

FULL STACK PROJECT REPORT

On

“Unacademy 2.0”

Submitted by

**Prateek Sagar Richhariya
(191500591)**

**Krishna Bansal
(191500409)**

**Prashant Sahu
(191500581)**

**Ritik Gupta
(191500659)**

Department of Computer Engineering & Applications
Institute of Engineering & Technology



**GLA University
Mathura- 281406, INDIA**



Department of computer Engineering and Applications
GLA University, Mathura

17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

We hereby declare that the work which is being presented in the Full Stack Project “**Unacademy 2.0**”, in partial fulfillment of the requirements for Full Stack Project viva voce, is an authentic record of our own work carried by the team members under the supervision of our mentor Mr. Pankaj Kapoor.

Group Members: Krishna Bansal (191500409)

Ritik Gupta (191500659)

Prashant Sahu (191500581)

Prateek Sagar Richhariya (191500591)

Course: B.Tech (Computer Science and Engineering)

Year: 3rd

Semester: 5th

Supervised By:

Mr. Pankaj Kapoor, Assistant Professor,

GLA University, Department of Computer Engineering & Application



Department of computer Engineering and Applications

GLA University, Mathura

17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,

Mathura – 281406

Certificate

This is to certify that the above statements made by the candidates are correct to the best of my/our knowledge and belief.

Supervisor

Mr. Pankaj Kapoor

Technical Trainer

Dept of CEA, GLA University

Program Coordinator

(Mr. Shashi Shekar)

About the Project

Unacademy2.0 is a e-learning platform which offers a wide variety of courses, certification-courses and many more where students can study with high qualified teachers and students have lot of choices and clear our doubts at any time. Live lectures are also there where students can make live interactions with teachers. Recorded videos are also provided to students.

Unacademy2.0 objective is to provide best service among all who want to study anything at any time which are available at free of cost. User can also give mock papers and prepare our interviews and also make the people aware about what type of new technology is currently working in the market.

.

Motivation

There are many websites who provide courses and study material but finding the good and secure websites is difficult and tedious task so to solve this problem of the user we are making a project which provide an e-learning platform to the user where they can find all their lectures or study material in one place and can easily access them in free.

The purpose is to make this project is to provide a platform where any one can learn anything at any time.

- As we know there are many sites available to give such type of information but we will provide a every new technology what the learns want to learn.
- All study material also provided here so any chances of any doubts may not come.
- There is section where we call discussion forum where any one can discuss and clear our doubts.

Requirements

a). Software Requirements:

- Technology Implemented: Full Stack Web Development
- Languages/Technologies Used: MERN Stack
- IDE Used: Visual Studio Code
- Web Browser: Google Chrome
- GitHub: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. GitHub Repository: A GitHub repository can be used to store a development project. It can contain folders and any type of files (HTML, CSS, JavaScript, Documents, Data, Images). A GitHub repository should also include a license file and a README file about the project. A GitHub repository can also be used to store ideas, or any resources that you want to share.
- Visual Studio Code: Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. [7] Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. Microsoft has released Visual Studio Code's source code on the VS Code repository of GitHub.com, under the permissive MIT License, while the compiled binaries are freeware.

b). Hardware Requirements:

- Processor Required: Intel i5
- Operating System: Windows 10
- RAM: 8GB
- Hardware Devices: Computer System
- Hard Disk: 256GB

Acknowledgement

We thank the almighty for giving us the courage and perseverance in completing the project. This project itself is an acknowledgement for all those people who have given us their heartfelt co-operation in making this project a grand success. We extend our sincere thanks to Mr. Pankaj Kapoor, Assistant Professor at “GLA University, Mathura” for providing his valuable guidance at every stage of this project work. We are profoundly grateful towards the unmatched services rendered by him. And last but not least, we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation in doing the main project.

Unacademy2.0

Abstract

Unacademy2.0 is a e-learning platform which offers a wide variety of courses, certification-courses and many more where students can study with high qualified teachers and students have lot of choices and clear our doubts at any time. Live lectures are also there where students can make live interactions with teachers. Recorded videos are also provided to students.

Unacademy2.0 objective is to provide best service among all who want to study anything at any time which are available at free of cost. User can also give mock papers and prepare our interviews and also make the people aware about what type of new technology is currently working in the market.

.

Contents

Acknowledgment.....08

Abstract.....09

1. Introduction:

Introduction to MERN Stack.....12-13

Pre-requisites.....14

2. Technologies Used:

HTML.....15

CSS.....16

JS.....17-18

3. List of Figures.....20-26

4. Software Testing.....	27-31
5.Conclusion.....	32
6.Bibliography.....	33

Chapter 1

Introduction

Today Developers around the world are making efforts to enhance user experience of using application as well as to enhance the developer's workflow of designing applications to deliver projects and rollout change requests under strict timeline. Stacks can be used to build web applications in the shortest span of time. The stacks used in web development are basically the response of software engineers to current demands. They have essentially adopted pre-existing frameworks (including JavaScript) to make their lives easier. While there are many, MEAN and MERN are just two of the popular stacks that have evolved out of JavaScript. Both stacks are made up of open-source components and offer an end-to-end framework for building comprehensive web apps that enable browsers to connect with databases. The common theme between the two is JavaScript and this is also the key benefit of using either stack. One can basically avoid any syntax errors or any confusion by just coding in one programming language, JavaScript. Another advantage of building web projects with MERN is the fact that one can benefit from its enhanced flexibility. In order to understand MERN stack, we need to understand the four components that make up the MERN namely

– MongoDB, Express.js, React and Node.js

The MERN architecture allows you to easily construct a 3-tier architecture (frontend, backend, database) entirely using JavaScript and JSON.

React.js Front End

The top tier of the MERN stack is React.js, the declarative JavaScript framework for creating dynamic client-side applications in HTML. React lets you build up complex interfaces through simple Components, connect them to data on your backend server, and render them as HTML.

React's strong suit is handling stateful, data-driven interfaces with minimal code and minimal pain, and it has all the bells and whistles you'd expect from a modern web framework: great support for forms, error handling, events, lists, and more.

Express.js and Node.js Server Tier

The next level down is the Express.js server-side framework, running inside a Node.js server. Express.js bills itself as a “fast, unopinionated, minimalist web framework for Node.js,” and that is indeed exactly what it is. Express.js has powerful models for URL routing (matching an incoming URL with a server function), and handling HTTP requests and responses.

By making XML HTTP Requests (XHRs) or GETs or POSTs from your React.js front-end, you can connect to Express.js functions that power your application. Those functions in turn use MongoDB's Node.js drivers, either via callbacks for using Promises, to access and update data in your MongoDB database.

MongoDB Database Tier

If your application stores any data (user profiles, content, comments, uploads, events, etc.), then you're going to want a database that's just as easy to work with as React, Express, and Node.

That's where MongoDB comes in: JSON documents created in your React.js front end can be sent to the Express.js server, where they can be processed and (assuming they're valid) stored directly in MongoDB for later retrieval. Again, if you're building in the cloud, you'll want to look at Atlas.

Pre-requisite

Hands-on knowledge of JavaScript, HTML and CSS is essential before working on the concepts for making of webpages. Make sure that you have the browser or chrome installed and running before opening website.

Chapter 2

Technologies Used

HTML

The **HyperText Markup Language**, or **HTML** is the standard [markup language](#) for documents designed to be displayed in a [web browser](#). It can be assisted by technologies such as [Cascading Style Sheets](#) (CSS) and [scripting languages](#) such as [JavaScript](#).

[Web browsers](#) receive HTML documents from a [web server](#) or from local storage and [render](#) the documents into multimedia web pages. HTML describes the structure of a [web page semantically](#) and originally included cues for the appearance of the document.

[HTML elements](#) are the building blocks of HTML pages. With HTML constructs, [images](#) and other objects such as [interactive forms](#) may be embedded into the rendered page. HTML provides a means to create [structured documents](#) by denoting structural [semantics](#) for text such as headings, paragraphs, lists, [links](#), quotes and other items. HTML elements are delineated by *tags*, written using [angle brackets](#). Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](#) such as [JavaScript](#), which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](#) (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

Key Components Of HTML

In 1980, physicist [Tim Berners-Lee](#), a contractor at [CERN](#), proposed and prototyped [ENQUIRE](#), a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an [Internet](#)-based [hypertext](#) system.^[3] Berners-Lee specified HTML and wrote the browser and server software in late 1990. That year, Berners-Lee and CERN data systems engineer [Robert Cailliau](#) collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes^[4] from 1990 he listed^[5] "some of the many areas in which hypertext is used" and put an encyclopedia first.

The first publicly available description of HTML was a document called "[HTML Tags](#)", first mentioned on the Internet by Tim Berners-Lee in late 1991.^{[6][7]} It describes 18 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by [SGMLguid](#), an in-house [Standard Generalized Markup Language](#) (SGML)-based documentation format at CERN. Eleven of these elements still exist in HTML 4.^[8]

HTML is a [markup language](#) that [web browsers](#) use to interpret and [compose](#) text, images, and other material into visual or audible web pages. Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of [CSS](#). Many of the text elements are found in the 1988 ISO technical report TR 9537 *Techniques for using SGML*, which in turn covers the features of early text formatting languages such as that used by the [RUNOFF command](#) developed in the early 1960s for the [CTSS](#) (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and markup; HTML has been progressively moved in this direction with CSS.

Berners-Lee considered HTML to be an application of SGML. It was formally defined as such by the [Internet Engineering Task Force](#) (IETF) with the mid-1993 publication of the first proposal for an HTML specification, the "Hypertext Markup Language (HTML)" Internet Draft by Berners-Lee and [Dan Connolly](#), which included an SGML [Document type definition](#) to define the grammar.^{[9][10]} The draft expired after six months, but was notable for its acknowledgment of the [NCSA](#)

[Mosaic](#) browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes. Similarly, [Dave Raggett](#)'s competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.^[11]

After the HTML and HTML+ drafts expired in early 1994, the IETF created an HTML Working Group, which in 1995 completed "HTML 2.0", the first HTML specification intended to be treated as a standard against which future implementations should be based.^[12]

Further development under the auspices of the IETF was stalled by competing interests. Since 1996, the HTML specifications have been maintained, with input from commercial software vendors, by the [World Wide Web Consortium](#) (W3C).^[13] However, in 2000, HTML also became an international standard (ISO/IEC 15445:2000). HTML 4.01 was published in late 1999, with further errata published through 2001. In 2004, development began on HTML5 in the [Web Hypertext Application Technology Working Group](#) (WHATWG), which became a joint deliverable with the W3C in 2008, and completed and standardized on 28 October 2014.^[14]

CSS

Cascading Style Sheets (CSS) is a [style sheet language](#) used for describing the [presentation](#) of a document written in a [markup language](#) such as [HTML](#).^[1] CSS is a cornerstone technology of the [World Wide Web](#), alongside HTML and [JavaScript](#).^[2]

CSS is designed to enable the separation of presentation and content, including [layout](#), [colors](#), and [fonts](#).^[3] This separation can improve content [accessibility](#), provide more flexibility and control in the specification of presentation characteristics, enable multiple [web pages](#) to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be [cached](#) to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or [screen reader](#)), and on [Braille-based](#) tactile devices. CSS also has rules for alternate formatting if the content is accessed on a [mobile device](#).^[4]

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the [World Wide Web Consortium](#) (W3C). Internet media type ([MIME type](#)) `text/css` is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free [CSS validation service](#) for CSS documents.^[5]

In addition to HTML, other markup languages support the use of CSS including [XHTML](#), [plain XML](#), [SVG](#), and [XUL](#).

CSS has a simple [syntax](#) and uses a number of English keywords to specify the names of various style properties.

A style sheet consists of a list of *rules*. Each rule or rule-set consists of one or more [selectors](#), and a [declaration block](#).

Selector[[edit](#)]

In CSS, *selectors* declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

Selectors may apply to the following:

- all [elements](#) of a specific type, e.g. the second-level headers [h2](#)
- elements specified by [attribute](#), in particular:
 - *id*: an identifier unique within the document, identified with a hash prefix e.g. `#id`
 - *class*: an identifier that can annotate multiple elements in a document, identified with a period prefix e.g. `.classname`

- elements depending on how they are placed relative to others in the [document tree](#).

Classes and IDs are case-sensitive, start with letters, and can include alphanumeric characters, hyphens, and underscores. A class may apply to any number of instances of any elements. An ID may only be applied to a single element.

Pseudo-classes are used in CSS selectors to permit formatting based on information that is not contained in the document tree. One example of a widely used pseudo-class is `:hover`, which identifies content only when the user "points to" the visible element, usually by holding the mouse cursor over it. It is appended to a selector as in `a:hover` or `#elementid:hover`. A pseudo-class classifies document elements, such as `:link` or `:visited`, whereas a *pseudo-element* makes a selection that may consist of partial elements, such as `::first-line` or `::first-letter`.^[6]

Selectors may be combined in many ways to achieve great specificity and flexibility.^[7] Multiple selectors may be joined in a spaced list to specify elements by location, element type, id, class, or any combination thereof. The order of the selectors is important. For example, `div.myClass {color: red;}` applies to all elements of class `myClass` that are inside `div` elements, whereas `.myClass div {color: red;}` applies to all `div` elements that are inside elements of class `myClass`. This is not to be confused with concatenated identifiers such as `div.myClass {color: red;}` which applies to `div` elements of class `myClass`.

The following table provides a summary of selector syntax indicating usage and the version of CSS that introduced it.^[8]

JS

JavaScript ([/'dʒɑ:və skɪpt/](#)),^[10] often abbreviated as **JS**, is a [programming language](#) that conforms to the [ECMAScript](#) specification.^[11] JavaScript is [high-level](#), often [just-in-time compiled](#) and [multi-paradigm](#). It has [dynamic typing](#), [prototype-based object-orientation](#) and [first-class functions](#).

Alongside [HTML](#) and [CSS](#), JavaScript is one of the core technologies of the [World Wide Web](#).^[12] Over 97% of [websites](#) use it [client-side](#) for [web page](#) behavior,^[13] often incorporating third-party [libraries](#).^[14] All major [web browsers](#) have a dedicated [JavaScript engine](#) to execute the code on the [user's](#) device.

As a multi-paradigm language, JavaScript supports [event-driven](#), [functional](#), and [imperative programming styles](#). It has [application programming interfaces](#) (APIs) for working

with text, dates, [regular expressions](#), standard [data structures](#), and the [Document Object Model](#) (DOM).

The ECMAScript standard does not include any [input/output](#) (I/O), such as [networking](#), [storage](#), or [graphics](#) facilities. In practice, the web browser or other [runtime system](#) provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but they are now core components of some [servers](#) and a variety of [applications](#). The most popular runtime system for this usage is [Node.js](#).

Although there are similarities between JavaScript and [Java](#), including language name, [syntax](#), and respective [standard libraries](#), the two languages are distinct and differ greatly in design.

The use of JavaScript has expanded beyond its [web browser](#) roots. [JavaScript engines](#) are now embedded in a variety of other software systems, both for [server-side](#) website deployments and non-browser [applications](#).

Initial attempts at promoting server-side JavaScript usage were [Netscape Enterprise Server](#) and [Microsoft's Internet Information Services](#),^{[41][42]} but they were small niches.^[43] Server-side usage eventually started to grow in the late 2000s, with the creation of [Node.js](#) and [other approaches](#).^[43]

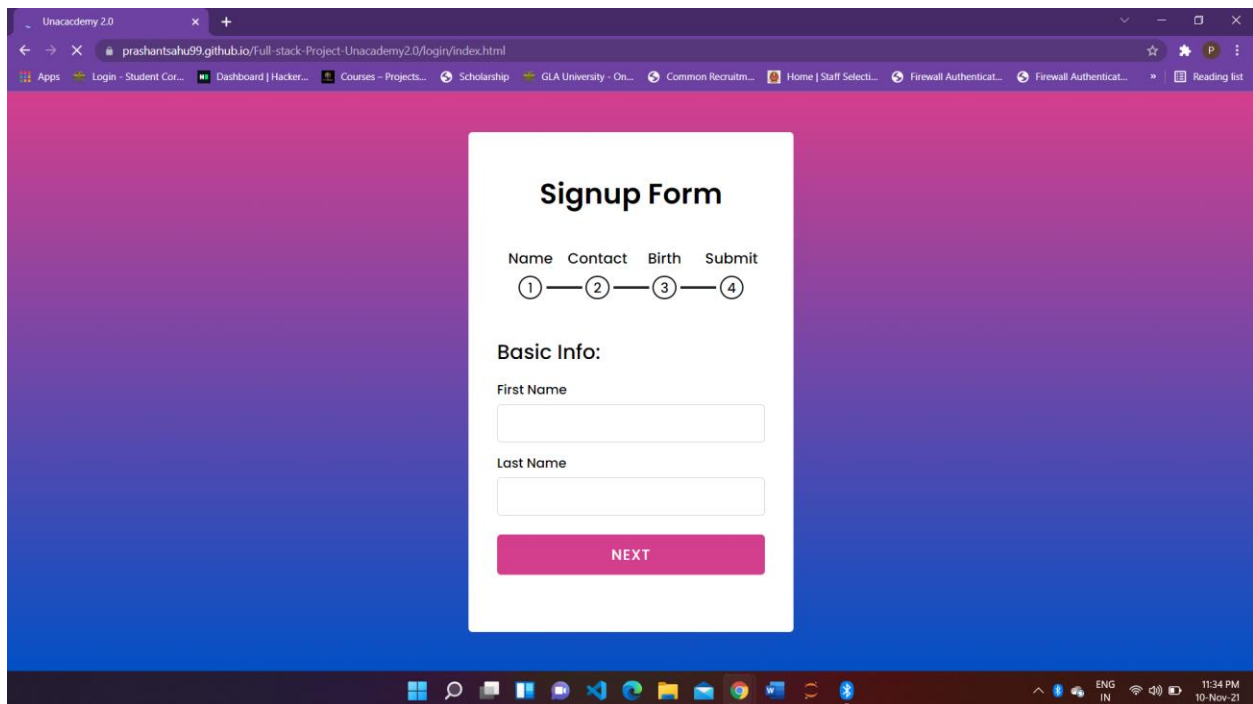
[Electron](#), [Cordova](#), [React Native](#), and other [application frameworks](#) have been used to create many applications with behavior implemented in JavaScript. Other non-browser applications include [Adobe Acrobat](#) support for scripting [PDF](#) documents^[44] and [GNOME Shell](#) extensions written in JavaScript.^[45]

JavaScript has recently begun to appear in some [embedded systems](#), usually by leveraging Node.js.

Chapter 3

List of Figures

1. SignUp Page



The screenshot shows a web browser window with the URL `prashantsahu99.github.io/full-stack-Project-Unacademy2.0/login/index.html`. The page features a purple-to-blue gradient background. In the center is a white box titled "Signup Form". Above the form, a progress indicator shows four steps: "Name", "Contact", "Birth", and "Submit", each with a circled number (1, 2, 3, 4) and connected by horizontal lines. The "Basic Info:" section contains two input fields: "First Name" and "Last Name". Below these fields is a pink button labeled "NEXT". The browser's address bar and taskbar are visible at the top and bottom of the window.

Unacademy 2.0

prashantsahu99.github.io/full-stack-Project-Unacademy2.0/login/index.html

Apps Login - Student Cor... Dashboard | Hacker... Courses - Projects... Scholarship GLA University - On... Common Recruitm... Home | Staff Select... Firewall Authentica... Firewall Authentica... Reading list

Signup Form

Name Contact Birth Submit

① — ② — ③ — ④

Basic Info:

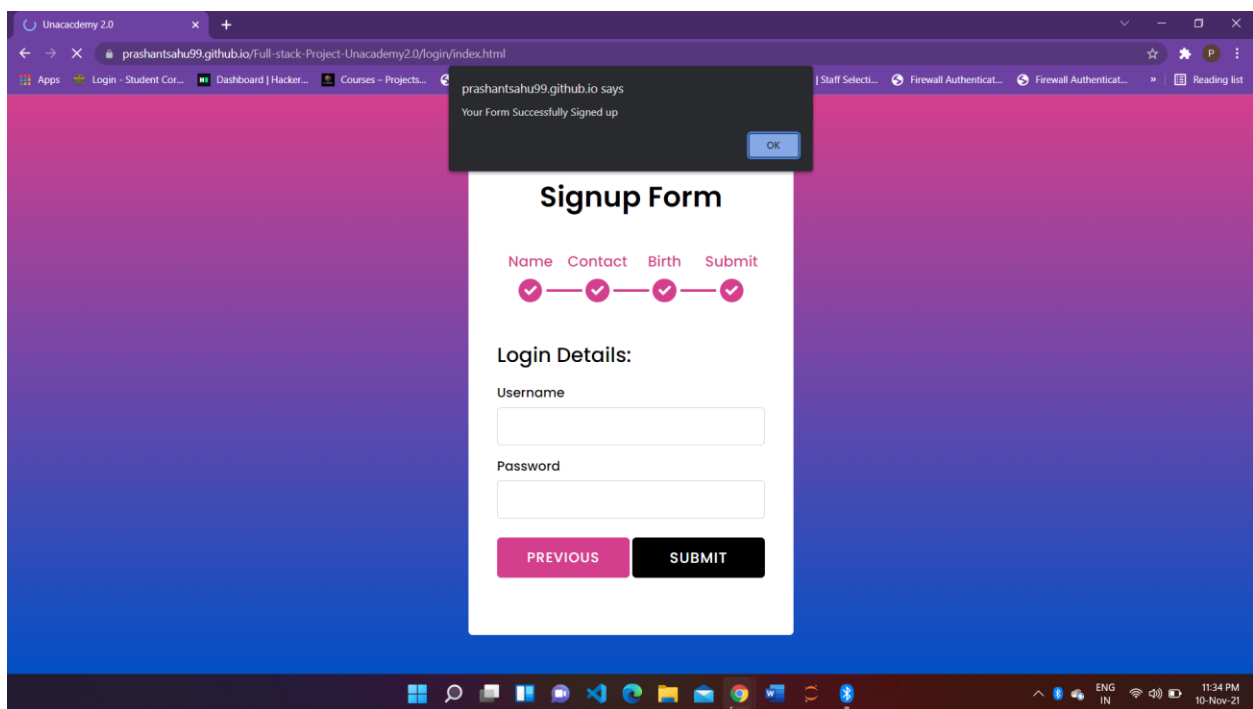
First Name

Last Name

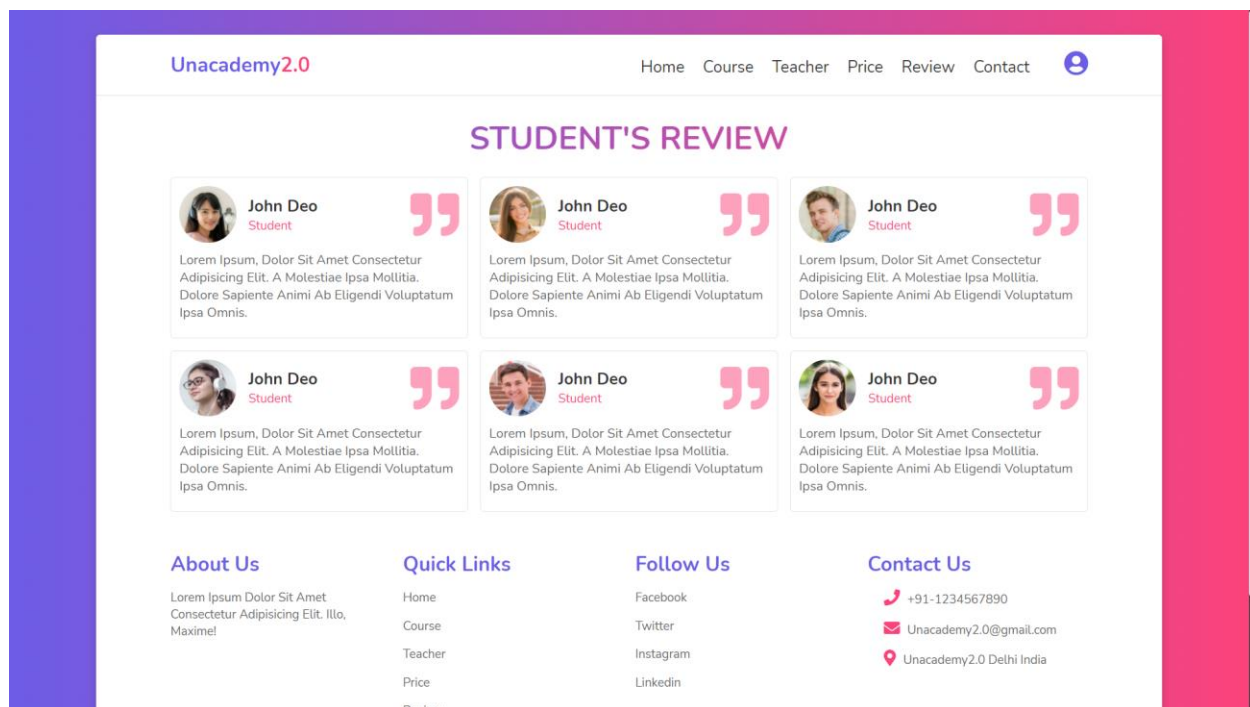
NEXT

11:34 PM 10-Nov-21

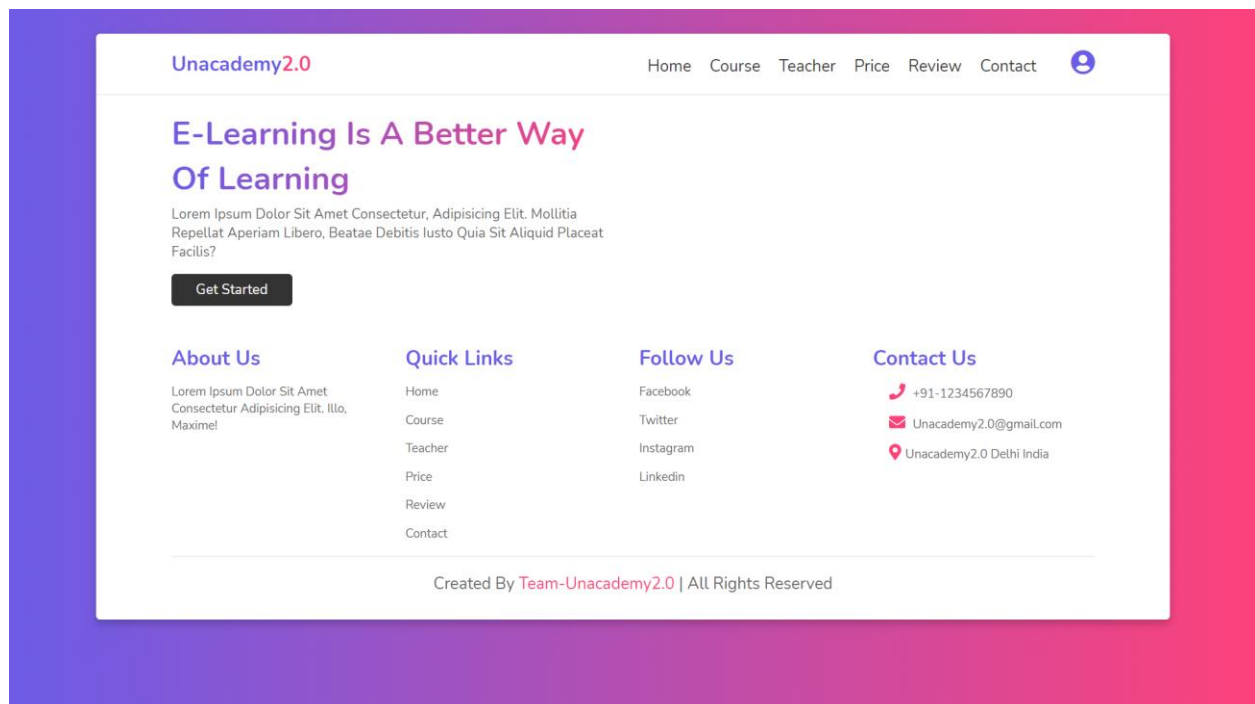
2. Login Page



3. Searching someone's profile




4. Home Page



5.

₹4500



★★★★☆

JAVA Programming

Lorem Ipsum Dolor, Sit Amet Consectetur
 Adipiscing Elit. Cumque Ullam Omnis Odit
 Accusamus Doloremque? Cum Aliquid Unde Nulla.
 Vitae, Adipisci?


Learn More

2 Hours

6 Months

12 Modules

₹3500



★★★★☆

Data-Structure

Lorem Ipsum Dolor, Sit Amet Consectetur
 Adipiscing Elit. Cumque Ullam Omnis Odit
 Accusamus Doloremque? Cum Aliquid Unde Nulla.
 Vitae, Adipisci?


Learn More

2 Hours

6 Months

12 Modules

₹5000



★★★★☆

Beta-Labs [By Pankaj Sir]

Lorem Ipsum Dolor, Sit Amet Consectetur
 Adipiscing Elit. Cumque Ullam Omnis Odit
 Accusamus Doloremque? Cum Aliquid Unde Nulla.
 Vitae, Adipisci?

Learn More

2 Hours

6 Months

12 Modules

About Us

Lorem Ipsum Dolor Sit Amet
 Consectetur Adipiscing Elit. Illo,
 Maxime!

Quick Links

- Home
- Course
- Teacher
- Price
- Review
- Contact

Follow Us

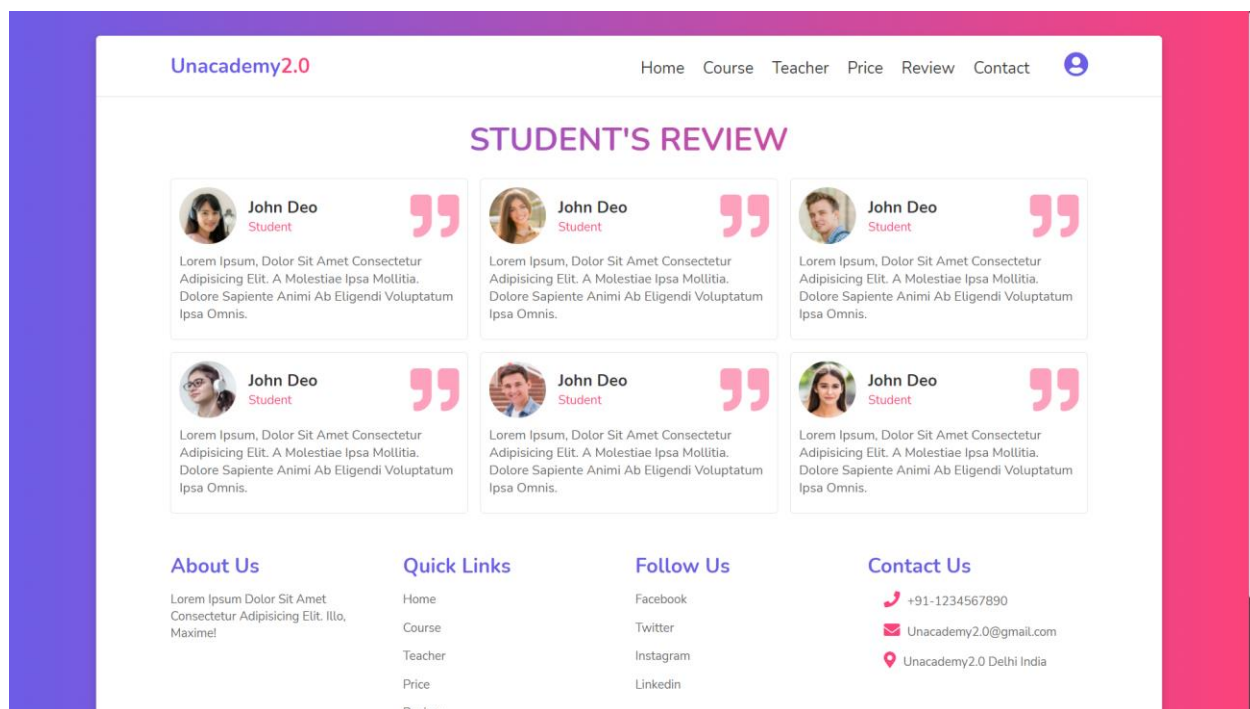
- Facebook
- Twitter
- Instagram
- LinkedIn

Contact Us

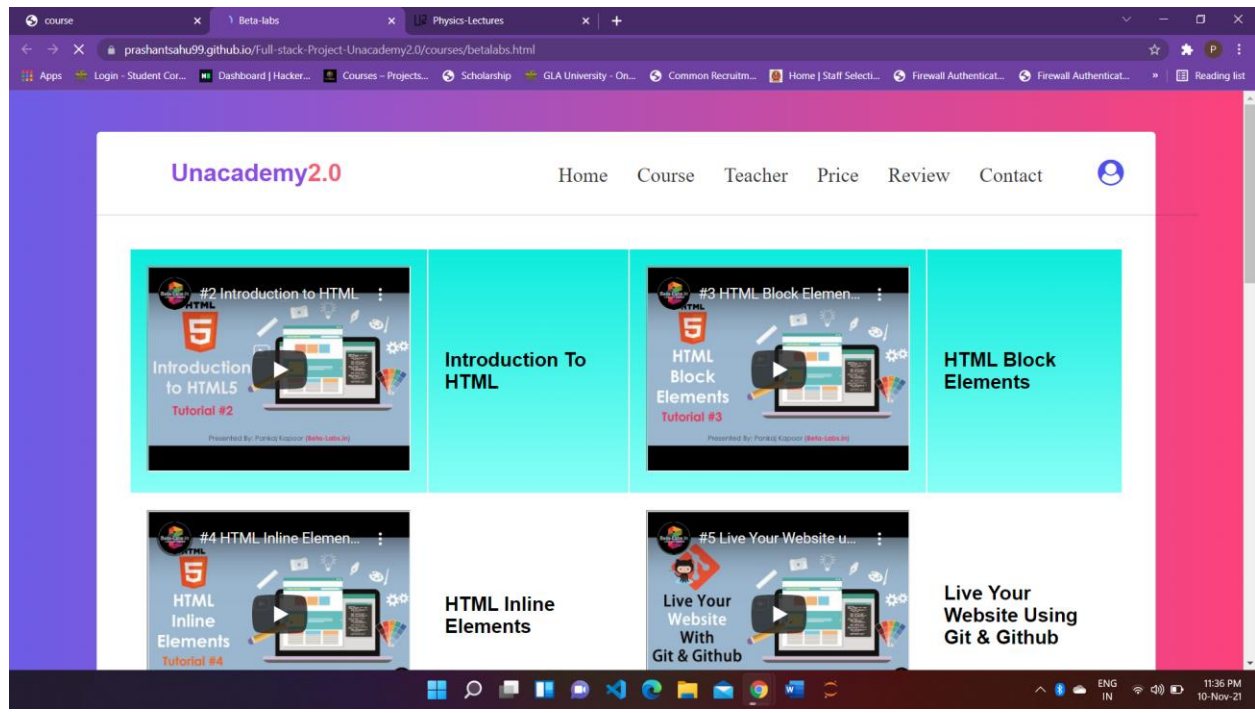
- +91-1234567890
- Unacademy2.0@gmail.com
- Unacademy2.0 Delhi India

Created By Team-Unacademy2.0 | All Rights Reserved

6.



7.



Chapter 4

Software Testing

Once source code has been generated, software must be tested to uncover as many errors as possible before delivery. It is very important to work the system successfully and achieve high quality of software. Testing include designing a series of test cases that have a high likelihood of finding errors by applying software-testing techniques. System testing makes logical assumptions that if all the parts of the system are correct, the goal will be successfully achieved. The system should be checked logically. Validations and cross checks should be there. Avoid duplications of record that cause redundancy of data. In other Words, Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. It is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

The preliminary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing and modifications are eased. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are the hallmark of good programs, obscurity, cleverness, and complexity are indications of inadequate design and misdirected thinking. Source code clarity is enhanced by structured coding techniques, by good coding style, by, appropriate supporting documents, by good internal comments, and by feature provided in modern programming languages. The implementation team should be provided with a well-defined set of software requirement, an architectural design specification, and a detailed design description. Each team member must understand the objectives of implementation.

4.1 TERMINOLOGY

Error The term error is used in two ways. It refers to the difference between the actual output of software and the correct output, in this interpretation, error is essential a measure of the difference between actual and ideal. Error is also used to refer to human action that result in software containing a defect or fault.

Fault is a condition that causes to fail in performing its required function. A fault is a basic reason for software malfunction and is synonymous with the commonly used term Bug.

Failure is the inability of a system or component to perform a required function according to its specifications. A software failure occurs if the behavior of the software is the different from the specified behavior. Failure may be caused due to functional or performance reasons.

4.2 TYPES OF TESTING

a. Unit Testing The term unit testing comprises the sets of tests performed by an individual programmer prior to integration of the unit into a larger system. A program unit is usually small enough that the programmer who developed it can test it in great detail, and certainly in greater detail than will be possible when the unit is integrated into an evolving software product. In the unit testing the programs are tested separately, independent of each other. Since the check is done at the program level, it is also called program teasing.

b. Module Testing A module and encapsulates related component. So can be tested without other system module.

c. Subsystem Testing Subsystem testing may be independently design and implemented common problems are sub-system interface mistake in this checking we concenton it. There are four categories of tests that a programmer will typically perform on a program unit.

i Functional test

ii Performance test

iii Stress test

iv Structure test

Functional Test Functional test cases involve exercising the code with Nominal input values for which expected results are known; as well as boundary values (minimum values, maximum values and values on and just outside the functional boundaries) and special values.

Performance Test Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, response time, and device utilization by the program unit. A certain amount of avoid expending too much effort on fine-tuning of a program unit that contributes little to the overall performance of the entire system. Performance testing is most productive at the subsystem and system levels.

Stress Test Stress test are those designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

Structure Test Structure tests are concerned with exercising the internal logic of a program and traversing particular execution paths. Some authors refer collectively to functional performance and stress testing as “black box” testing. While structure testing is referred to as “white box” or “glass box” testing. The major activities in structural testing are deciding which path to exercise, deriving test data to exercise those paths, determining the test coverage criterion to be used, executing the test, and measuring the test coverage achieved when the test cases are exercised.

Chapter 5

Conclusion

We have completed our project within time limit with the coordination of our team members under the supervision of our mentor Mr. Pankaj Kapoor.

Our project repository is available at

<https://prashantsahu99.github.io/Full-stack-Project-Unacademy2.0/>

Chapter 6

Bibliography

www.google.com

www.geeksforgeeks.org

www.youtube.com

www.w3schools.com

www.beta-labs.in