MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

SUMY STATE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

BACHELOR'S THESIS

On the topic:

"Travel Website based on Bootstrap Framework and backend"

Head of Department **Dovbysh A.S.**

Student Group IN-75AH **Taweh B. Johnson**

Supervisor **Protsenko O. B.**

SUMY 2021

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

SUMYSTATEUNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Approved _____

Head of department Dovbysh A.S.

"_____” 2021

Task of Bachelor Thesis

Fourth-year student, group IN-75AH specialty “ Programming of intelligent information systems” Taweh B. Johnson

On the topic: “Travel Website based on Bootstrap Framework and backend”

Approved by order of the Sumy State University

№_____ of _____ 2021

1.1 Contents Explanatory Note: 1) An informational review of the literature; 1.1) Bootstrap Framework; 1.2) Clientside/Serverside programming Research; 1.3) Problems and statements ; 2) Programming instruments; 2.1) Framework Bootstrap; 2.2) PhpMyAdmin 2.3) Brackets 2.4) JavaScript 2.5) Cascading Style Sheets 3) Website realization ; 3.1) Modeling of website; 3.2); Website realization ; 3.3) Website HTML code; Conclusion, References.

Date of issuance of the task

"_____" 2021

Supervisor _____ Protsenko O. B.

Task adopted to be implemented by _____ Taweh B. Johnson

ABSTRACT

Note: 33 pages, 17 figures , 15 sources literature, 1 website.

Object of study - the developing of Travel Website based on Bootstrap Framework and backend.

Purpose - To develop a travel website.

Research methods – Qualitative research method was utilized for the travel website based on Bootstrap Framework and backend.

Results – travel website developed based on bootstrap framework.

Keywords: Bootstrap, web programming , frontend, backend, prototype.

Table of Contents

INTRODUCTION

We live in a world where you can access information on a smartphone or any device with internet connectivity concerning any place on this globe, view pictures and read about the lifestyle of people who live in a particular area. But nothing surpasses having first hand experience of a country by traveling there and experiencing life there.

Learning the language people speak and living the culture of the people in a country broadens your horizons. Traveling to experience the world is what young people want to do but factors such as conflict, strong ties to families amongst others prevent young people from traveling. As a result of this, I will develop a website from scratch that will ignite the passion in young people who are still hesitating to leave their confort zones to travel and conquer the world.

According to the World Tourism Organization 2016 Global Report on the Power of Youth Travel, young people attract other visitors to the destinations they visit. It was estimated that each young person taking a course in higher education was visited by an average of 1.3 people during their stay generating an additional AUD 1.2 billion for the Australian Economy each year. This website would attract other young people to various destinations other young people have been by sharing their adventures and life-long experiences.

In order to build this website , we will utilize Bootstrap which is the most popular Front-end framework for HTML, CSS, and JS to develop responsive projects on the web. We will also create a database that would store user’s contact information using phpmyAdmin.

1 INFORMATION REVIEW

1.1 Bootstrap Framework

Bootstrap is a web framework that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements.

The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. Before Bootstrap, various libraries were used for interface development, which led to inconsistencies and a high maintenance burden. According to Twitter developer Mark Otto:

A super small group of developers and I got together to design and build a new internal tool and saw an opportunity to do something more. Through that process, we saw ourselves build something much more substantial than another internal tool. Months later, we ended up with an early version of Bootstrap as a way to document and share common design patterns and assets within the company.

After a few months of development by a small group, many developers at Twitter began to contribute to the project as a part of Hack Week, a hackathon-style week for the Twitter development team.

It was renamed from Twitter Blueprint to Bootstrap, and released as an open source project on August 19, 2011.It has continued to be maintained by Mark Otto,

Jacob Thornton, and a small group of core developers, as well as a large community of contributors.

Bootstrap is a HTML, CSS & JS Library that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking.

Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap also comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields.

The most prominent components of Bootstrap are its layout components, as they affect an entire web page. The basic layout component is called "Container", as every other element in the page is placed in it. Developers can choose between a fixed-width container and a fluid-width container. While the latter always fills the width of the web page, the former uses one of the five predefined fixed widths, depending on the size of the screen showing the page:

1.2 Clientside scripting Language

In web development, 'client side' refers to everything in a web application that is displayed or takes place on the client (end user device). This includes what the user sees, such as text, images, and the rest of the UI, along with any actions that an application performs within the user's browser.

Markup languages like HTML and CSS are interpreted by the browser on the client side. In addition, many contemporary developers are including client-side processes in their application architecture and moving away from doing everything on the server side; business logic for dynamic webpages, for instance, usually runs client side in a modern web application. Client-side processes are almost always written in JavaScript.

Much of the Internet is based on the client-server model. In this model, user devices communicate via a network with centrally located servers to get the data they need, instead of communicating with each other. End user devices such as laptops, smartphones, and desktop computers are considered to be 'clients' of the servers, as if they were customers obtaining services from a company. Client devices send requests to the servers for webpages or applications, and the servers serve up responses.

The client-server model is used because servers are typically more powerful and more reliable than user devices. They also are constantly maintained and kept in controlled environments to make sure they're always on and available; although individual servers may go down, there are usually other servers backing them up. Meanwhile, users can turn their devices on and off, or lose or break their devices, and it should not impact Internet service for other users.

Servers can serve multiple client devices at once, and each client device sends requests to multiple servers in the course of accessing and browsing the Internet.

Typically, a client is a computer application, such as a web browser, that runs on a user's local computer, smartphone, or other device, and connects to a server as necessary. Operations may be performed client-side because they require access to information or functionality that is available on the client but not on the server, because the user needs to observe the operations or provide input, or because the server lacks the processing power to perform the operations in a timely manner for all of the clients it serves.

Additionally, if operations can be performed by the client, without sending data over the network, they may take less time, use less bandwidth, and incur a lesser security risk.

When the server serves data in a commonly used manner, for example according to standard protocols such as HTTP or FTP, users may have their choice of a number of client programs (e.g. most modern web browsers can request and receive data using both HTTP and FTP). In the case of more specialized applications, programmers may write their own server, client, and communications protocol which can only be used with one another.

Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be termed client-side operations. Client side Mechanism:

- The user opens his web browser (client)
- The user starts browsing (for example http://bbc.com)
- The client forwards this request to the server, for accessing their web page.
- The server then acknowledges the request and replies back to the client program.(An access link to that web page)
- The client then receives the page source and renders it. (Into a viewable/under a stable website) Now the user types into the search bar
- The client then submits data to the server
- The server processes the data and replies back with a related search result

- The client again renders it back for the user's view
- The user gets access to the requested link.

Client-side Uses

- Makes interactive web pages
- Make stuff work dynamically
- Interact with temporary storage
- Works as an interface between user and server
- Sends requests to the server
- Retrieval of data from Server
- Interact with local storage
- Provides remote access for client-server program

Client-side Languages Example

There are many client-side scripting languages too.

- JavaScript
- VBScript
- HTML (Structure)
- CSS (Designing)
- AJAX
- jQuery etc.

(Some other languages also can be used on the basis of the modeling/designing /graphics/animations and for extra functionalities.)

Client-side Example

Figure 1.2

```
// sample HTML code
<html>
<head>
    <title>Client Side </title>
</head>
<body>
    <h1>
        Hello C# Corner
    </h1>
     </body>
     </html>
```

Much of the Internet is based on the client-server model. In this model, user devices communicate via a network with centrally located servers to get the data they need, instead of communicating with each other. End user devices such as laptops, smartphones, and desktop computers are considered to be 'clients' of the servers, as if they were customers obtaining services from a company. Client devices send requests to the servers for webpages or applications, and the servers serve up responses.

The client-server model is used because servers are typically more powerful and more reliable than user devices. They also are constantly maintained and kept in controlled environments to make sure they're always on and available; although individual servers may go down, there are usually other servers backing them up. Meanwhile, users can turn their devices on and off, or lose or break their devices, and it should not impact Internet service for other users.

Servers can serve multiple client devices at once, and each client device sends requests to multiple servers in the course of accessing and browsing the Internet.

Much like with client side, 'server side' means everything that happens on the server, instead of on the client. In the past, nearly all business logic ran on the

server side, and this included rendering dynamic webpages, interacting with databases, identity authentication, and push notifications.

The problem with hosting all of these processes on the server side is that each request involving one of them has to travel all the way from the client to the server, every time. This introduces a great deal of latency. For this reason, contemporary applications run more code on the client side; one use case is rendering dynamic webpages in real time by running scripts within the browser that make changes to the content a user sees.

Like with 'frontend' and 'client-side,' backend is also a term for the processes that take place on the server, although backend only refers to the types of processes and server-side refers to the location where processes run.

Client-side scripting simply means running scripts, such as JavaScript, on the client device, usually within a browser. All kinds of scripts can run on the client side if they are written in JavaScript, because JavaScript is universally supported. Other scripting languages can only be used if the user's browser supports them

In web development, 'client side' refers to everything in a web application that is displayed or takes place on the client (end user device). This includes what the user sees, such as text, images, and the rest of the UI, along with any actions that an application performs within the user's browser.

Markup languages like HTML and CSS are interpreted by the browser on the client side. In addition, many contemporary developers are including client-side processes in their application architecture and moving away from doing everything on the server side; business logic for dynamic webpages, for instance, usually runs client side in a modern web application. Client-side processes are almost always written in JavaScript.

In the netflix.com example above, the HTML, CSS, and JavaScript that dictate how the Netflix main page appears to the user are interpreted by the browser on the client side. The page can also respond to 'events': For instance, if the user's mouse hovers over one of the movie thumbnail images, the image expands and adjacent thumbnails move slightly to one side to make room for the larger image. This is an example of a client-side process; the code within the webpage itself responds to the user's mouse and initiates this action without communicating with the server.

The client side is also known as the frontend, although these two terms do not mean precisely the same thing. Client-side refers solely to the location where processes run, while frontend refers to the kinds of processes that run client-side.

A dynamic webpage is a webpage that does not display the same content for all users and changes based on user input. The Facebook homepage is a dynamic page; the Facebook login page is for the most part static.

1.3 Sever side scripting Language

Web browsers communicate with web servers using the HyperText Transfer Protocol (HTTP). When you click a link on a web page, submit a form, or run a search, an HTTP request is sent from your browser to the target server.

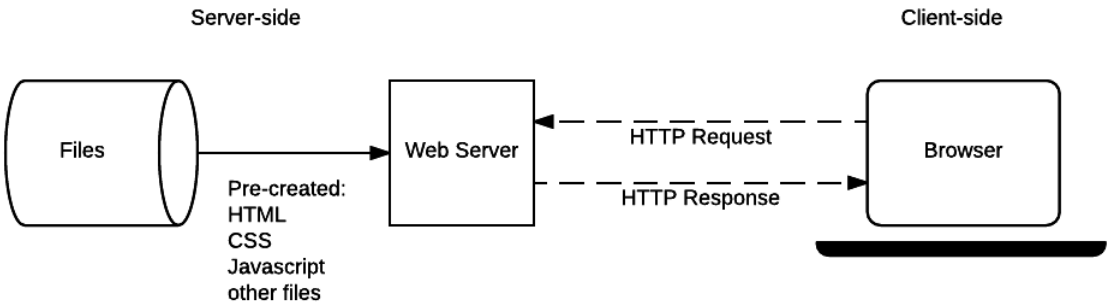
The request includes a URL identifying the affected resource, a method that defines the required action (for example to get, delete, or post the resource), and may include additional information encoded in URL parameters (the field-value pairs sent via a query string), as POST data (data sent by the HTTP POST method), or in associated cookies. Web servers wait for client request messages, process them when they arrive, and reply to the web browser with an HTTP response message. The response contains a status line indicating whether or not the request succeeded (e.g. "HTTP/1.1 200 OK" for success). In the simplest of terms, scripting is laying out a set of instructions to be carried out by a computer, for example, automating a spreadsheet so that it auto-calculates values when some of its cells are filled.

Scripting can be done client-side (e.g web browsers with JavaScript) or server-side. Typically, client-side scripting handles how the content can be viewed and manipulated by the user, while server-side scripting focuses on what content is delivered, how it's delivered, and how it's stored, among other things.

Some functions can be done by either side. For example, if a user has to input a number and they instead write a letter, the browser can check the input before it sends the data to the server (client-side verification) or it can be processed by the server which returns an error message (server-side verification).

The body of a successful response to a request would contain the requested resource (e.g. a new HTML page, or an image, etc...), which could then be displayed by the web browser.

The diagram below shows basic web server architecture for a *static site* (a static site is one that returns the same hard-coded content from the server whenever a particular resource is requested). When a user wants to navigate to a



page, the browser sends an HTTP "GET" request specifying its URL.

Figure 1.1 – Web server architecture

A dynamic website is one where some of the response content is generated *dynamically*, only when needed. On a dynamic website HTML pages are normally created by inserting data from a database into placeholders in HTML templates (this is a much more efficient way of storing large amounts of content than using static websites).

A dynamic site can return different data for a URL based on information provided by the user or stored preferences and can perform other operations as part of returning a response (e.g. sending notifications).

Most of the code to support a dynamic website must run on the server. Creating this code is known as "**server-side programming**" (or sometimes "**back-end scripting**").

The diagram below shows a simple architecture for a dynamic website. As in the previous diagram, browsers send HTTP requests to the server, then the server processes the requests and returns appropriate HTTP responses.

Server-side Uses

- It processes the user input
- Displays the requested pages
- Structure of web applications
- Interaction with servers/storages
- Interaction with databases
- Querying the database
- Encoding of data into HTML
- Operations over databases like delete, update.

The most popular Server side scripting language

PHP is by far the most used server-side scripting language. Just above 80% of websites are running on PHP. It was the first programming language designed specifically for the web, and that lead to its dominance in the Web 2.0 (blogging, content creation) era of the 2000s. Furthermore, Wordpress runs on it and powers 25% of the websites today, including most popular blogs and news websites.
There are several languages that can be used for server-side programming:

- PHP
- ASP.NET (C# OR Visual Basic)

- C++
- Java and JSP
- Python
- Ruby on Rails and so on.

Server-side Example

Figure 1.2

```
1. // This is a sample C# code.
2. using System;
3. // namespace
4. class ServerSide
5. {
6.     public static void Main()
7.     {
8.         System.Console.WriteLine(“Hello C# Corner”);
9.         // printing a line
10.       }
11.   }
12.
```

Much like with client side, 'server side' means everything that happens on the server, instead of on the client. In the past, nearly all business logic ran on the server side, and this included rendering dynamic webpages, interacting with databases, identity authentication, and push notifications.

The problem with hosting all of these processes on the server side is that each request involving one of them has to travel all the way from the client to the server, every time.

This introduces a great deal of latency. For this reason, contemporary applications run more code on the client side; one use case is rendering dynamic webpages in real time by running scripts within the browser that make changes to the content a user sees.

Like with 'frontend' and 'client-side,' backend is also a term for the processes that take place on the server, although backend only refers to the types of processes and server-side refers to the location where processes run.

In serverless computing, all server-side or backend processes still run on servers instead of client devices, but they are not deployed on any specific server or set of servers. Backend processes are broken up into functions, which run on demand, and scale up automatically. Developers can still build all the functionality that normally runs server-side within a serverless architecture.

1.3 Statement of Problem

The Website will have Logo, Header, Navigation Bar, Body and Footer. This format applies to the four sections of the webpage page. The four sections of the webpage are: Home, About, Our Story and Contact. On the home section, there will be a picture with hyperlink “Why travel abroad?” Once the user clicks on that link, it takes the user to the About section and the user can read the benefits of traveling abroad. On the About section, the user would read about the purpose of the website.

On Our Story section, that is where user can read how this website all began and stats of young people traveling the world. One the home section, there will be an image slider showing some of our travelers’ adventures. One the Contact section, there will be the location, email address and contact form of Students Xplore website. The form on the Contact page will be validated using JavaScript. CSS and JavaScript files will be external. Those interested in sharing their traveling experiences can reach to us via the contact form. The Administrator will reach out to them in order to get their photos and stories so that it would be published on the website. Besides, users can ask questions using the contact form and our administrator would get in touch as soon as possible. The user’s contact information would be stored in a database. We would use phpmyAdmin to achieve this.

2 SELECTION OF PROGRAMMING INSTRUMENTS

Bootstrap is a web framework that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

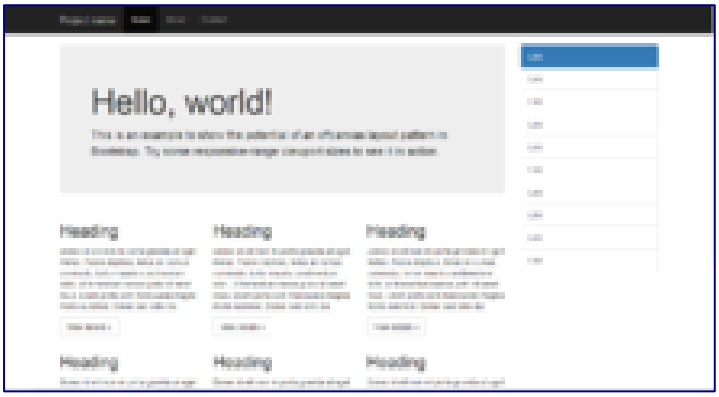
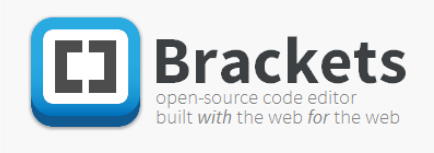


Figure 1.2 – Example of a webpage using Bootstrap framework

The most prominent components of Bootstrap are its layout components, as they affect an entire web page. The basic layout component is called "Container", as every other element in the page is placed in it. Developers can choose between a fixed-width container and a fluid-width container. While the latter always fills the width of the web page, the former uses one of the four predefined fixed widths, depending on the size of the screen showing the page:

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.



Brackets is a source code editor with a primary focus on web development. Created by Adobe Systems, it is free and open-source software licensed under the MIT License, and is currently maintained on GitHub by Adobe and other open-source developers. It is written in JavaScript, HTML and CSS

JavaScript, also known as JS, is a programming language that follows the ECMAScript standard. JavaScript is a multi-paradigm, high-level programming language that is often compiled just-in-time. Curly-bracket syntax, dynamic typing, prototype-based object orientation, and first-class functions are all features of this language.

Cascading Style Sheets (CSS) is a term for defining the appearance of a document written in a markup language like HTML. Along with HTML and JavaScript, CSS is a key component of the World Wide Web.

3 WEBSITE REALIZATION

3.1 Modeling of website

Figure 3.1 – Protype of the home section of the website

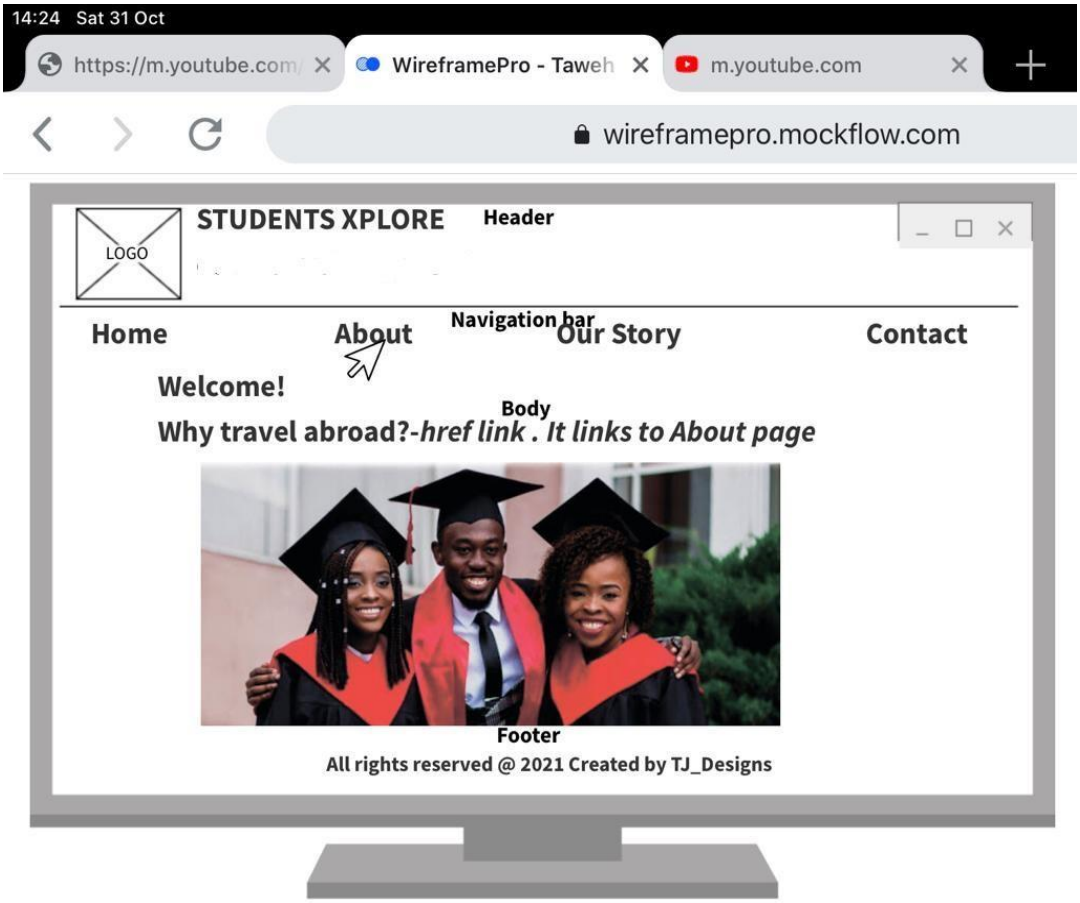


Figure 3.2 – The prototype of About Section of the website

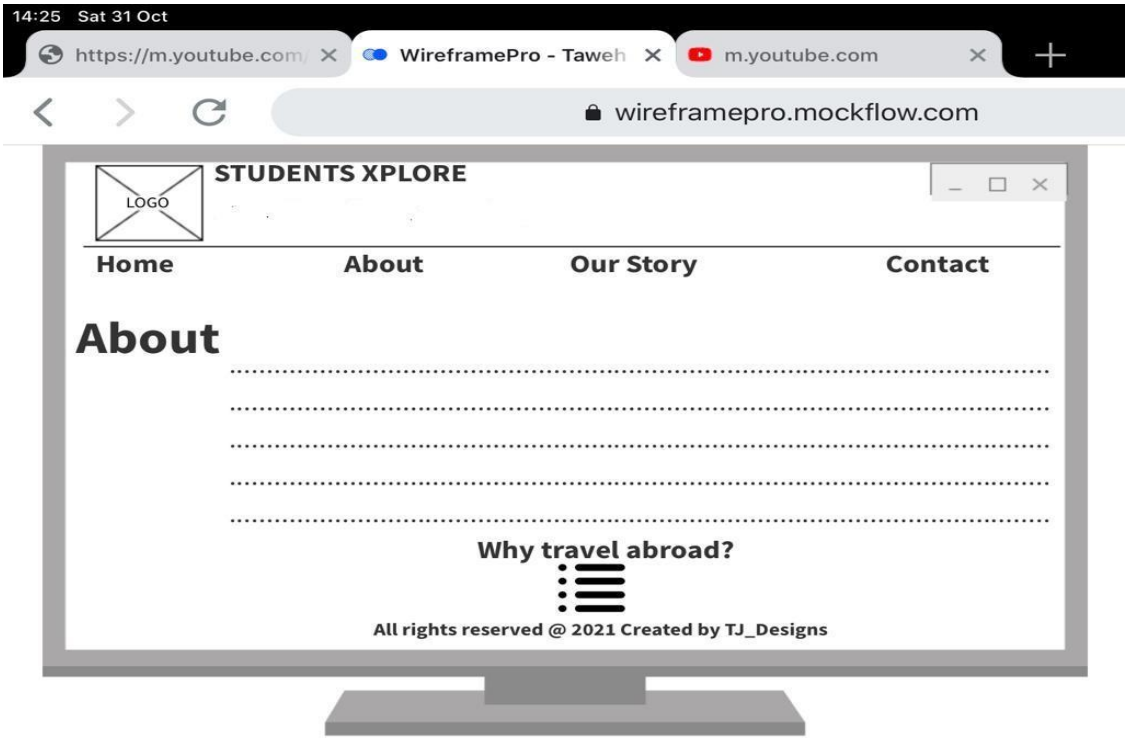


Figure 3.3– Prototype of the our story section of the website

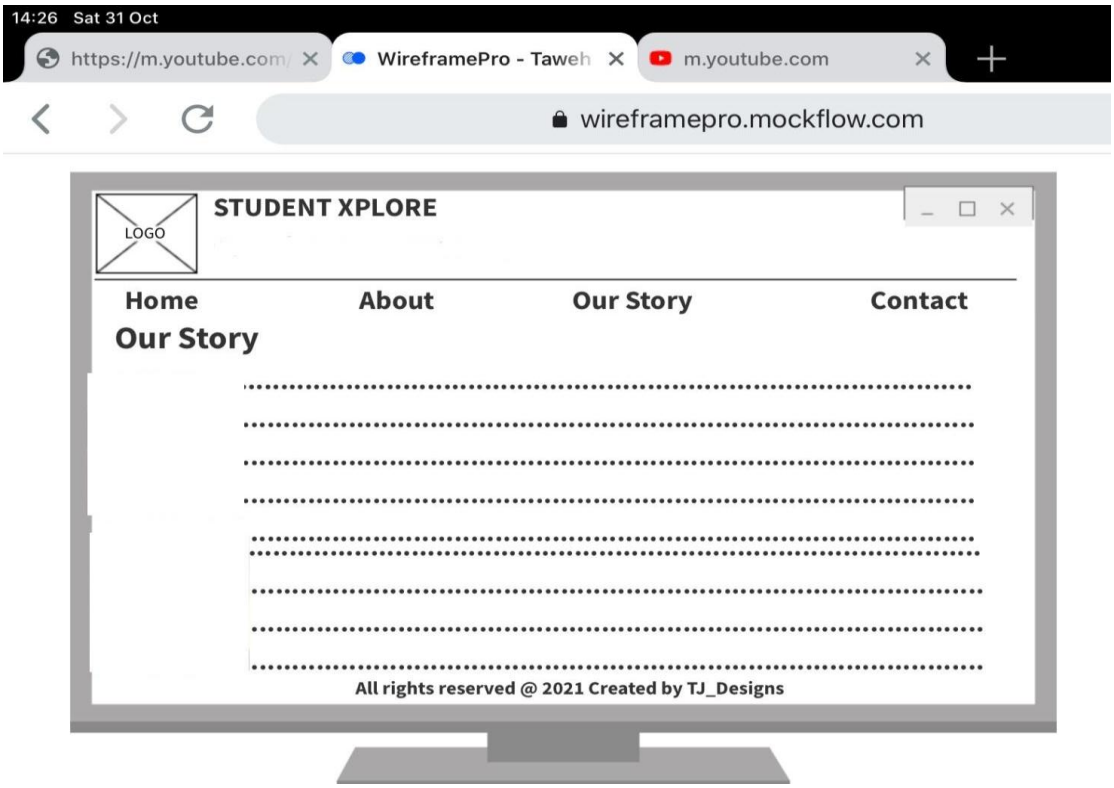


Figure 3.4 – Prototype of the contact section of the website

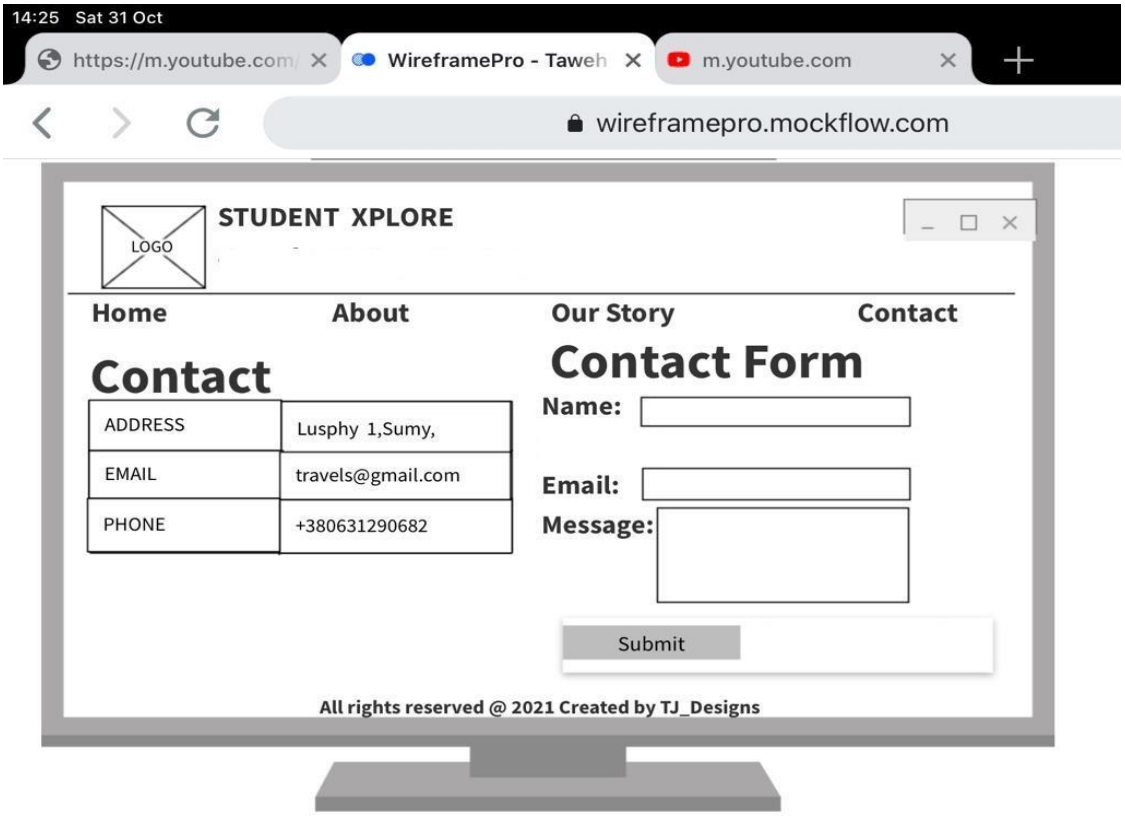


Figure 3.5 – Sample of backend of the website

Website prototype back-end

User's input on the contact page will be stored in a database using phpmyAdmin

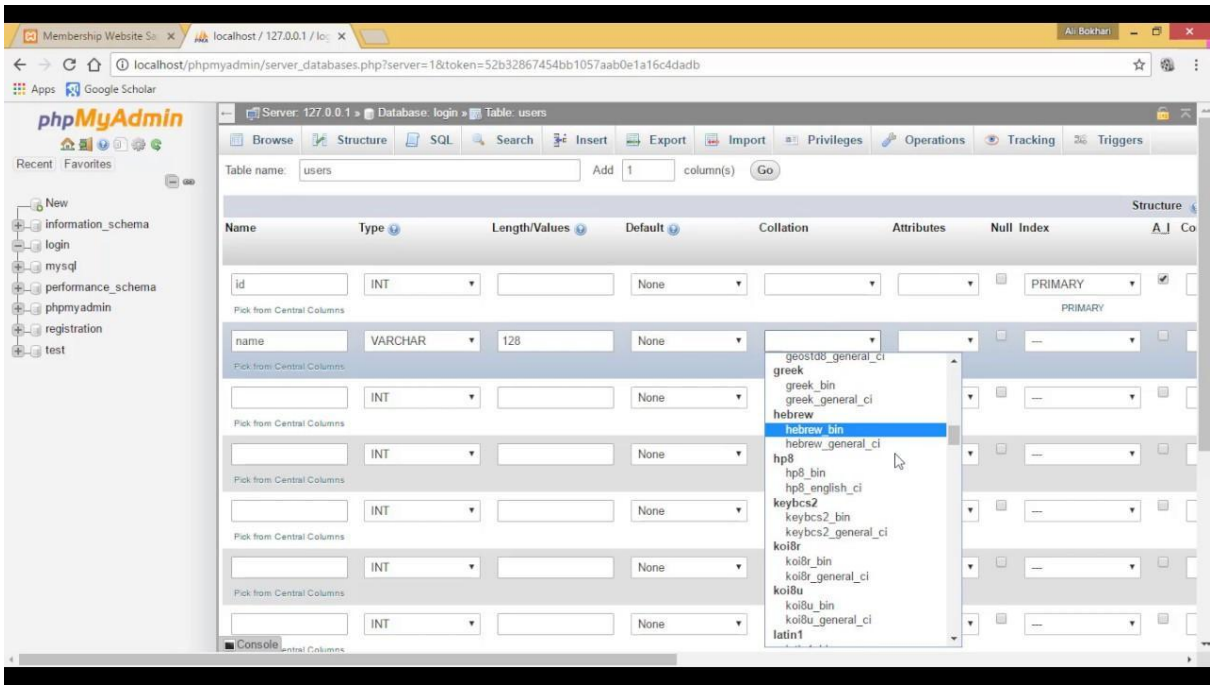


Figure 3.6 – Home section Students' Xplore website

3.2 Website realization

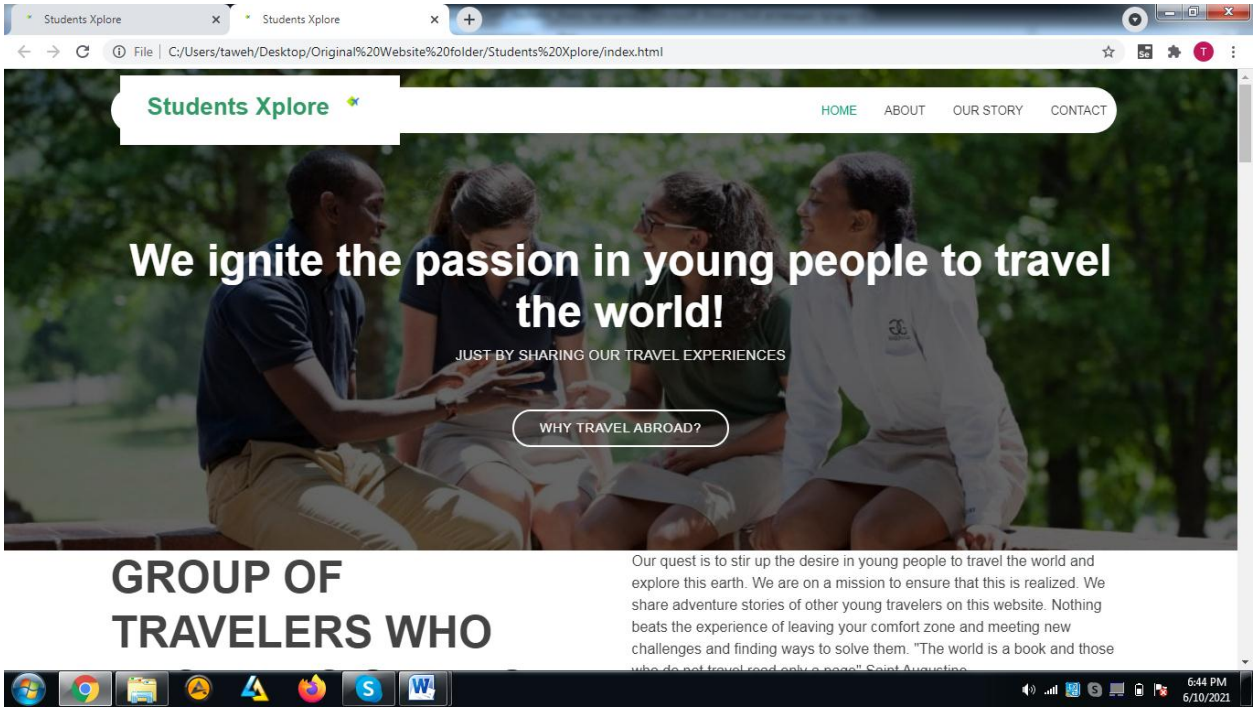


Figure 3.7 – About section of Students' Xplore website

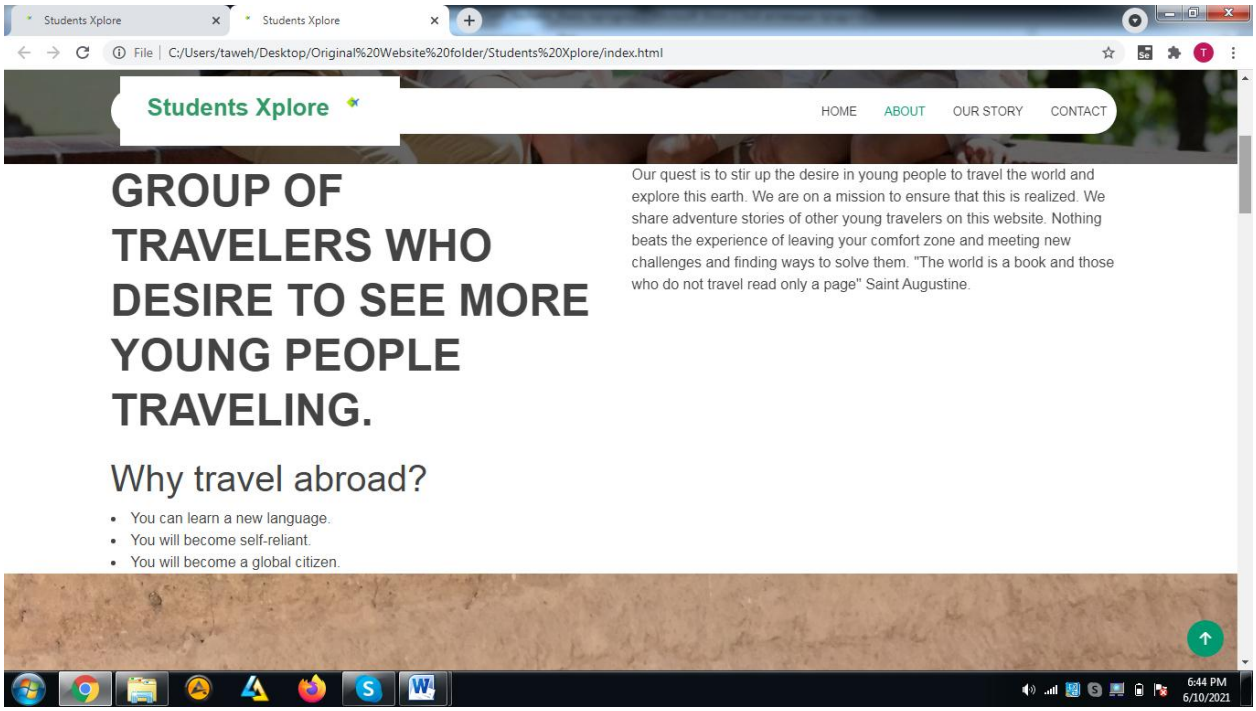


Figure 3.8 – Our Story section of Student's Xplore website

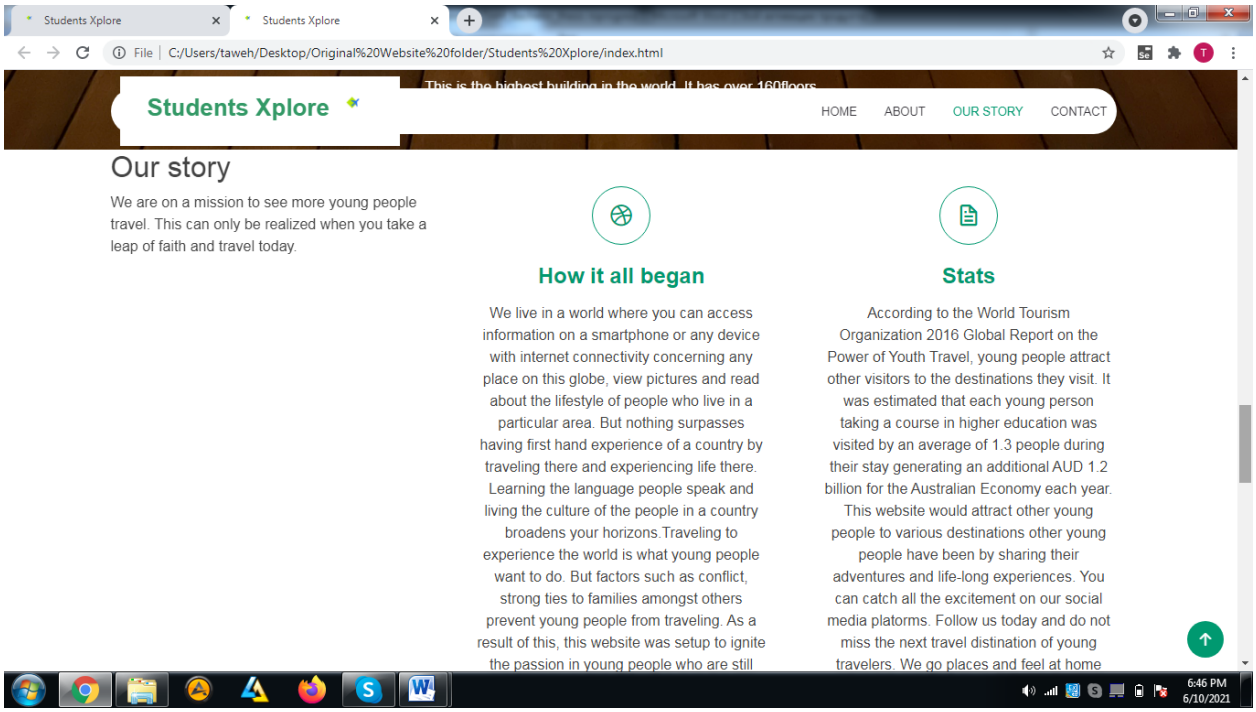


Figure 3.9 –Contact section of Students' Xplore website

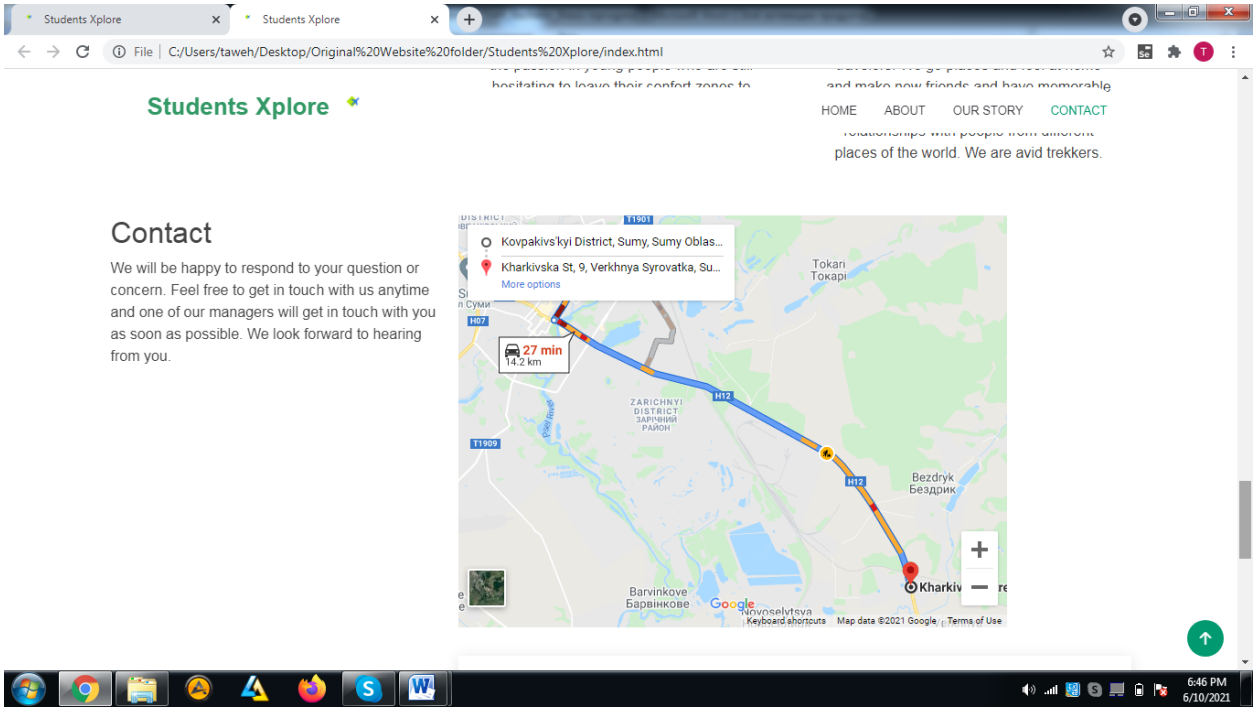


Figure 3.10 Contact form of Student's Xplore website

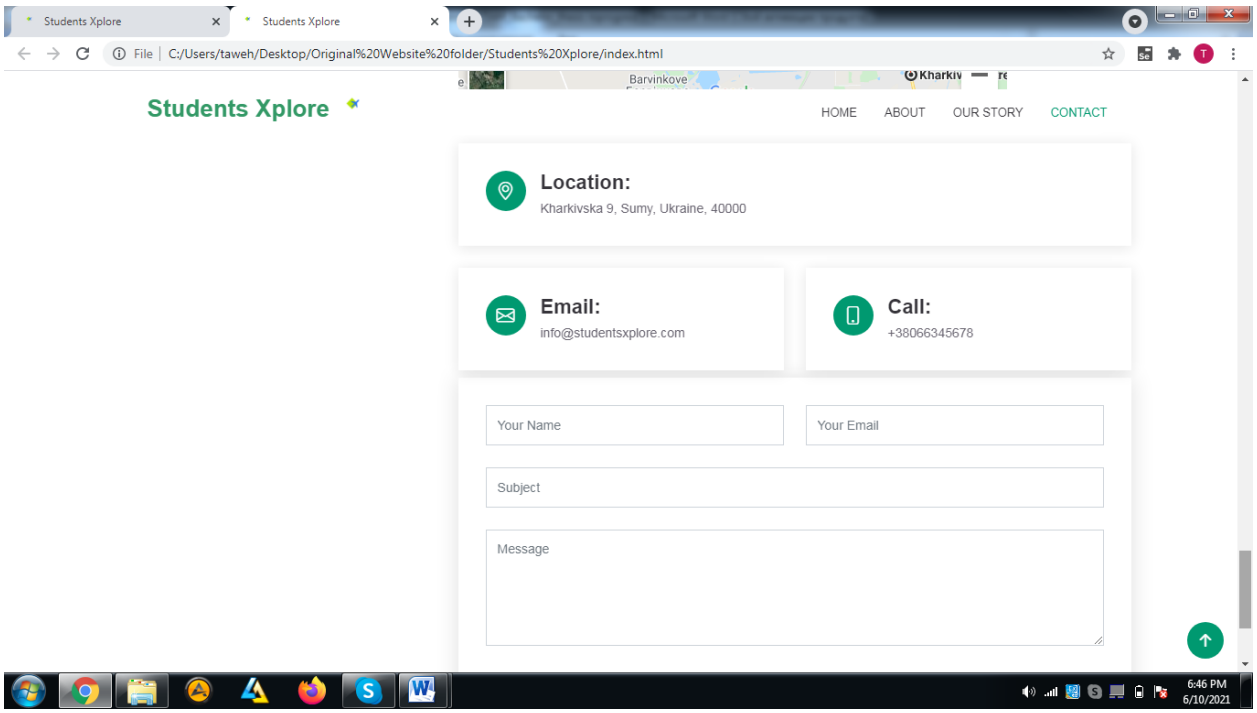


Figure 3.11 – The footer of Student's Xplore website

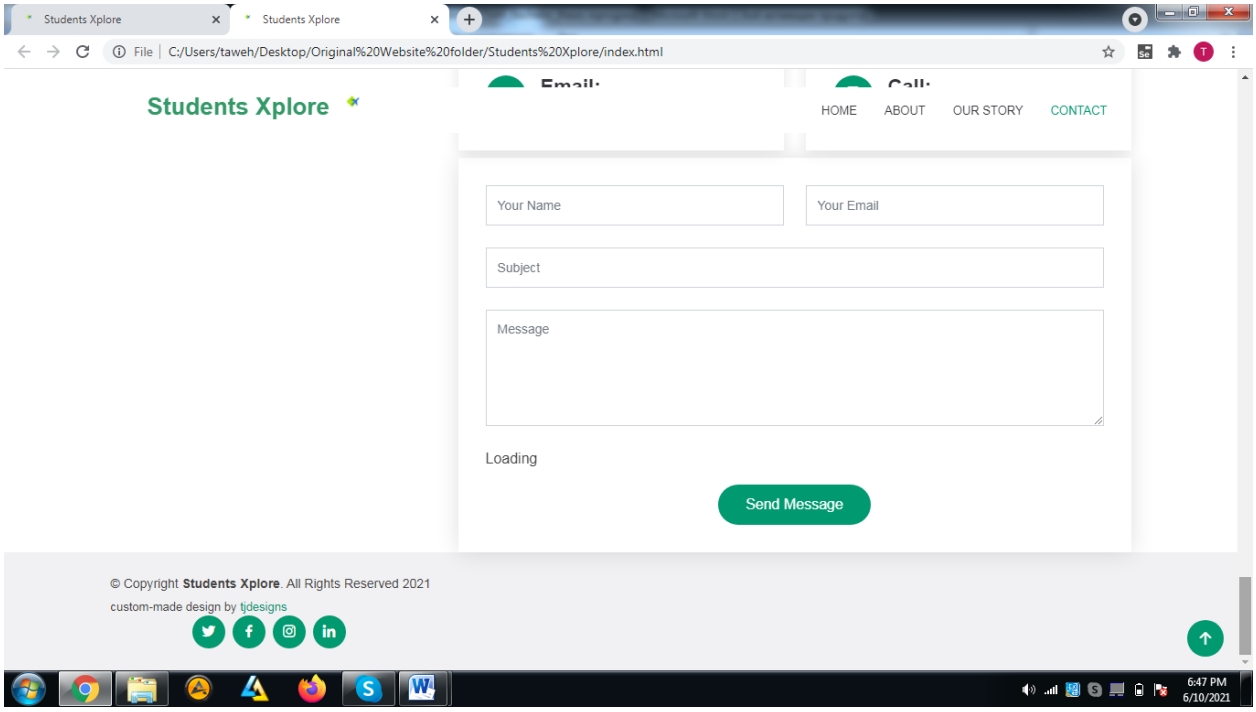


Figure 3.12 – SQL of PhpMyadmin for Students' Xplore website

The Backend of the website

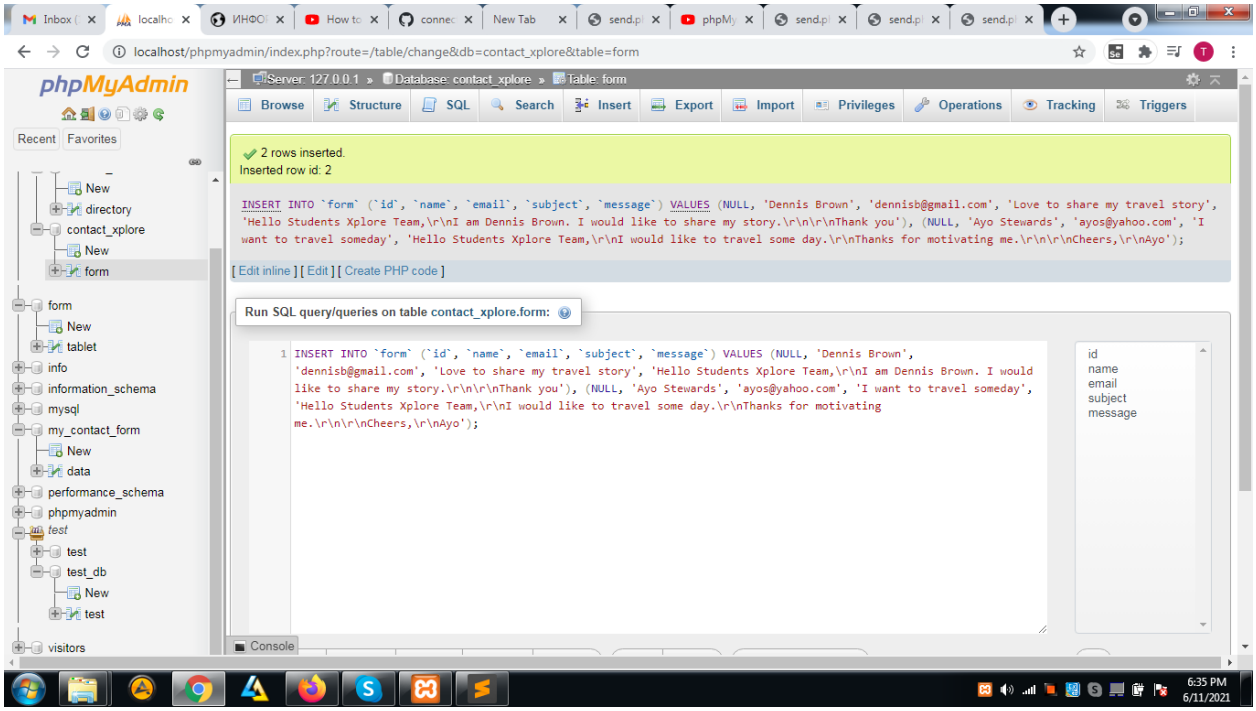
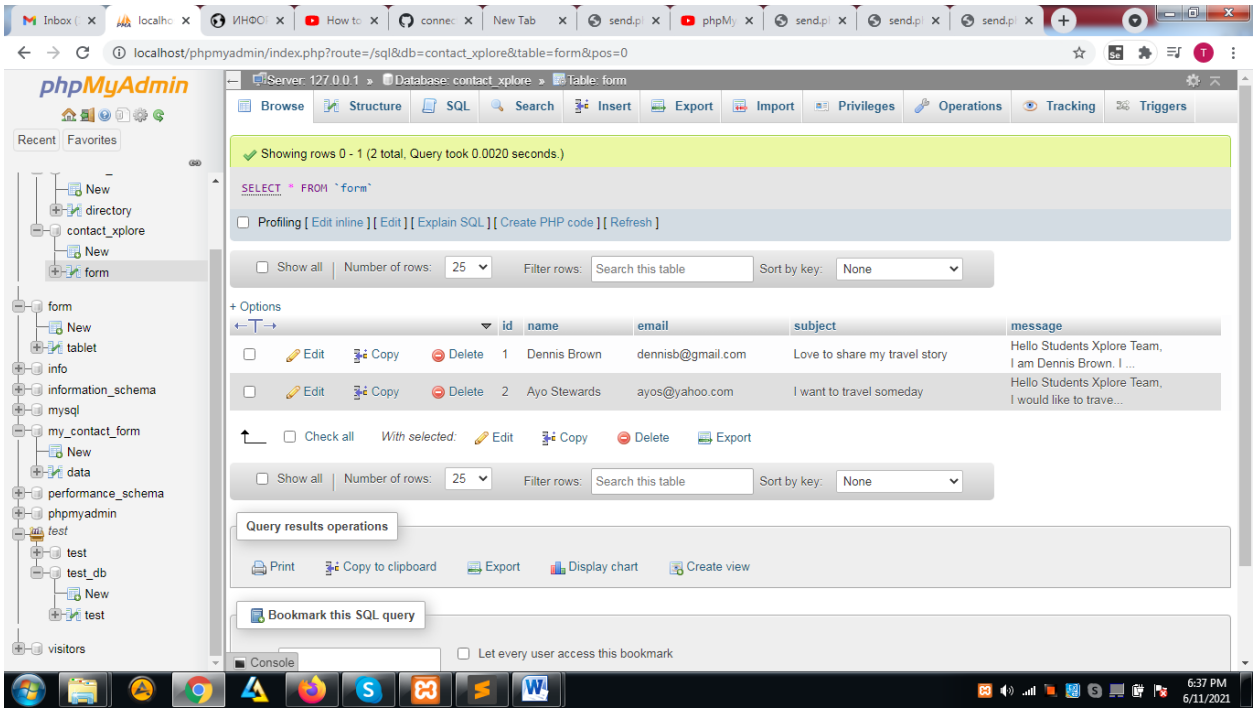


Figure 3.13 – Contact form Table of PhpMyadmin for Students' Xplore website



3.3 Website HTML code

Snippet of the HTML Code and href links and script links

<!DOCTYPE html>

```
<html lang="en">

<head>

This is the head of Students Xplore website.
It contains the title and makes the website responsive
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">

<title>Students Xplore</title>
<meta content="" name="description">
<meta content="" name="keywords">

Here contains the favicon of Students Xplore website.
There is an href for the logo of the website
<!-- Favicons -->
<link href="assets/img/xplorelogo.png" rel="icon">
<link href="assets/img/xplorelogo.png" rel="xplore-icon">

Here contains the Cascading style sheet(CSS) files.
We used the CSS to make the website look attractive.
<!-- CSS Files -->
<link href="assets/features/aos/aos.css" rel="stylesheet">
<link href="assets/features/bootstrap/css/bootstrap.min.css" rel="stylesheet">
<link href="assets/features/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
<link href="assets/features/boxicons/css/boxicons.min.css" rel="stylesheet">
<link href="assets/features/glightbox/css/glightbox.min.css" rel="stylesheet">

Here contains the main Cascading Style sheet file
<!-- Template Main CSS File -->
<link href="assets/css/style.css" rel="stylesheet">

Here contains the Main JavaScript file and other JavaScript files.
We used Javascript to make the website lively and
it is also used to validate fields on the contact form.
<!—Template Main JS File →
<script src=”assets/js/main.js”></script>
<!—Features JS Files →
<script src=”assets/features/aos/aos.js”></script>
<script src=”assets/features/glightbox/js/glightbox.min.js”></script>
<script src=”assets/features/bootstrap/js/bootstrap.bundle.min.js”></script>
```

<script src=”assets/features/php-email-form/validate.js”></script>

<!—close of head→
</head>

<!—Body→

<body>
Here contains the body of the website.
Here you can find the home, about, our story
and contact sections of the website.
<!-- ======= Header ======= -->
<header id="header" class="fixed-top d-flex align-items-center">

<h1 class="text-light">Students Xplore <img src="assets/img/xplorelogo.png" height="40" width="40" alt="students xplore logo"></h1>
Here contains the image slider.
This slider has images of our student travelers and their adventures.
<!--
Carousel-->

Here
contains the
image slider

<button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
<button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="1" aria-label="Slide 2"></button>
<button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="2" aria-label="Slide 3"></button>

<img src="assets/img/sliderimg-one.jpg" class="d-block w-100" alt="Slider image one">

<h5>At the top of Burj Khalifa in Dubai</h5>
This is the highest building in the world. It has over 160floors

```
<img src="assets/img/sliderimg-two.jpg" class="d-block w-100" alt="Slider image two">
<div class="carousel-caption d-none d-md-block">
<h5>Dubai Frame</h5>
<p>This is the largest frame in the world</p>
</div>
</div>
<div class="carousel-item">
<img src="assets/img/sliderimg-three.jpg" class="d-block w-100" alt="slider image three">
<div class="carousel-caption d-none d-md-block">
<h5>Bur Dubai- Old Dubai</h5>
<p>Old Dubai was a fishing town.</p>
</div>
</div>
</div>
<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide="prev">
<span class="carousel-control-prev-icon" aria-hidden="true"></span>
<span class="visually-hidden">Previous</span>
</button>
<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide="next">
<span class="carousel-control-next-icon" aria-hidden="true"></span>
<span class="visually-hidden">Next</span>
</button>
</div>
<!--Carousel End-->

    Here is the footer of the website.
    It contains social media icons of the website.
    </div>
<div class="container d-
md-flex py-4">
                <div class="me-md-auto text-center text-md-start">
                <div class="copyright">
                &copy; Copyright <strong><span>Students Xplore</span></strong>. All Rights Reserved 2021
                <div class="credits">
                custom-made design by <a href="www.bootstrap.org">tjdesigns</a>
                </div>
                </div>
                <div class="social-links text-center text-md-right pt-3 pt-md-0">
```

```
                    <a href="https://twitter.com" class="twitter"\ target="_blank"><i
                    class="bx bxl-twitter"></i></a>
                    <a href="https://www.facebook.com"\ target="_blank"
                    class="facebook"><i class="bx bxl-facebook"></i></a>
                    <a href="https://www.instagram.com" class="instagram"\
                    target="_blank"><i class="bx bxl-instagram"></i></a>
                    <a href="https://www.linkedin.com" class="linkedin"\
                    target="_blank"><i class="bx bxl-linkedin"></i></a>
                    </div>
                    </div>
                    </footer><!-- End Footer -->
                    <a href="#" class="back-to-top d-flex align-items-center justify-
                    content-center"><i class="bi bi-arrow-up-short"></i></a>

</body>
```

Conclusion

The website would serve as a motivational source for many young people to travel the world. I do believe that young people will take action immediately to explore this world. The time put into this project worth its intended outcome. Travel today!

The Website has Logo, Header, Navigation Bar, Body and Footer. This format applies to the four sections of the webpage page. The four sections of the webpage are: Home, About, Our Story and Contact.

On the home section, there is a picture with hyperlink “Why travel abroad?” Once the user clicks on that link, it takes the user to the About section and the user can read the benefits of traveling abroad. On the About section, the user would read about the purpose of the website.

On Our Story section, that is where user can read how this website all began and stats of young people traveling the world. One the home section, there is an image slider showing some of our travelers’ adventures. One the Contact section, the user can find the location, email address and contact form of Students Xplore website.

The form on the Contact page is validated using JavaScript. CSS and JavaScript files are external. Those interested in sharing their traveling experiences can reach to us via the contact form.

The Administrator can reach out to them in order to get their photos and stories so that it would be published on the website. Besides, users can ask questions using the contact form and our administrator would get in touch as soon as possible. The user’s contact information would be stored in a database. We would use phpmyAdmin to achieve this.

References

1. Clientside scripting language https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/

2. Server side scripting- https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

3.PHP- https://twm.me/best-programming-languages-and-frameworks-for-server-side-web-development/ https://www.phpmyadmin.net/

4. PhpMyAdmin-https://www.phpmyadmin.net/

5.Clientside programming-https://www.c-sharpcorner.com/UploadFile/2072a9/client-side-vs-server-side-programming-languages/

6.Server side programming-https://www.c-sharpcorner.com/UploadFile/2072a9/client-side-vs-server-side-programming-languages/

7.Scripting.Languages-https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/

8. Bootstrap-https://getbootstrap.com/

9. JavaScript-https://developer.mozilla.org/en-US/docs/Web/JavaScript

10. Brackets-http://brackets.io/

11. Cascading Style sheets-https://www.w3schools.com/css/

12. Web Server Architecture- https://www.techopedia.com/definition/30263/web-server-architecture

13. Web Development roadmaps- https://www.w3schools.com/whatis/

14.Website architecture- https://terakeet.com/blog/website-architecture/

15. Website prototyping- https://www.experienceux.co.uk/faqs/what-is-a-website-prototype/

ADDITION