</h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

</div>

<div className="row mt-5">

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<i className='fa fa-laptop fa-4x mb-4 text-primary'></i>

<h5 className="card-title mb-3 fs-4 fw-bold">No leave (record) policy </h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<i className='fa fa-money fa-4x mb-4 text-primary'></i>

<h5 className="card-title mb-3 fs-4 fw-bold">Free Source</h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

<div className="col-md-4">

<div className="card p-3">

<div className="card-body text-center">

<i className='fa fa-star-half-o fa-4x mb-4 text-primary'></i>

<h5 className="card-title mb-3 fs-4 fw-bold">Doubt Clearing</h5>

<p className="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>

</div>

</div>

</div>

</div>

</div>

</section>

</>

)

}

export default Services

………………………………………………………………………………………………

import React, { useState } from 'react'

import { NavLink, useNavigate } from 'react-router-dom'

import axios from 'axios'

function Signup() {

// /////////////////////////////////////////////////////////////////////////////////////////////////////////

const history = useNavigate();

const [name, setName] = useState("")

const [email, setEmail] = useState("")

const [profession, setProfession] = useState("")

const [phone, setPhone] = useState("")

const [password, setPassword] = useState("")

const [cpassword, setCpassword] = useState("")

const handelSubmit = (e) => {

e.preventDefault();

console.log(name, email, profession, phone, password, cpassword)

axios.post("http://localhost:8000/register",

{

name: name,

email: email,

profession: profession,

phone: phone,

password: password,

cpassword: cpassword

}

).then(res => {

console.log(res.data)

if (res.data.code !== 200) {

alert("Error Found")

return

} else {

alert('Register Successfully... Redirecting to Login..')

history('/login')

}

}).catch(err => { console.log(err) })

}

// ////////////////////////////////////////////////////////////////////////////////////////////////////

return (

<>

<div className="container shadow my-5">

<div className="row justify-content-center-">

<div className="ml-5 col-md-5 d-flex flex-column text-dark align-item-center justify-content-center form order-2">

<h1 className="display-4 text-center fw-bolder">Welcome </h1>

<p className="lead text-center ">Enter Your Credential To Sign-Up</p>

<h5 className='mb-3 text-center'>Or</h5>

<h5 className='mb-3 ml-3 text-center'>Already User❓ Just Go with it ⤵️</h5>

<div className="container justify-content-center">

<NavLink to='/login' className='btn text-center justify-content-center btn-outline-dark rounded-pill mx-5 pb-2 w-100'>Login</NavLink>

</div>

</div>

<div className="col-md-6 p-5">

<form method='POST'>

<div className="form-group" id='register-form'>

<label >Name</label>

<input type="text" name='name' value={name} onChange={(e) => { setName(e.target.value) }} className="form-control" id="name" />

</div>

<div className="form-group">

<label>Email address</label>

<input type="email" name='email' value={email} onChange={(e) => { setEmail(e.target.value) }} className="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" />

</div>

<div className="form-group">

<label>profession</label>

<input type="text" name='profession' value={profession} onChange={(e) => { setProfession(e.target.value) }} className="form-control" id="profession" />

</div>

<div className="form-group">

<label >Phone</label>

<input type="text" name='phone' value={phone} onChange={(e) => { setPhone(e.target.value) }} className="form-control" id="phone" />

</div>

<div className="form-group">

<label>Password</label>

<input type="password" name='password' value={password} onChange={(e) => { setPassword(e.target.value) }} className="form-control" id="password" />

</div>

<div className="form-group">

<label>Confirm Password</label>

<input type="password" name='cpassword' value={cpassword} onChange={(e) => { setCpassword(e.target.value) }} className="form-control" id="cpassword" />

</div>

<button type="submit" onClick={handelSubmit} className="btn btn-primary">Submit</button>

</form>

</div>

</div>

</div>

</>

)

}

export default Signup

………………………………………………………………………………………………

***CSS parts of application***

@import "bootstrap/functions";

@import "bootstrap/variables";

@import "bootstrap/mixins";

$font-family-sans-serif-2: 'Poppins', sans-serif;

$font-family-serif: 'Source Serif Pro', serif;

body {

font-family: $font-family-sans-serif;

background-color: #272343;

}

p {

color: darken(#ccc, 10%);

font-weight: 300;

}

h1, h2, h3, h4, h5, h6,

.h1, .h2, .h3, .h4, .h5, .h6 {

font-family: $font-family-sans-serif;

}

a {

transition: .3s all ease;

&, &:hover {

text-decoration: none!important;

}

}

.content {

padding: 7rem 0;

background-color: #272343;

}

h2 {

font-size: 20px;

}

/\* 10 \*/

.site-blocks-cover {

background-size: cover;

background-repeat: no-repeat;

background-position: top;

background-attachment: fixed ;

position: relative;

&.overlay {

position: relative;

&:before {

position: absolute;

content: "";

left: 0;

bottom: 0;

right: 0;

top: 0;

background: rgba($black, .2);

}

}

&, & > .container > .row {

min-height: 600px;

height: calc(100vh);

}

h1 {

font-size: 8rem;

font-weight: 900;

line-height: 1;

@include media-breakpoint-down(md) {

font-size: 4rem;

}

}

.sub-text {

font-size: 1.4rem;

color: lighten($black, 50%);

font-weight: 300;

@include media-breakpoint-down(md) {

color: $black;

}

}

.img-wrap {

position: absolute;

z-index: -1;

width: calc(100% - 50%);

top: 0;

height: 100%;

z-index: 2;

right: 50%;

min-height: 600px;

overflow: hidden;

border-bottom-right-radius: 200px;

@include media-breakpoint-down(md) {

width: 100%;

right: 0%;

top: 0;

.hero-slider {

&:before {

position: absolute;

content: "";

background: $white;

opacity: .5;

z-index: 2;

top: 0;

left: 0;

right: 0;

bottom: 0;

}

}

}

.slide {

height: 100vh;

position: relative;

img {

position: absolute;

top: 0;

height: 100%;

width: 100%;

object-fit: cover;

}

}

}

.intro {

z-index: 3;

position: relative;

.heading {

margin-left: -150px;

@include media-breakpoint-down(md) {

margin-left: 0;

}

}

.text {

padding-left: 50px;

@include media-breakpoint-down(md) {

padding-left: 0;

}

}

}

}

.site-menu-toggle {

.menu-text {

position: relative;

top: -6px;

text-transform: uppercase;

}

}

.hero-slider {

.owl-nav {

position: absolute;

bottom: 120px;

right: 50px;

z-index: 100;

.owl-prev, .owl-next {

width: 50px;

height: 50px;

text-align: center;

line-height: 50px;

border-radius: 50%;

background: $primary;

color: $white;

margin-bottom: 10px;

font-size: 1.3rem;

span {

color: $white;

font-size: 30px;

}

&:active, &:focus {

outline: none;

}

}

}

}

.btn-pill {

border-radius: 30px;

padding: 15px 30px;

border-color: $white;

color: $white;

&:hover {

background: $white;

border-color: $white;

color: $black;

}

}

# REFERENCES

* <https://reactnative.dev/docs/getting-started>.
* <https://reactnative.dev/docs/components-and-apis>
* <https://reactnative.dev/docs/accessibilityinfo>
* <https://reactnative.dev/architecture/overview>
* <https://blog.expo.io/good-practices-why-you-should-use-javascript-whenever-possible-with-react-native-26478ec22334>
* <https://nodejs.org/en/docs/>
* <https://www.w3schools.com/js/default.asp>
* <https://www.youtube.com/watch?v=0-S5a0eXPoc&ab_channel=ProgrammingwithMosh>
* <https://docs.expo.dev/workflow/expo-go/>