Zoom Travel

Final Report

Website Link: http://zoomtravel.us-east-2.elasticbeanstalk.com/

Team #13

Members:

Arpita Deshmukh

Chiou-Jiin Huang

Shilpa Chanshetti

S. Sneha Bhat

Contents

PROJECT PROPOSAL AND USER PERSONA

ISTM 631 – Team 13

Zoom Travel

Zoom Travel (ZT) is an American airline company headquartered in Houston, Texas. This airline company was established in 1989. Zoom Travel extended its services from Domestic to International in the year 2001. At present, the airline has about 50,000 employees.

Zoom wants to create a better user web experience for all its travelers. Crystal Watson, the Chief Innovation Officer at Zoom has hired our team to build a website (Zoom Travel) that would guide the travelers throughout their flight journey. She wants the website to be highly responsive specially designed for their travelers so that they would not face any problem during their journey. Moreover, she expects a web application that supports desktop as well as mobile phone users.

Key Business Processes

Currently, Zoom manages a simple database of their flight information. With the series of feedback on in-airport issues faced by travelers, ZT team has decided to have a website with the following operations:

Traveler's Records

When a person is traveling with Zoom, they are required to sign up with the Zoom Travel for their travel. A person should provide their personal information such as Name, age, any disabilities, travel information questionnaire (ex: have they traveled before, visa status if traveling international), etc. These records will be stored in the database for customizing their in-airport travel services.

Traveler's Travel Information

For seamless navigation for travelers on the day of their travel, ZT is required to store the travel information of all their upcoming travels. Once in the system, a person can provide their travel details like International or Domestic travel, Airline, PNR, Departure Airport and Arrival Airport, Departure and Arrival Date, No. of check-in bags, No. of carry-on bags. The information provided by the traveler is then recorded in the database.

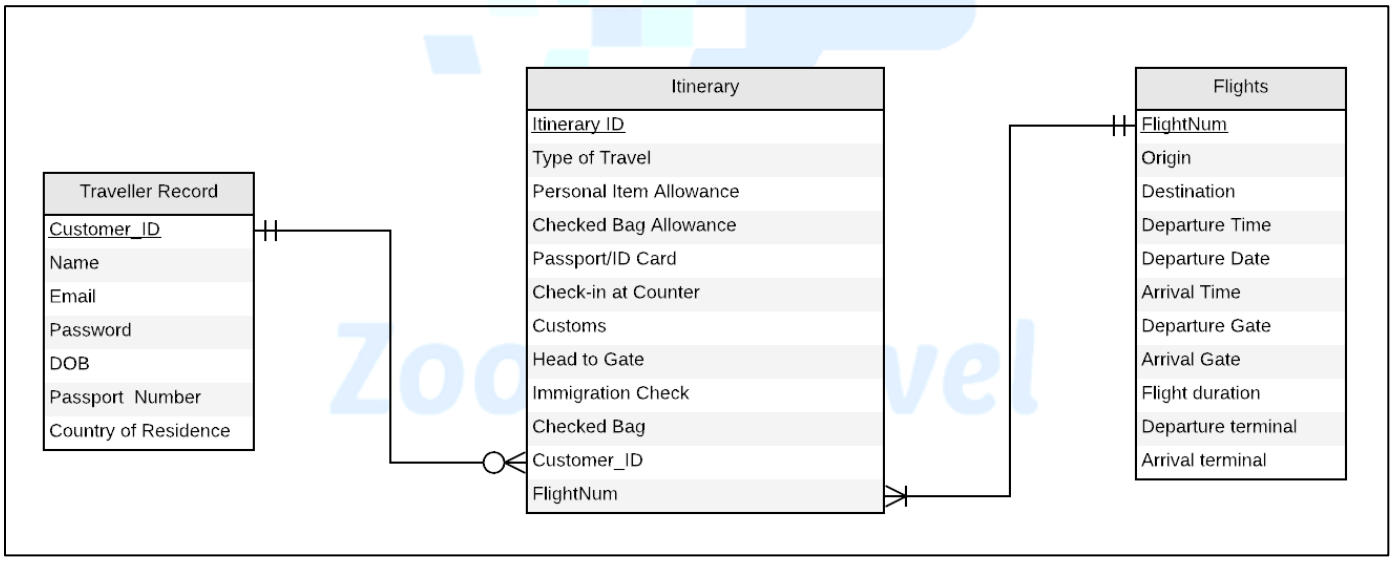
Travelers sometimes make some changes to their travel plans. Any cancellations of travels, the record will be deleted from the main table and would be stored in another table. Any updates on the travel, necessary changes will be reflected in the main database table.

Travel Checklist service

ZT's main motive is to provide seamless navigation on the day of the flight. This website will collect the required details of the flight when he/she logs in and provide a check-list of items on the day of travel. Travelers will be provided information such as check-in information, Flight details, regulations on bags and baggage weights, immigration information, boarding information, baggage claim, arrival city weather details.

Database Structure

For this application, we have decided to work with a SQL database to store the required data. The data for this database will be static for this project. To maintain the travel records, we would store in Traveler table. This would include all information that a traveler would provide from the website. The Traveler's information is also stored in the traveler record table. This table stores all the travel-related information. The itinerary would be another table that updates after each step of the checklist is completed. We would also require flight information from the airline which will be stored in the flights table. ERD displaying the data is shown below.



Entity Relationship Diagram

Expected Outcomes

Using the webserver, database server, and languages of our choice to design and build a website that (at minimum) will meet the following requirements:

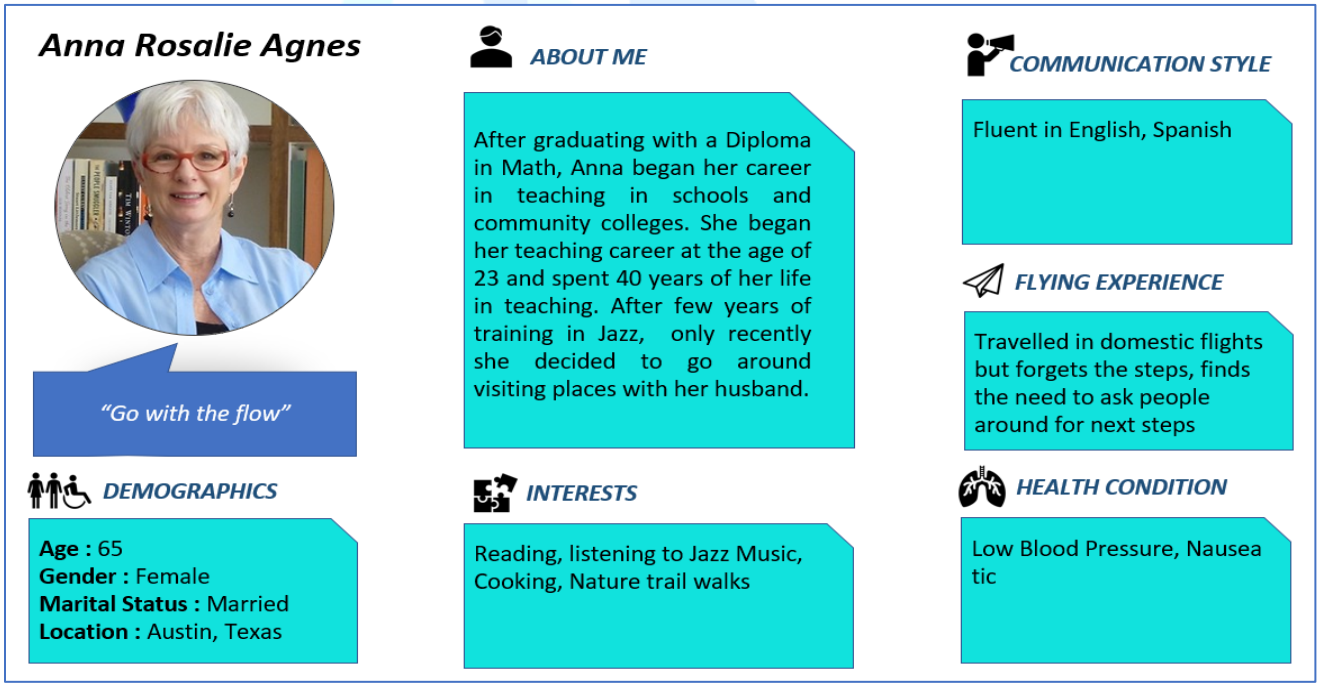
Traveler Records: New users must be able to create an account by filling up the sign-up sheet. The website allows travelers to add/update/delete their personal information to view their information later associated with their travel.

Traveler's Travel Information: The traveler would have entered basic travel information to get their travel updates displayed on the website.

Website displaying the checklist: All the steps at the traveler have to follow throughout his/her journey will be displayed on the user account in the format of a checklist. The traveler would be able to view checked as well as unchecked steps. The website will also display sub-steps wherein the traveler can view detailed steps, helping the traveler on every step of their journey.

User Personas

Persona 1: Elderly Travelers



Persona 2: First - Time Travelers

Keerthan Shetty

"There is no substitute for hard work"

ABOUT ME

Completed undergrad in India. Now moving to the United States to pursue master's in Management Information Systems at Texas A&M University. This would be my first international trip and I am very excited for this.

COMMUNICATION STYLE

Fluent in English

FLYING EXPERIENCE

First Time International Traveller, Has travelled in Indian (domestic flights)

DEMOGRAPHICS

Age : 23
Gender : Male
Marital Status : Single
Location : Bangalore, India

INTERESTS

Coding and Playing Soccer

HEALTH CONDITION

Feels nervous in long flights

Persona 3: Non - frequent travelers

Amy Watson McDonald

"I'm juggling so many aspects of my family – traveling should not be as hectic too"

ABOUT ME

After graduating from college, Amy got married and stayed home. She is a homemaker and takes care of her twin daughters. Everyday is a battle for Amy and whenever she has time, she loves to travel with her husband and children.

COMMUNICATION STYLE

Fluent in spanish, Beginner in English

FLYING EXPERIENCE

Has flown a couple of times within the country, tends to get confused with the procedures

DEMOGRAPHICS

Age : 31
Gender : Female
Marital Status : Married
Location : San Antonio, TX

INTERESTS

Movies, Brunches with friends, Traveling

HEALTH CONDITION

Nervous Flier

How the Project Will Be Conducted

Team Meeting

The team meeting would take place once a week. If required, we might schedule more meetings as per everybody's convenience.

Project Deliverables

Week of	***Tasks***
Jan 20	Finish/Submit Team Contract
Jan 20	Search for project Ideas
Jan 27	Finalize project idea and work on Project Proposal
Jan 30	Project Proposal Submission
Feb 3	Discussion on User Personas
Feb 10	Work on User Personas Report
Feb 17	Start working on the Web Design
Mar 2	Finish front-end of the application
Mar 16	Start working on Backend
Mar 30	Finish backend of the application
Apr 27	Project Due Date

Learning Expectations

1. Recognizing the importance of the interface design towards the success of the system
2. How to make a website functional, usable as well as understandable
3. Combine different elements of design to enhance the overall user experience
4. Understand how business requirements can be incorporated into the project

DESIGN PERSONA

Overview

*This is **Flappy**, the face of Zoom Travel. Flappy believes in travelers having experience the same as his name, which is, Fly Happily. Its cartoony face helps nervous fliers feel at ease and make their flying experience seamless. Flappy's kind face portrays to the passengers that he is here to help.*

Flappy's playful smile makes the users feel welcoming and imbibe in them the feeling of being in good hands. He calms down the anxious travelers by throwing funny quotes & traveling tips. Flappy ensures that the passengers have access to all the steps & guidelines during their entire flight journey. Also, Flappy loves to do two things: to communicate with the users and to fly. His conversations with users comprise words related to flying.

Flappy is a fun character but is aware of his priorities and knows when it's time to get back to work.

Brand Traits

Unique but not complicated

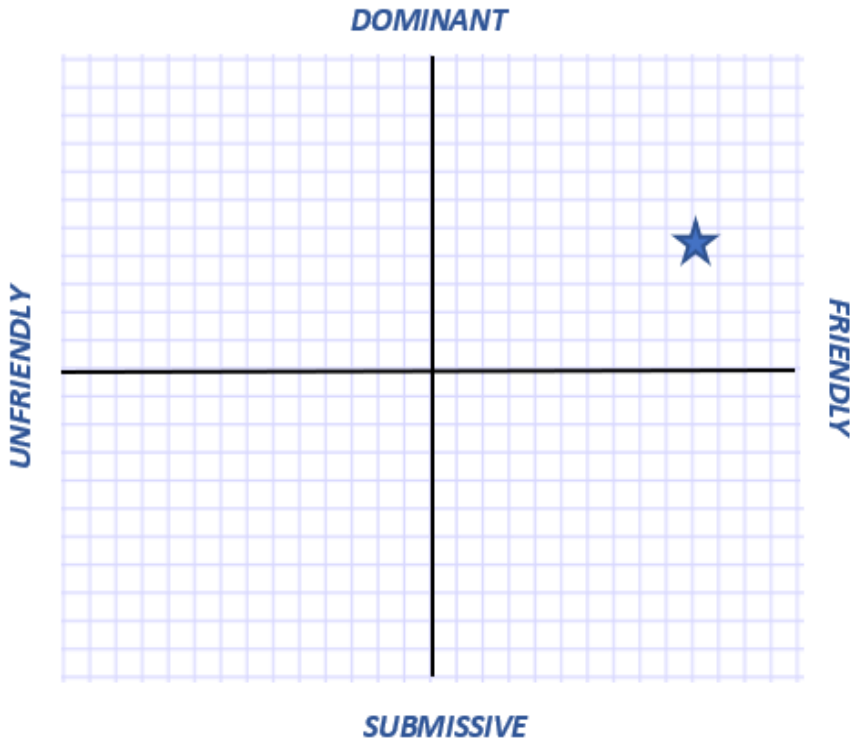
Seamless but not repetitive

Fun but not distracted

Sophisticated but not pretentious

Informal but not childish

Personality Map



Voice

The voice of Zoom Travel is our **Flappy**. He has a cute personality that shines through his words. Flappy helps people navigate through the air travel procedure and rules more easily. Nervous fliers will find his words appealing as he traverses through the steps or checklist which makes their travel pleasant and delightful.

Our main motive is to ease the traveling issues faced by the customers. It is overwhelming for first-time travelers or nervous fliers when they see a long list of steps. Flappy loves to talk and people feel connected hearing his adorable words. He incorporates various airplane terminologies like bad weather, turbulence, etc while speaking to the customers. Flappy celebrates with people on accomplishing the milestones and this helps people ease their nervousness.

Some of the examples are below:

General Message

Welcome Message

Voice contd.

Error Feedback

Critical Failure

Success Message

Marketing Copy

Visual Lexicon

Color

Blue (#4DA0FF)

The blue color is the central theme of the website. It is the color of the sky and sea. As our website is related to all the activities before sky travel, blue color resonates with it. It symbolizes trust, loyalty, and builds confidence, which we want to instill in our customer's minds as a long-lasting impression.

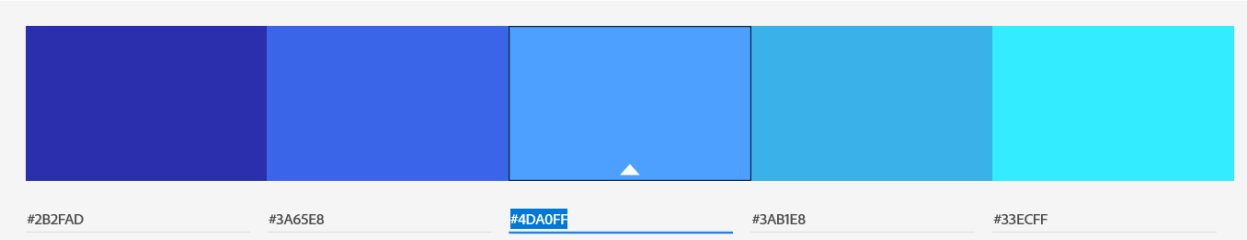


Image depicting blue color for the website

Orange (#FF9F31)

Of all the available colors, we chose orange because it creates a beautiful visual contrast to the central theme. This bright color conveys fun and humor.

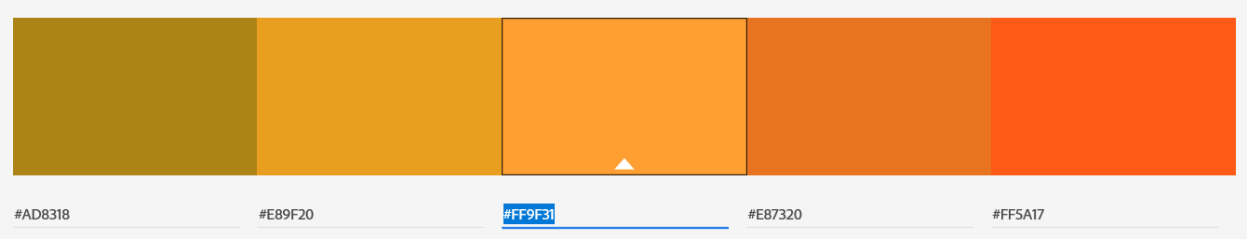


Image depicting orange color for the website

Black (#000000)

Black color would be a perfect color for the written content. This will be a great contrast to the light background of the website. This color makes the website elegant, formal yet friendly.

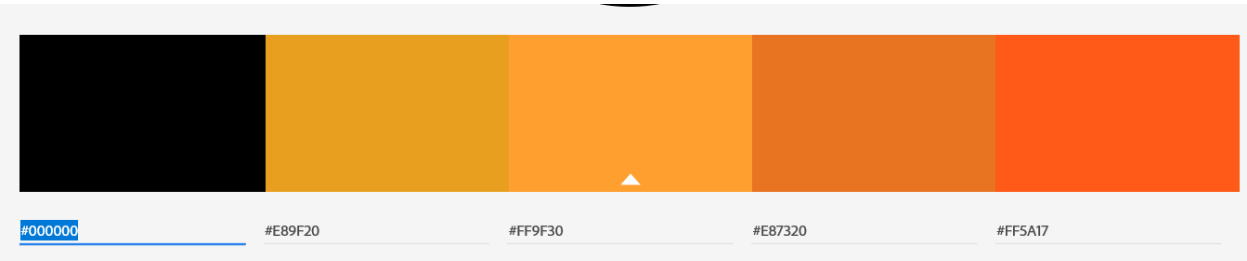
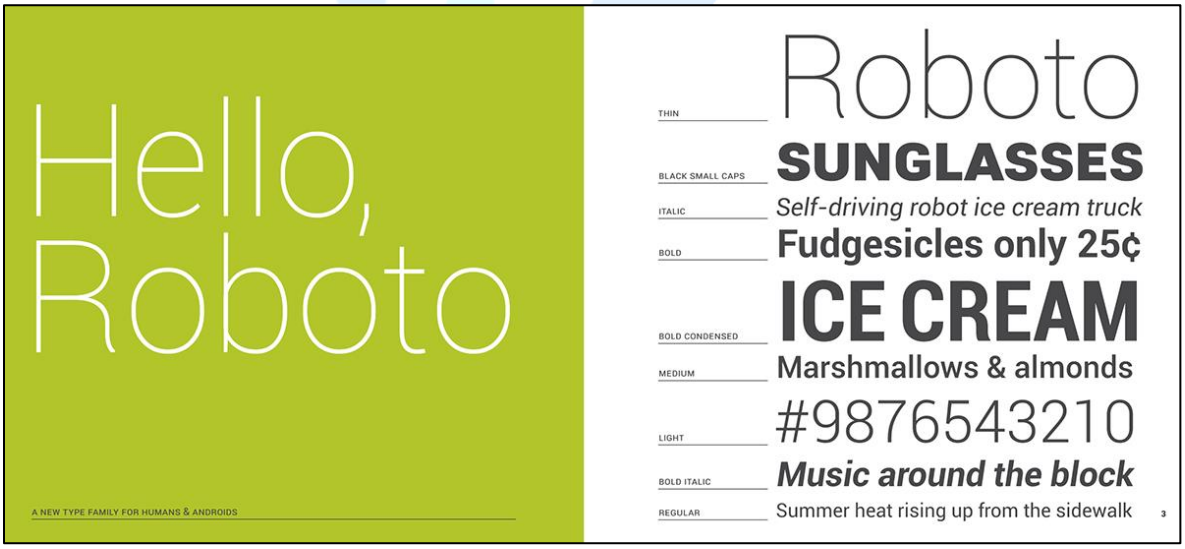


Image depicting orange color for the website

Typography

Zoom travels website is going to be designed in a way that is easy to read, understand, and visually appealing and engaging for the customers. To achieve this, typography plays an important role. To enhance the usability of the website and user experience, we will be using Roboto font of sans-serif typeface family for the content which will make it look stylish, exceptional, and easy to read.



Roboto Font

Proper sized and scaled sans-serif fonts will be used for the headings and labels which will attract the user's attention. A combination of capital letters for some part of the text will be used to make the site look professional and trustworthy. The color for the text will be contrasting or complementary to the background to make sure it's visible and readable.

General Style Notes

We will have different shades of the basic colors to create more contracts or emphasize more on a few important aspects of the website. We will include subtle textures in blank spaces so that it will not look very empty. The website will not be covered with too much text and images to make it look simple and attractive.

Flappy will not be used very extensively and will never get in the way. It might be used on some pages with different expressions as per the requirement.

Engagement Methods

Surprise & Delight

Random background options for the checklist that the users can choose from each time they log in.

Anticipation

Flappy will offer some traveling tips to the customer. Every time something is check-off the list, words of encouragement pop up.

Baby-Face Bias

The users will find Flappy's round features, big eyes, and short chin a sign of friendliness and honesty. The baby face bias helps the users reduce the anxiety of traveling.

Rewards

The app will run promotional offers and deals for the stores at the airports.

DESIGN MOCKUP

Overview

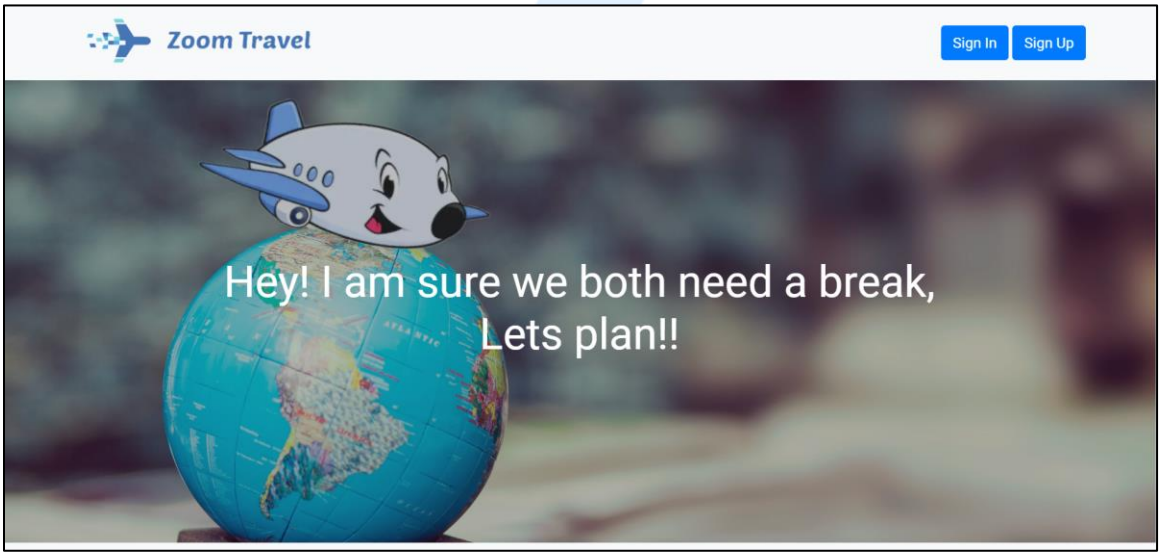
Zoom Travel's website is simple and concise. The face of Zoom Travel, Flappy, believes in travelers having experienced the same as his name, which is, Fly Happily. Its cartoony face helps nervous fliers feel at ease and make their flying experience seamless. Flappy's kind face portrays to the passengers that he is here to help.

We developed four main pages for our website namely home page, login page, error page, and about us page. The pages have been developed with Bootstrap CSS for a rich look and feel. The pages of the website are responsive and welcoming. We have implemented multiple features that are discussed below.

Website Design

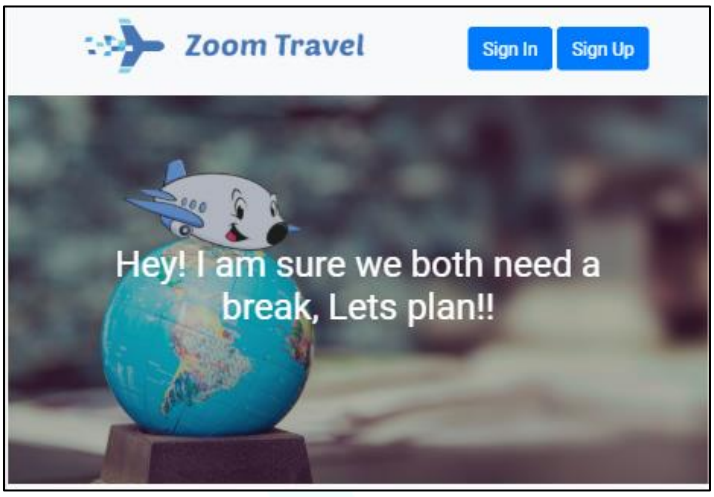
Home page or Landing Page

To begin with our website navigation, we have built our main home page or landing page that redirects any user when they hit the URL on any web browser. We first began the development of our homepage using Bootstrap CSS for creating a responsive web design.



Zoom Travel homepage on a desktop

The image above depicts the home page design for our website. The homepage design depicts the simplicity of our website. We kept a minimum number of elements and avoid unnecessary clutter.



Zoom Travel homepage on mobile

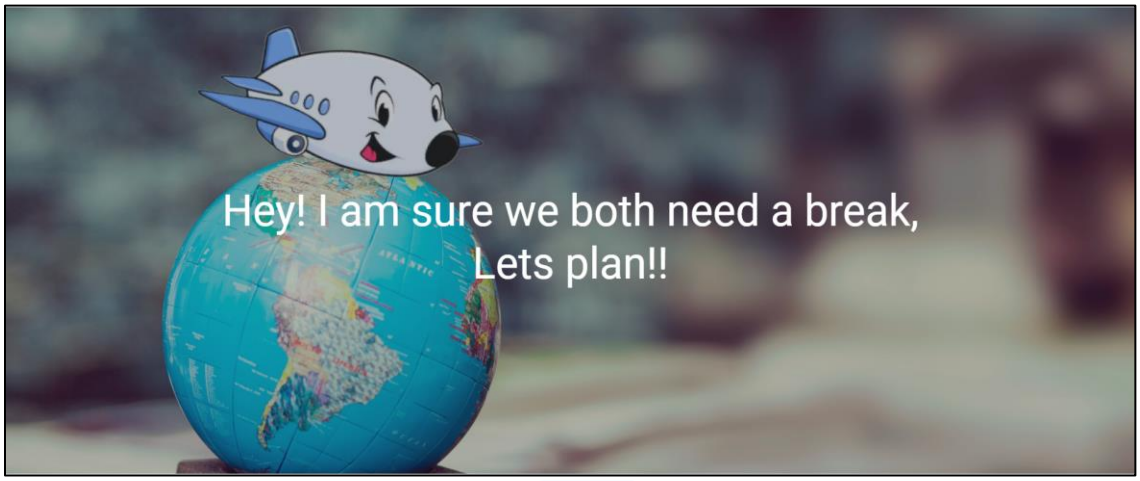
Navigation bar

Zoom Travel Navigation bar on the webpage

The navigation bar at the top of the page consists of a logo on the left and two main buttons on the right. We have kept the navigation bar simple white to maintain contrast with the below image on the website.

Zoom Travel Navigation bar on mobile

Flappy the mascot

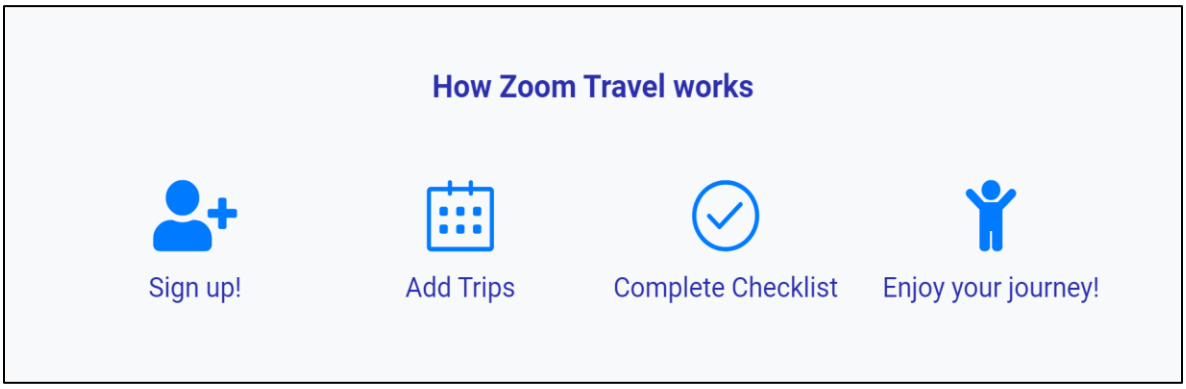


Flappy the mascot and welcome statement on the webpage

The next element we kept on our website is our mascot, Flappy. Flappy is a fun, cheerful airplane that will help navigate our customers through various checklists to make their air travel simpler. On our web page, Flappy is sitting on the globe that depicts that we help travelers all across the globe. The second most catchy element of our website is the welcome statement which says “Hey! I am sure we both need a break, Lets Plan”. The statement is simple and fun which attracts attention to our website.

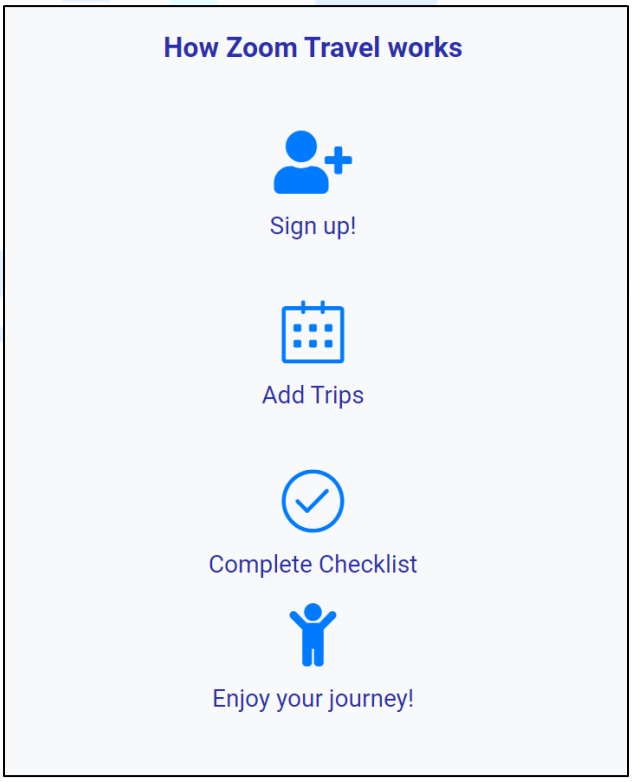
Flappy the mascot and welcome statement on mobile

How it works section



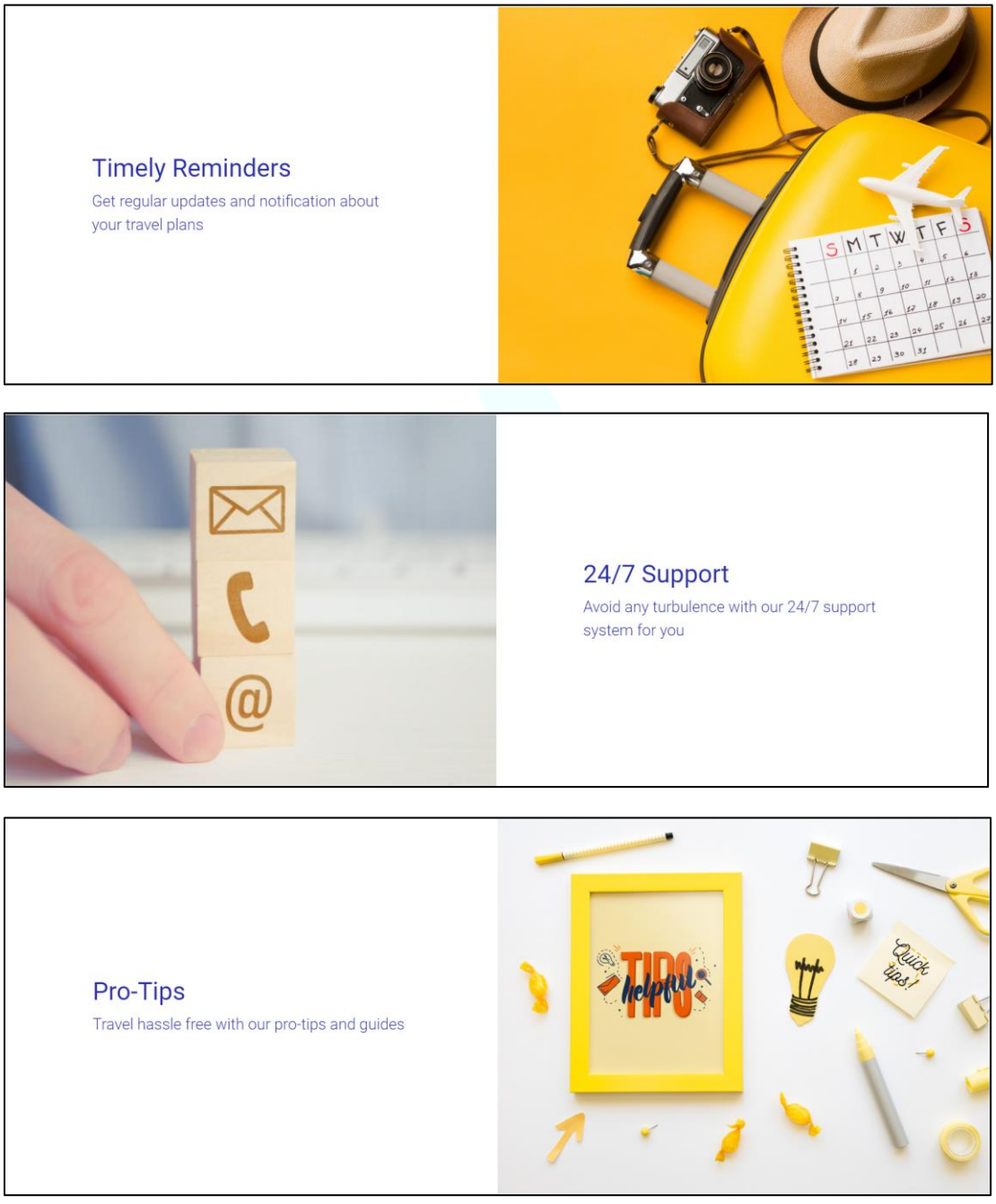
How it works section on a webpage

The main motive of our website is to ease nervous fliers and their concerns. Many customers would need more guidance or understanding of how our website would work. Hence, we have added the "How Zoom Travel works" section that provides 4 easy steps on how to work with the website. This will provide a clear picture and allow more customers to sign up with us.



How it works section on mobile

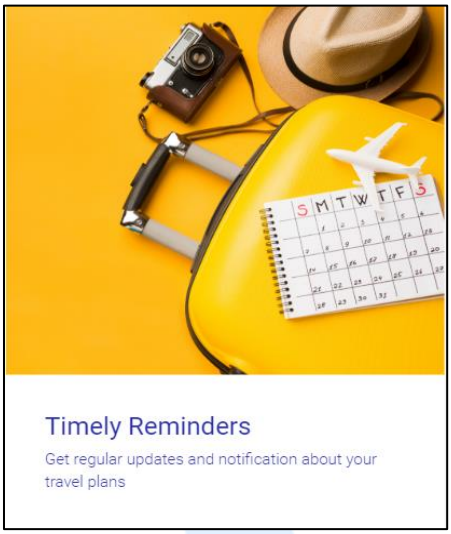
Travelling with Zoom Travel

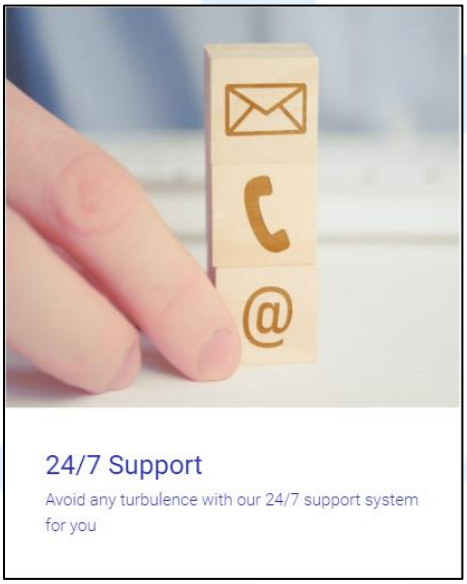


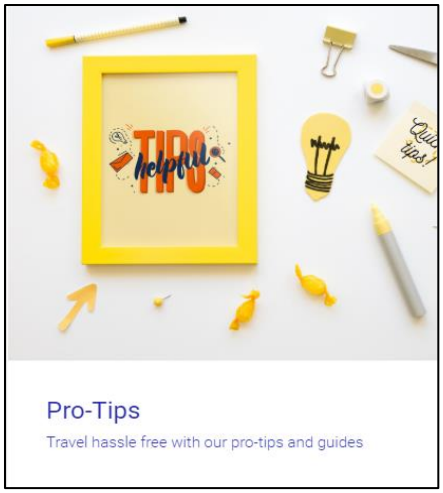
Traveling with Zoom Travel on desktop

The next section of our website is a perks section called “Travelling with Zoom Travel”. As we are just kicking off our new website, people might be skeptical about our services. Hence, to address any concerns we have added this section that might resolve any doubts about our

website and services for our customers. The section includes Timely Reminders, 24/7 Support, and Pro-tips.





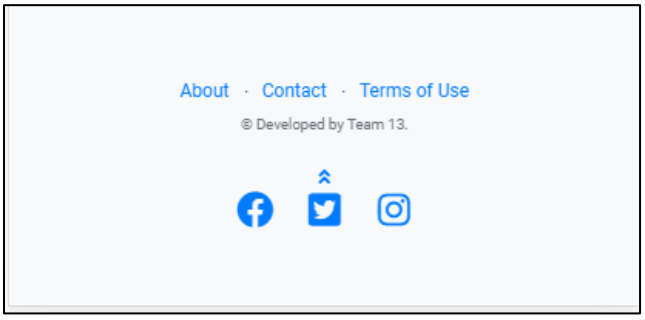


Traveling with Zoom Travel on mobile

Footer

Footer on desktop

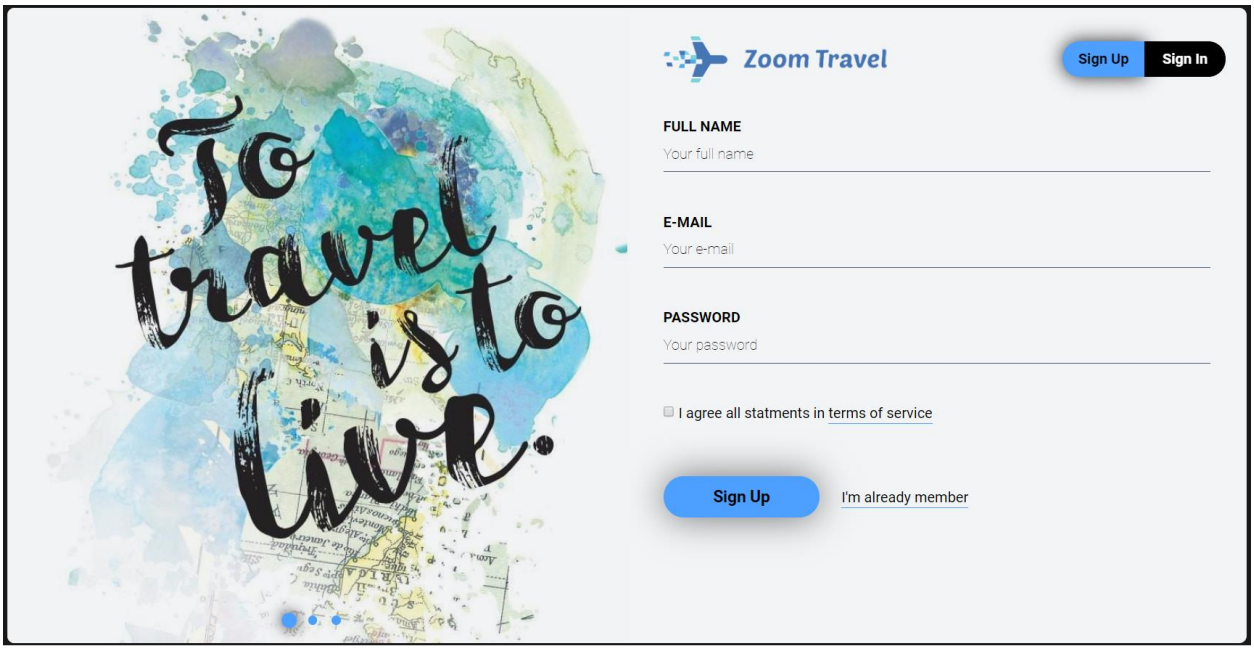
The footer section contains various links on our website such as About us, contact page, terms of use page. The section also attributes that the website is developed and maintained by Team 13. In the middle of the footer is the “navigate to top” icon that allows customers to navigate to the top of our website. The right side of the page consists of social media icons.



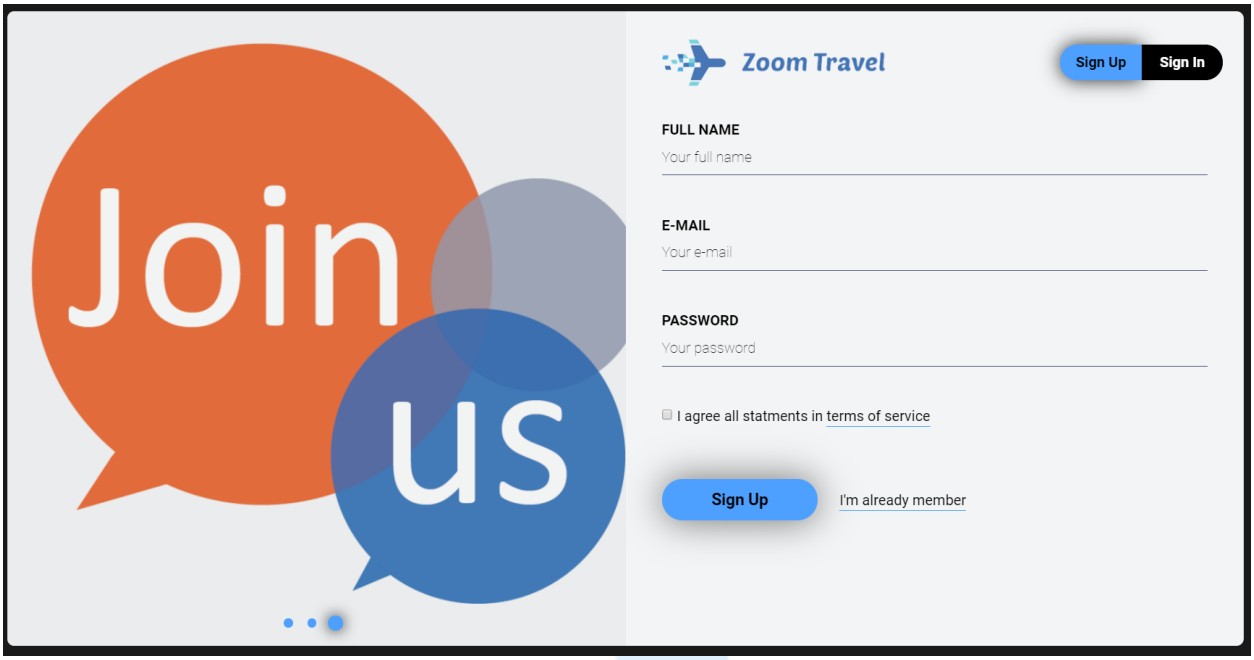
Footer on mobile

Sign-up or Sign-in page

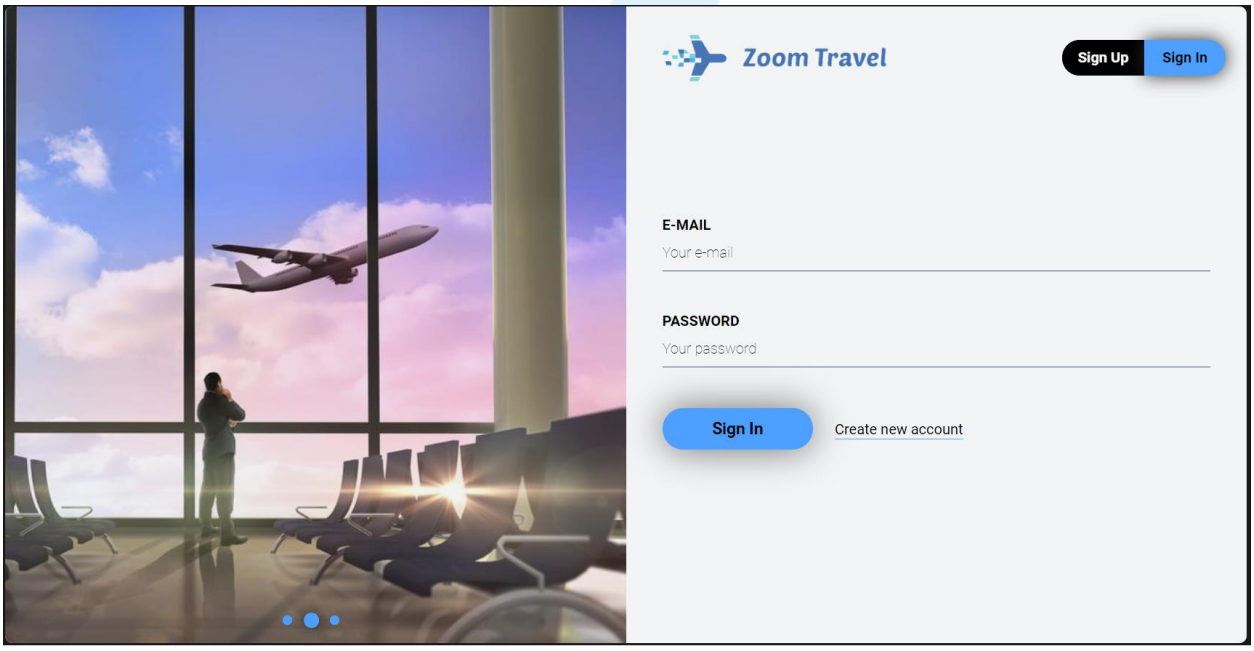
The images exhibited below are the sign-up sign-in pages for the desktop version. On the left side of the screen, there are 3 images relevant to the website. It has a slider that changes the images. There is a sign-up/sign-in form on the right-hand side of the page with a logo on the top. It has sign up/ sign-in buttons on the top which on click changes from the sign up to sign in and vice versa. Also, if you click on 'I'm already member' on the sign-up page, it will change to sign in form and if you click on 'Create new account' on the sign-in page, it will change to sign up form



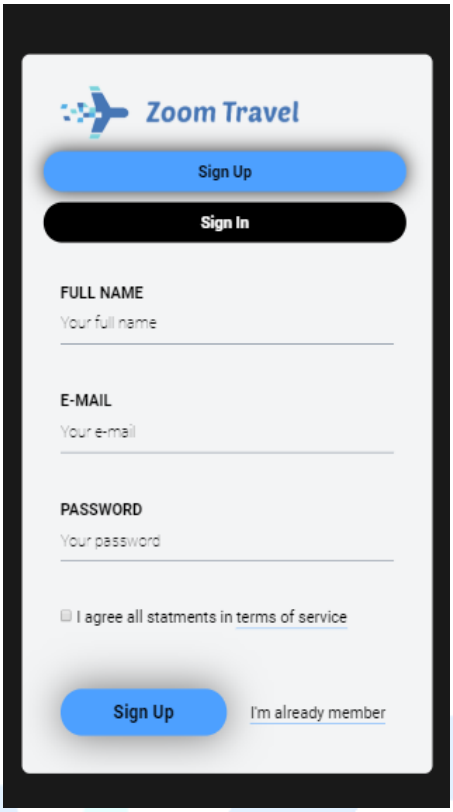
Sign up on desktop with first image



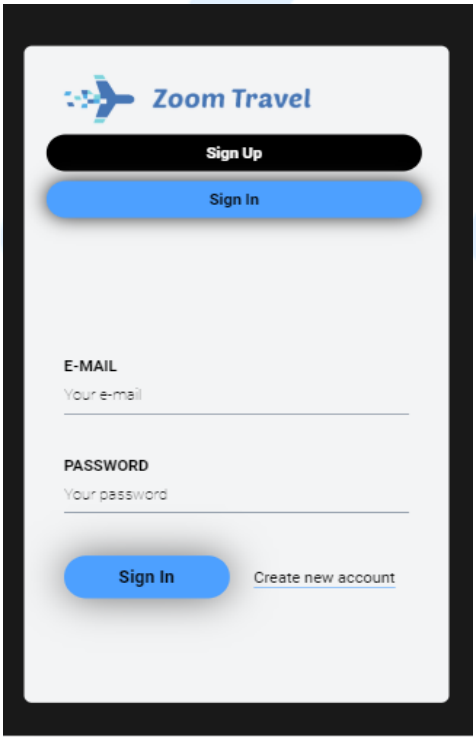
Sign up on desktop with second image



Sign in on desktop



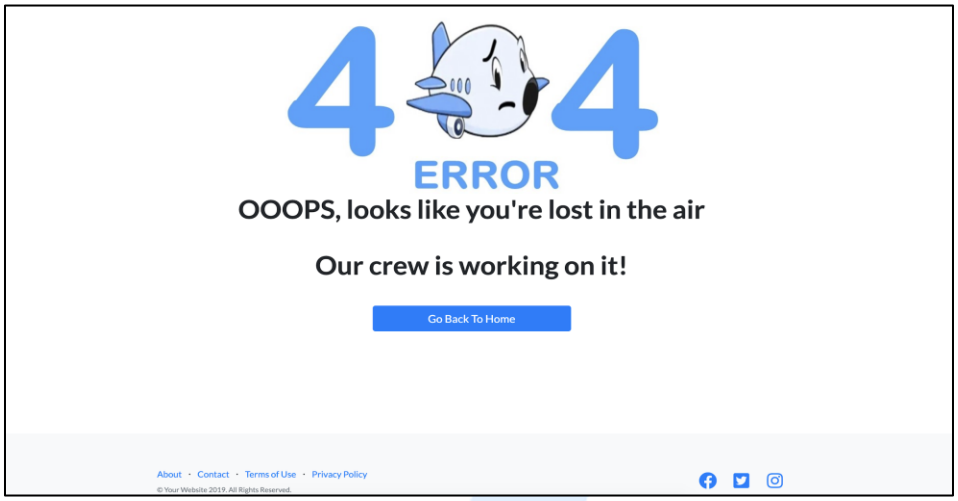
Sign up on mobile



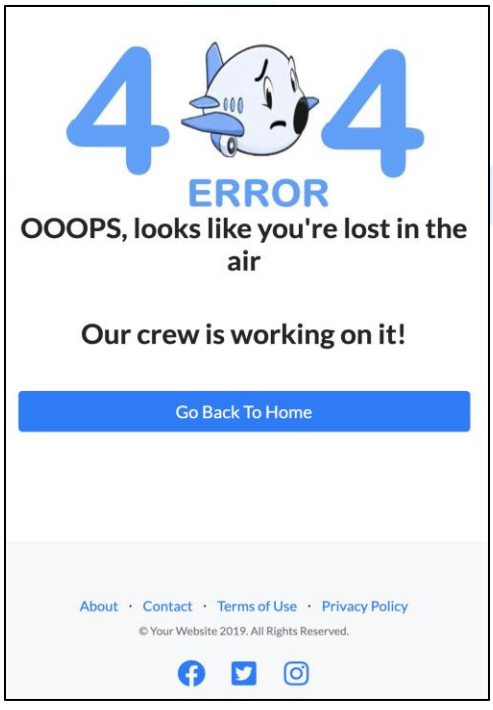
Sign in on mobile

Error page

When the user reaches this page, it might be because one of our pages is being maintained or because the user typed in random parameters to our page. We give the users the option to get back to the home page. The engagement method here is the babyface bias from our mascot. The page fits our fun but not distracting and informal but not childish brand traits so that the targeted users from elderly travelers, first-time travelers to non-frequent travelers don't get upset when they see the error page.



Error Page on desktop



Error Page on mobile

About Us page

This page majorly focuses on providing the visitor with the fair idea of what our company is about and what is our mission and our goal. The visitor also gets a sneak peek at the team which is working behind the curtains. Moreover, we have provided with a contact us section which enables the user to speak there and resolve any issues or concern they might have

Desktop Design

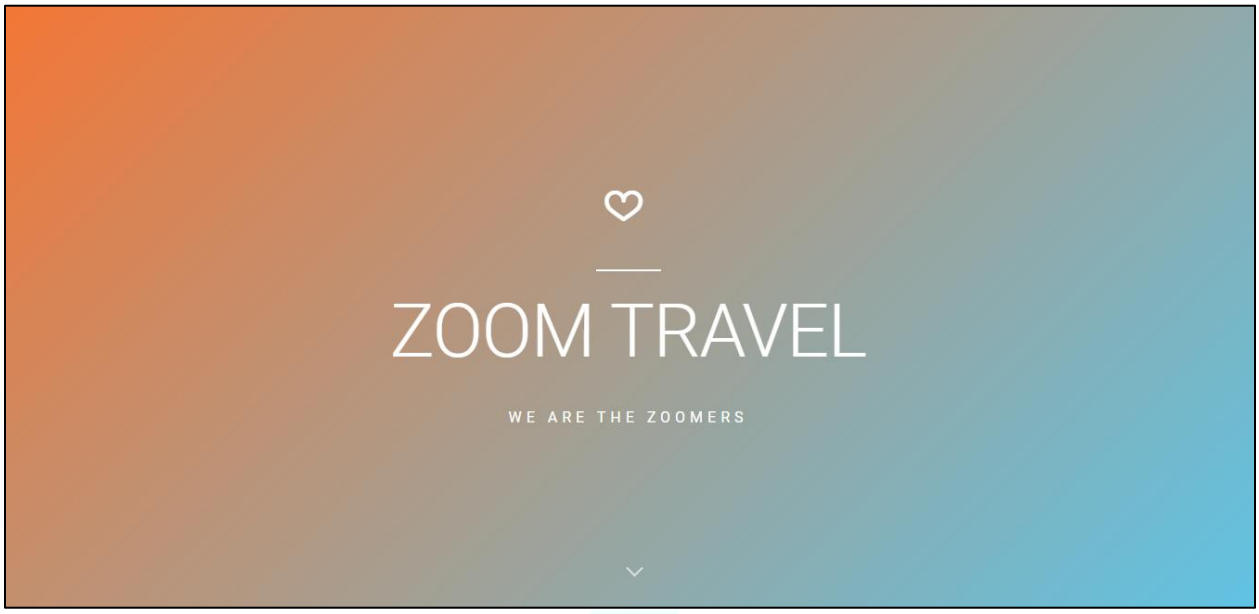
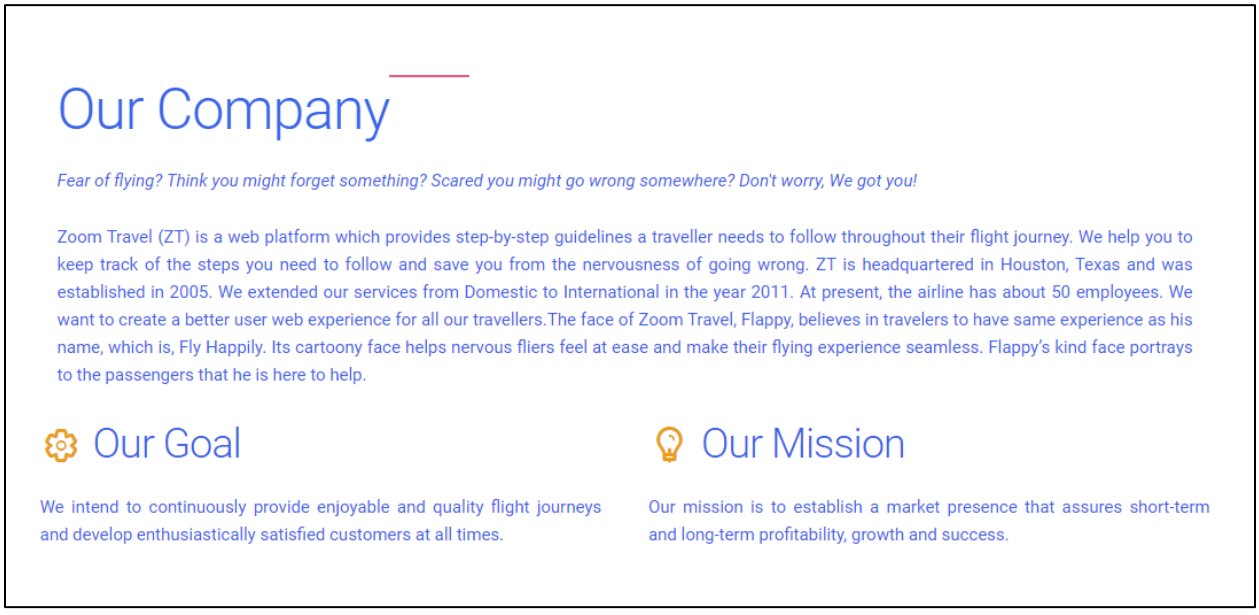
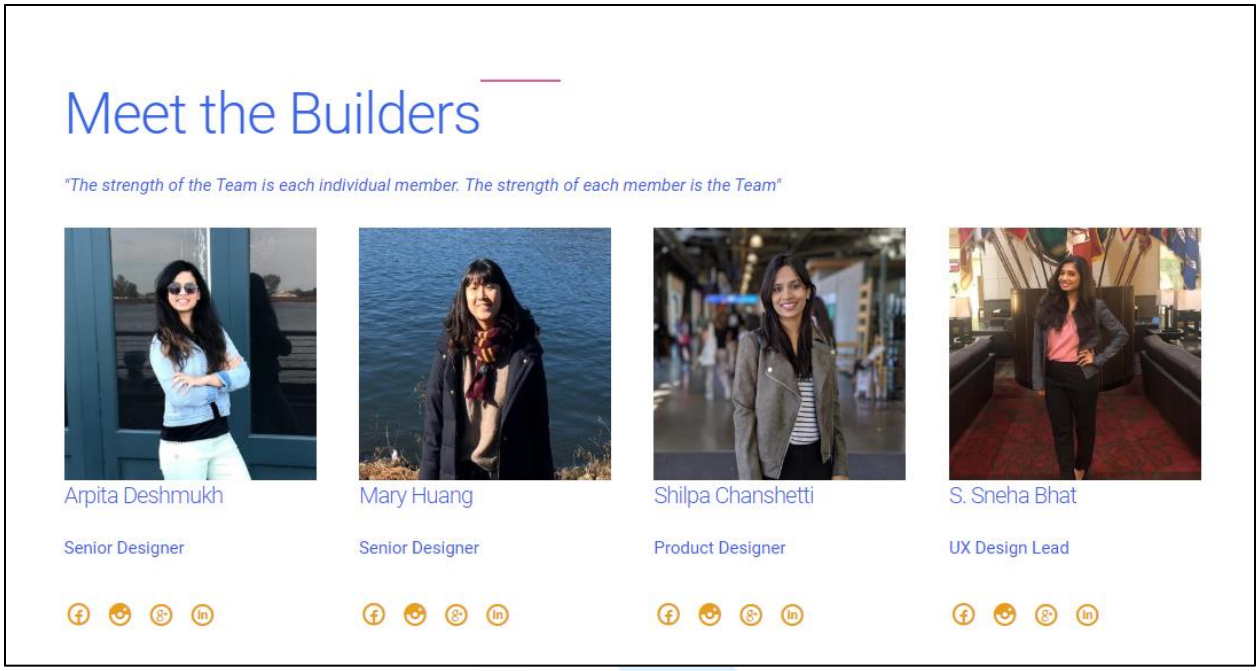


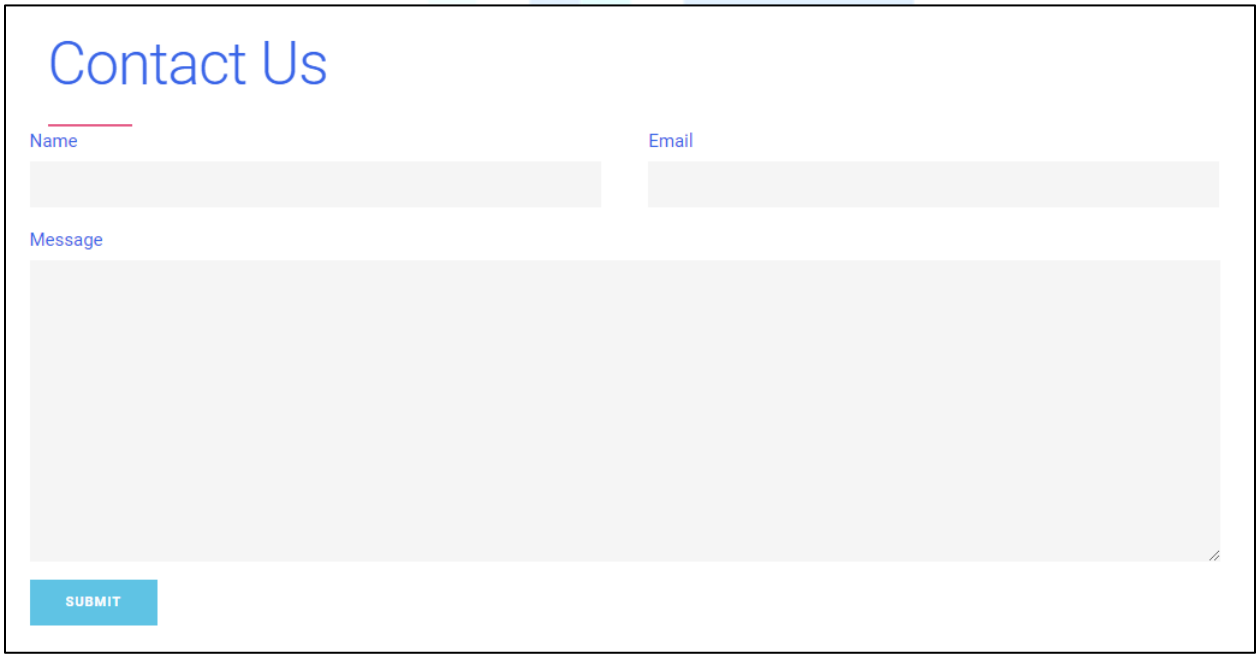
Fig 20: Header section on Desktop



About the company on Desktop



Team section on Desktop



Contact Us section on Desktop

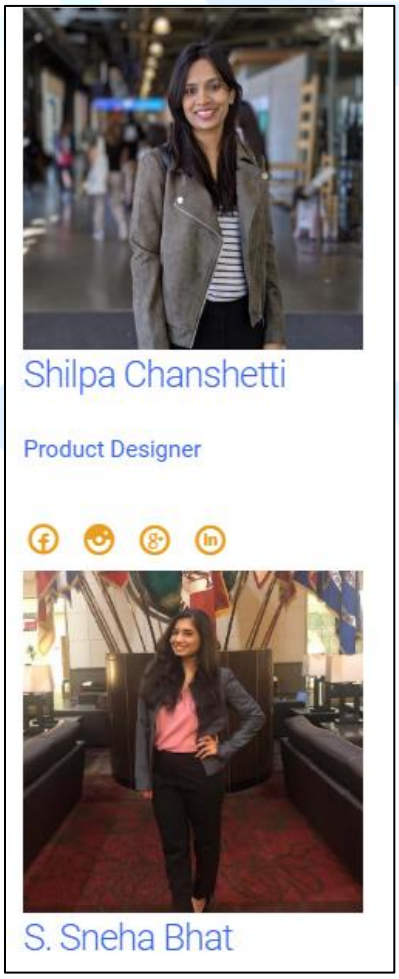
Footer on Desktop

Mobile Design

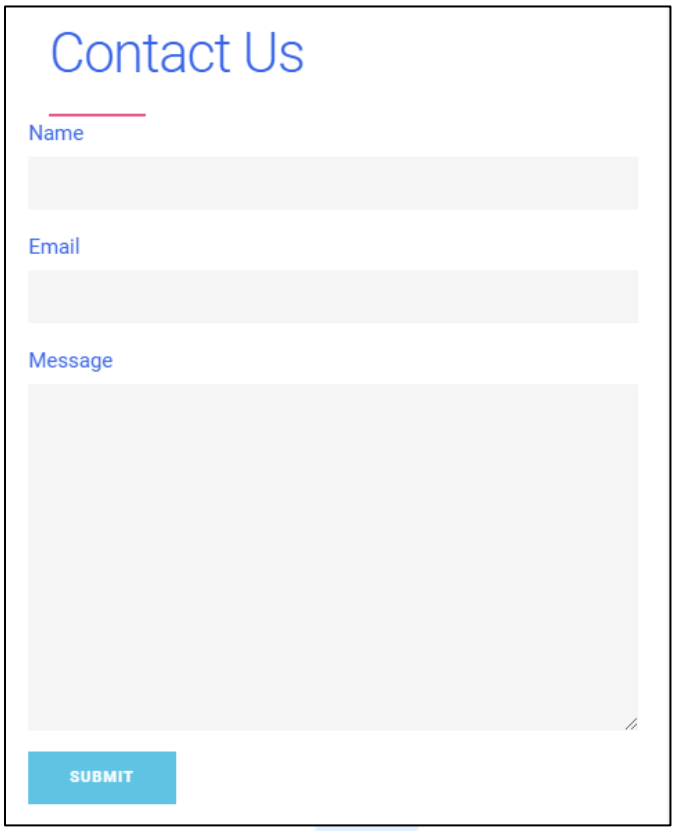
Header section on Mobile

About the company on mobile

About the company on mobile



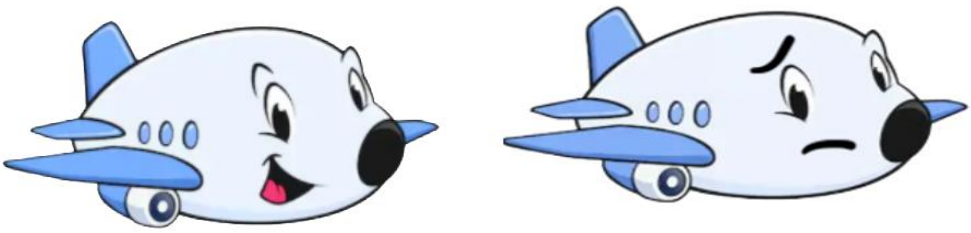
Team section on mobile



Contact us on mobile

Design Principles Applied

Baby Face Bias

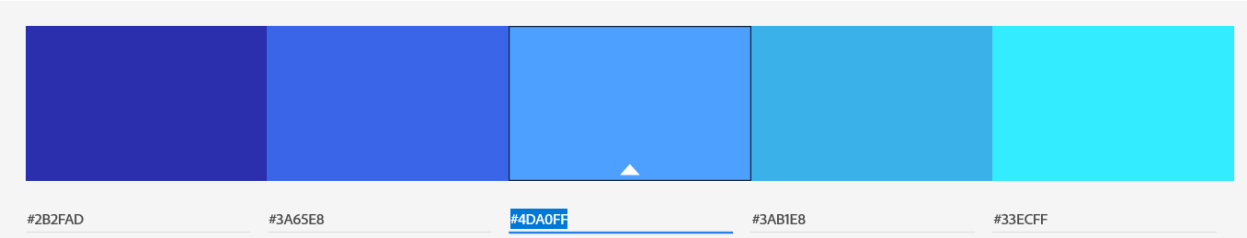


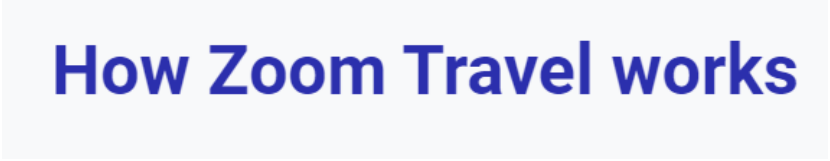
Flappy the mascot depicts baby face bias

We have our mascot, Flappy, which is a representation of the idea of babyface bias. This design element makes the view the baby-faced feature of the mascot as pure and honest

Color

The blue color is the central theme of the website. It is the color of the sky and sea. As our website is related to all the activities before sky travel, blue color resonates with it. It symbolizes trust, loyalty, and builds confidence, which we want to instill in our customer's minds as a long-lasting impression.





Font color used according to the color codes

Error message according to the color codes

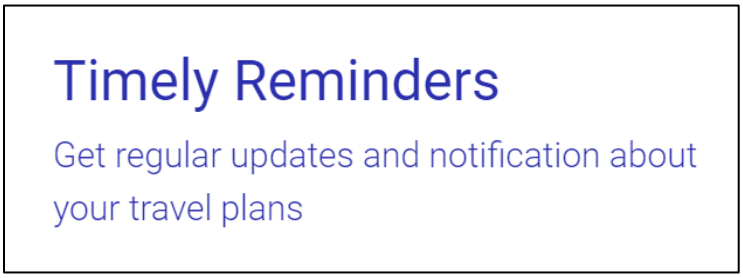
Typography

Zoom travels website is going to be designed in a way that is easy to read, understand, and visually appealing and engaging for the customers. To achieve this, typography plays an important role. To enhance the usability of the website and user experience, we will be using **Roboto** font of sans-serif typeface family for the content which will make it look stylish, exceptional, and easy to read.

This feature has been added to ensure that if the user's device does not support our primary font, Roboto, the website would still be visible in other acceptable font faces.

The following shows the implementation of adding multiple fonts.

```
*{
   font-family: 'Roboto', sans-serif;
   margin: auto;
}
```



Roboto Type

Golden Ratio

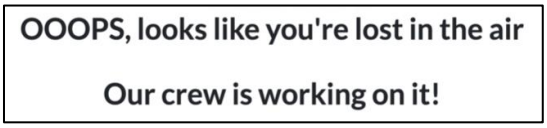
The golden ratio has been implemented on the website to make it more appealing to the human eye. The user will be naturally attracted to the website because of the golden ratio. The GIF covers 33.33% of the page, while the content covers 66.67% of the front page.

There are other examples of implementation of Golden Ratio in the webpage

Golden Ratio

Surprise & Delight

The error message is friendly and ties to our travel themed application.



Surprise Element

Visual Contrast

We have used two colors in combination to display visual contrast on the website. This is exhibited in the image below.

Visual contrast with orange and blue color

Linking user and Design persona

- The interface is simple and well designed so that it is easy for elderly travelers, first-time travelers, and non-frequent travelers to navigate what they need.
- The color scheme blue is often viewed as a non-threatening color that is the most accepted color scheme for most people
- The cute messages Flappy says can ease the concerns of the users when they are nervous about their flights
- The colorful images on the landing page can spark the users' excitement about the upcoming trip

Website Development

Software and Tools

We are using the following tools and software for developing the web application

- IDE: Visual Studio code
- Version control: Git
- Learning and Training: W3school
- Presentations: PowerPoint

Technology Stack

- Front-end: HTML, CSS, JavaScript, Bootstrap CSS
- Back-end: Python, Flask

Database

- SQLite

Host Suite

- Heroku

Conclusion

Project Plan

The team meeting would take place once a week. If required, we might schedule more meetings as per everybody's convenience. We have decided to have a Zoom call or google hangouts for the meeting purpose.

Project Deliverables

Week of	***Tasks***
Jan 20	Finish/Submit Team Contract
Jan 20	Search for project Ideas
Jan 27	Finalize project idea and work on Project Proposal
Jan 30	Project Proposal Submission
Feb 3	Discussion on User Personas
Feb 10	Work on User Personas Report
Feb 18	Work on Design Persona Report
Feb 27	Start working on the Web Design
Mar 10	Start front-end of the application
Mar 24	Work on Design Mockup Report
Mar 25	Start working on Back-end
Mar 30	Start working on Prototype
Apr 9	Work on Prototype video and report
Apr 23	Project Due Date

Learning Expectations

- Recognizing the importance of the interface design towards the success of the system
- How to make a website functional, usable as well as understandable
- Combine different elements of design to enhance the overall user experience
- Understand how business requirements can be incorporated into the project

PROTOTYPE REPORT

Website Link:

http://zoomtravel.us-east-2.elasticbeanstalk.com/

Website Design

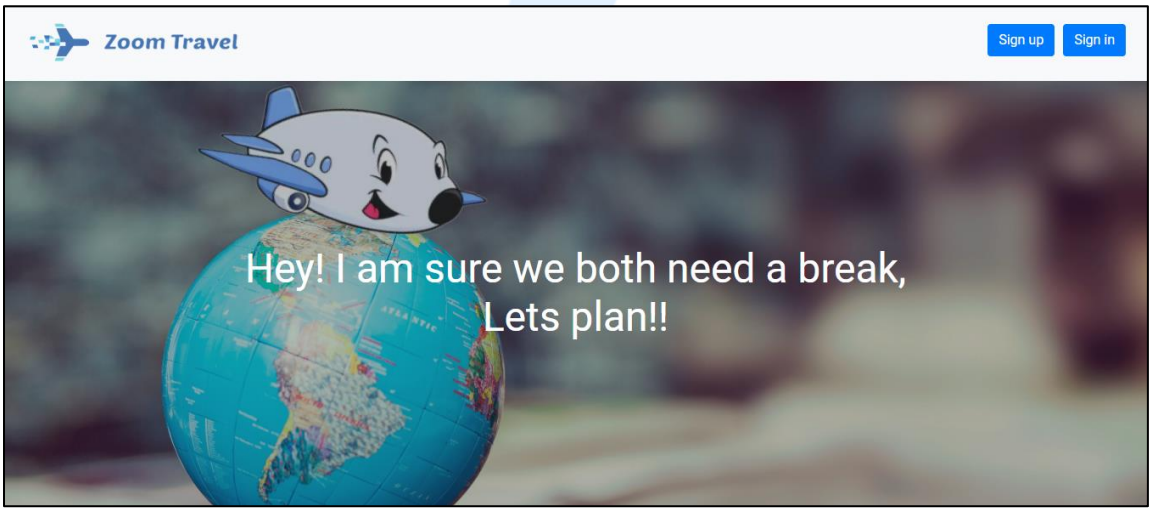
Zoom Travel (ZT) guides travelers throughout their flight journey, enabling them to have a stress free and step-by-step seamless travel experience. The face of Zoom Travel, Flappy, believes in travelers having experienced the same as his name, which is, Fly Happily. Its cartoony face helps nervous fliers feel at ease and make their flying experience seamless.

We have built a web application for easy access to the service which supports multiple devices and platforms.

This prototype report focusses on two or our functionalities i.e. User's home page and the sign-up/sign-in feature. Design principles and the responsiveness of the website is highlighted below

Home page

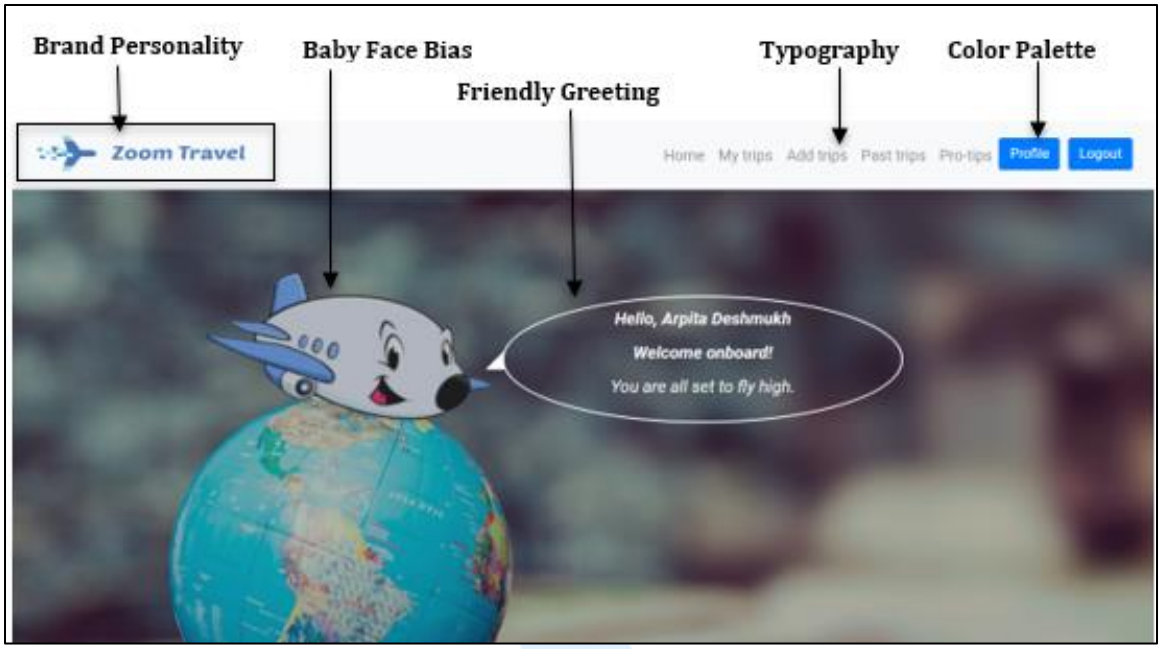
This page is displayed on the screen when the user signs-up and logs-in to their account. More navigation options are added to the previous landing page (before the user is logged in). The image below depicts the landing page design for our website. We kept a minimum number of elements and avoid unnecessary clutter.



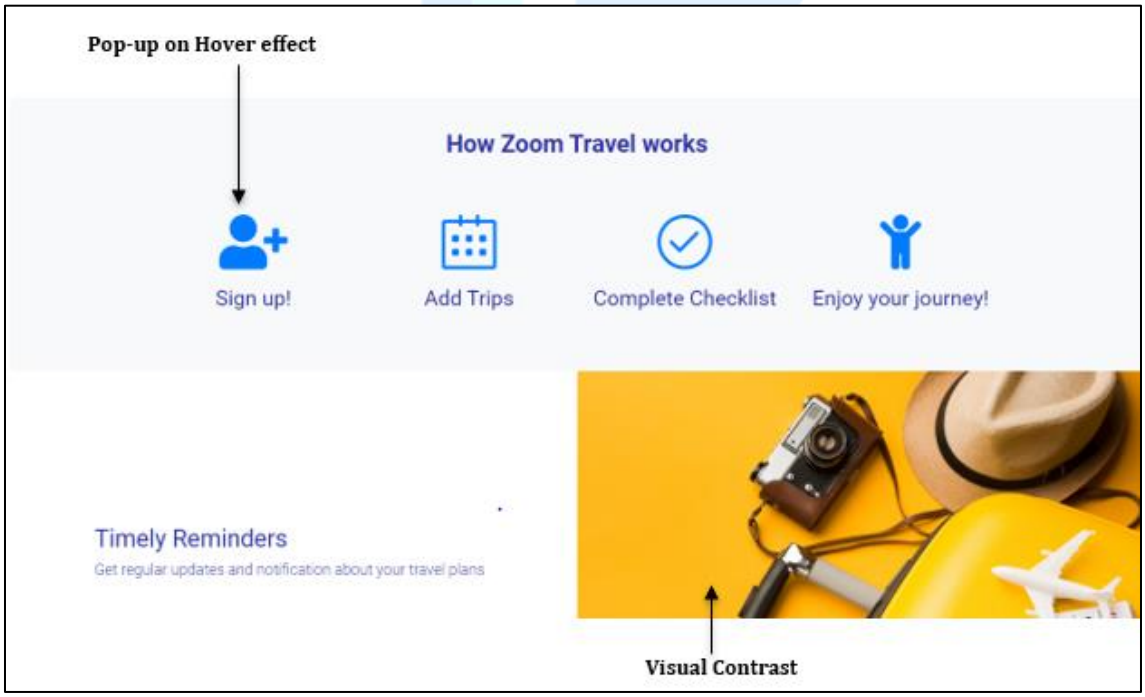
Landing Page of the Website

Design Principles

Below is the image of our landing page. The navigation bar contains the following navigation links: Home, My Trips, Add trips, Past Trips, Pro-Tips, Update Profile, and Logout



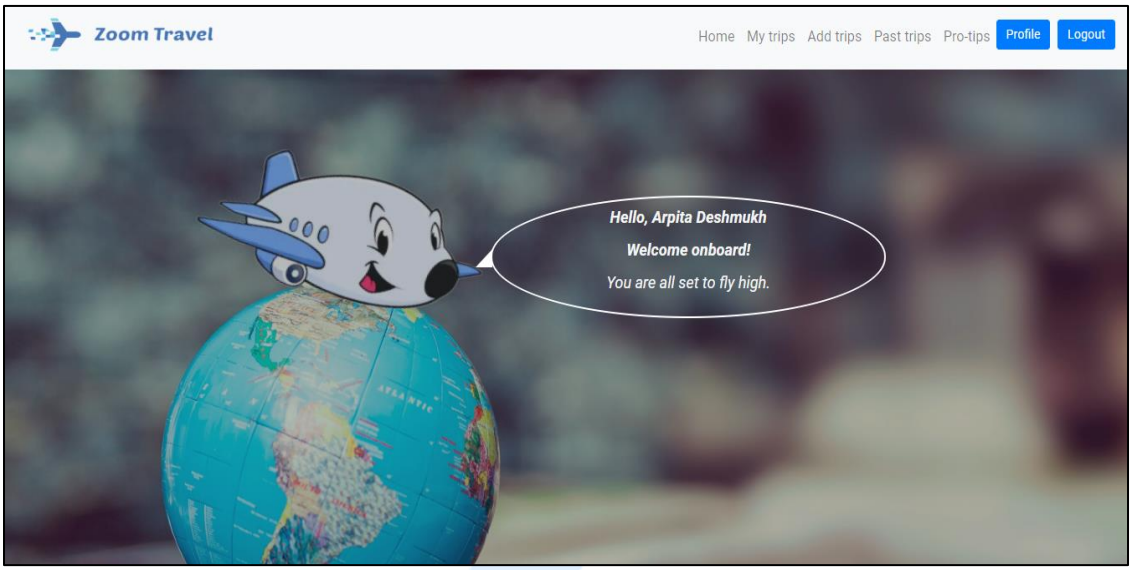
Home Page of the Website



Responsiveness

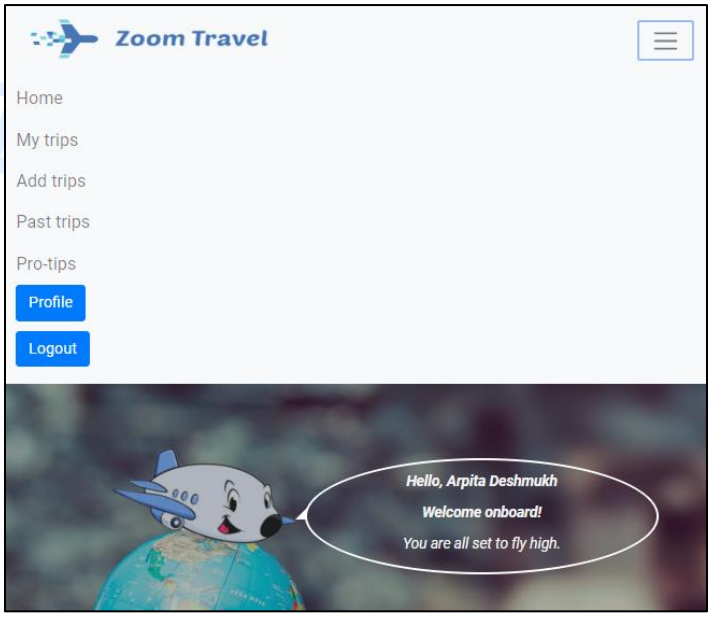
We have designed our web application in such a way which supports any device screen. We have reduced the size of the navigation bar and implemented the hamburger navigation to support mobile devices. The responsiveness of the website is shown below

Desktop Version



Landing Page on Desktop

Mobile Version

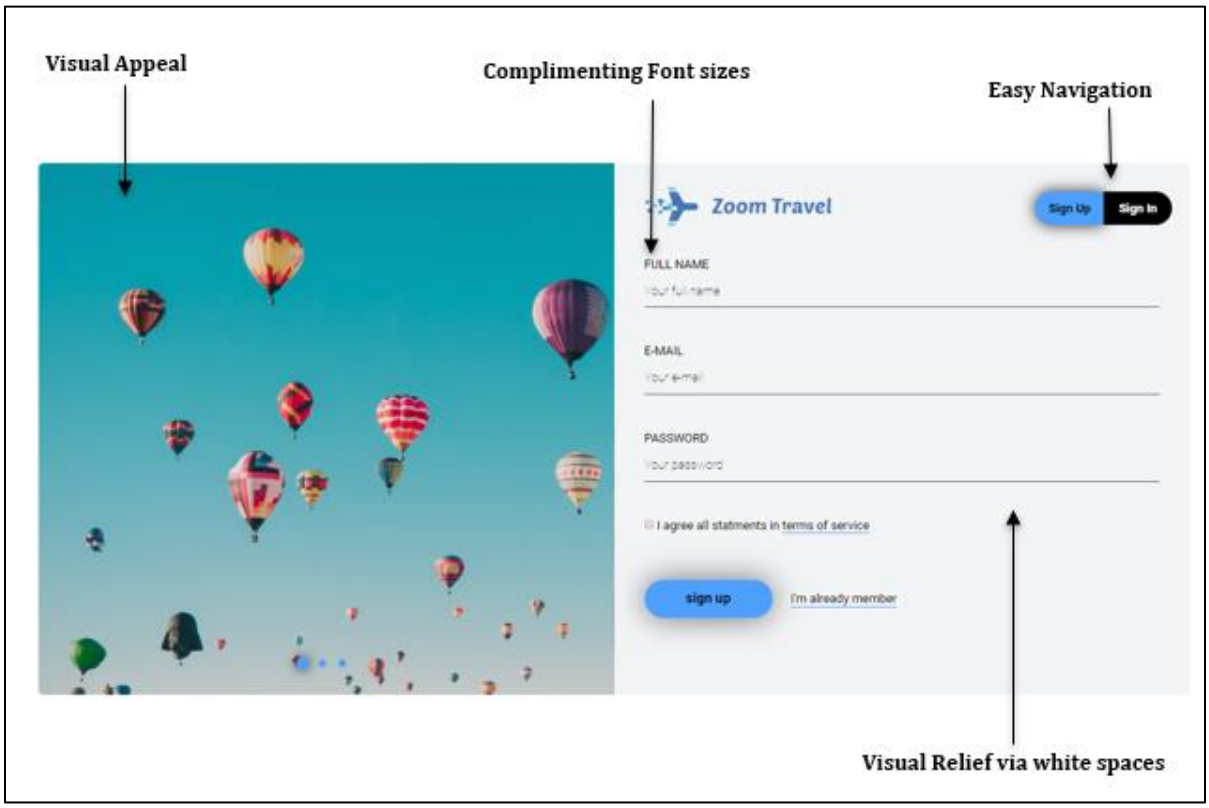


Landing Page on Mobile

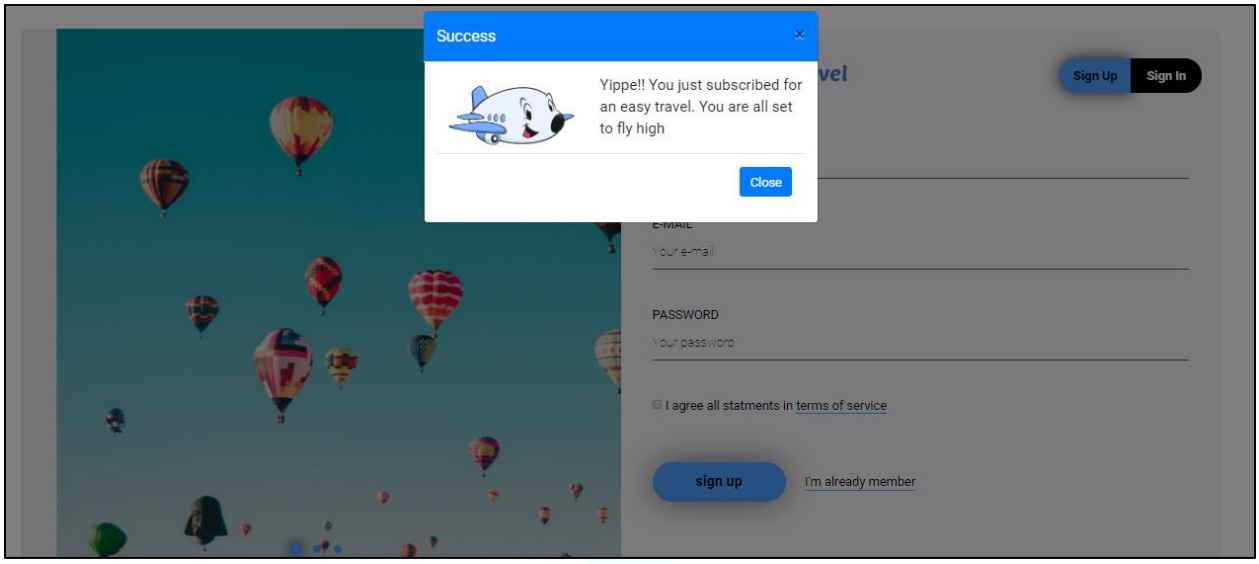
Sign-up and Sign-in page

This feature enables our users to gain access to our service by creating a personal account. On the left side of the screen, there are 3 images relevant to the website. It has a slider that changes the images. There is a sign-up/sign-in form on the right-hand side of the page with a logo on the top. It has the sign up/ sign-in buttons on the top which on click changes from sign-up to sign-in and vice versa. Also, if you click on 'I'm already member' on the sign-up page, it will change to sign in form and if you click on 'Create new account' on the sign-in page, it will change to sign up form

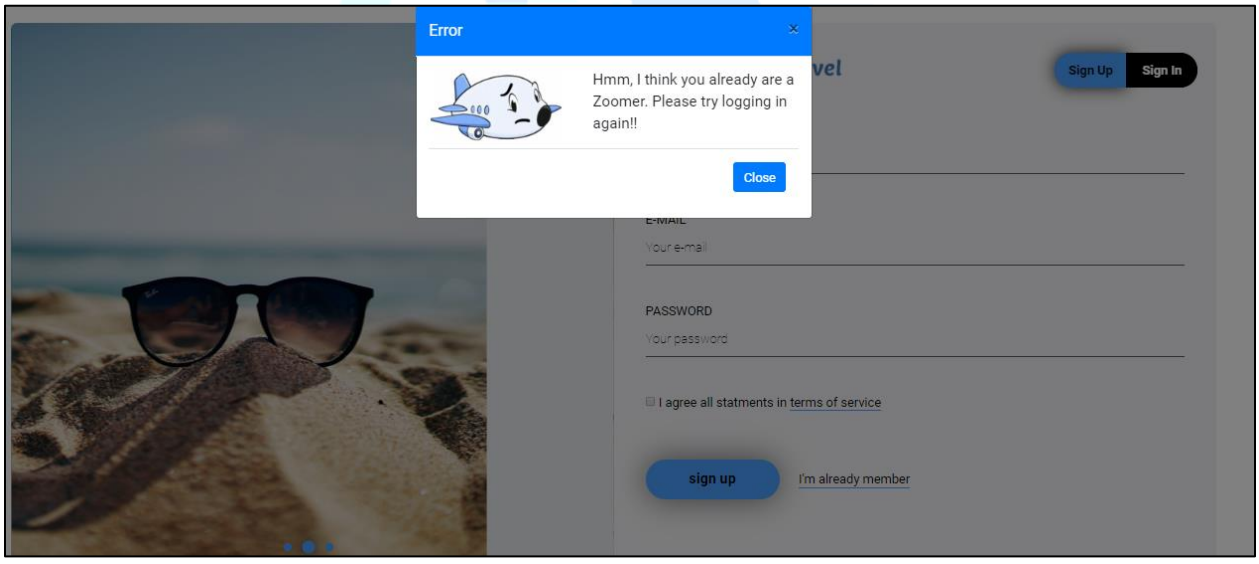
Design Principles



Sign-up page



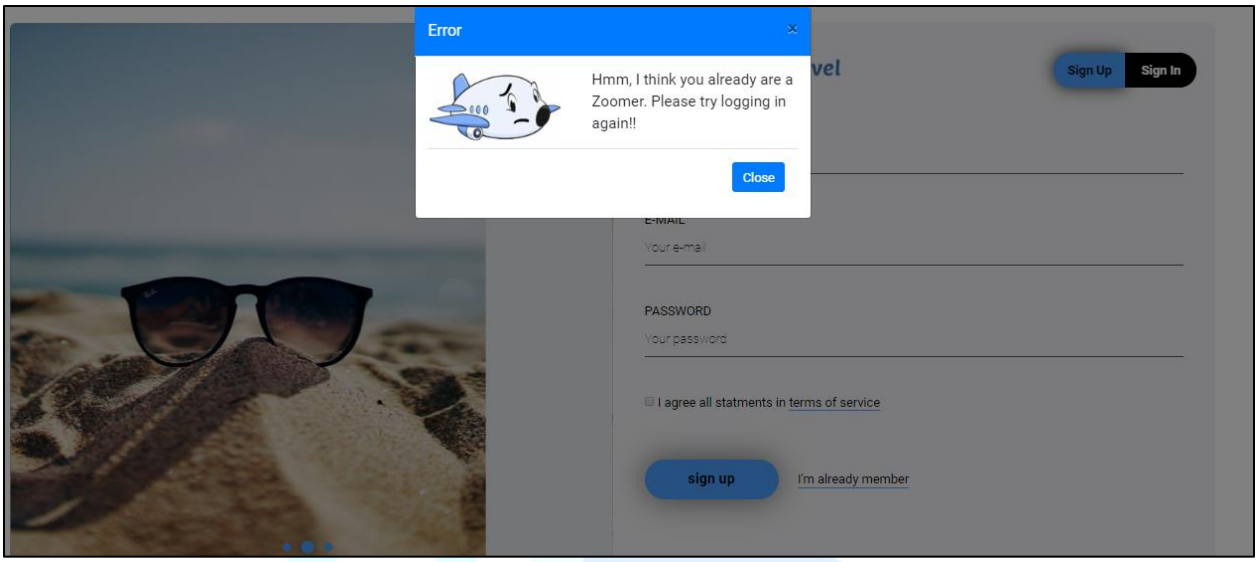
After Successful Sign Up



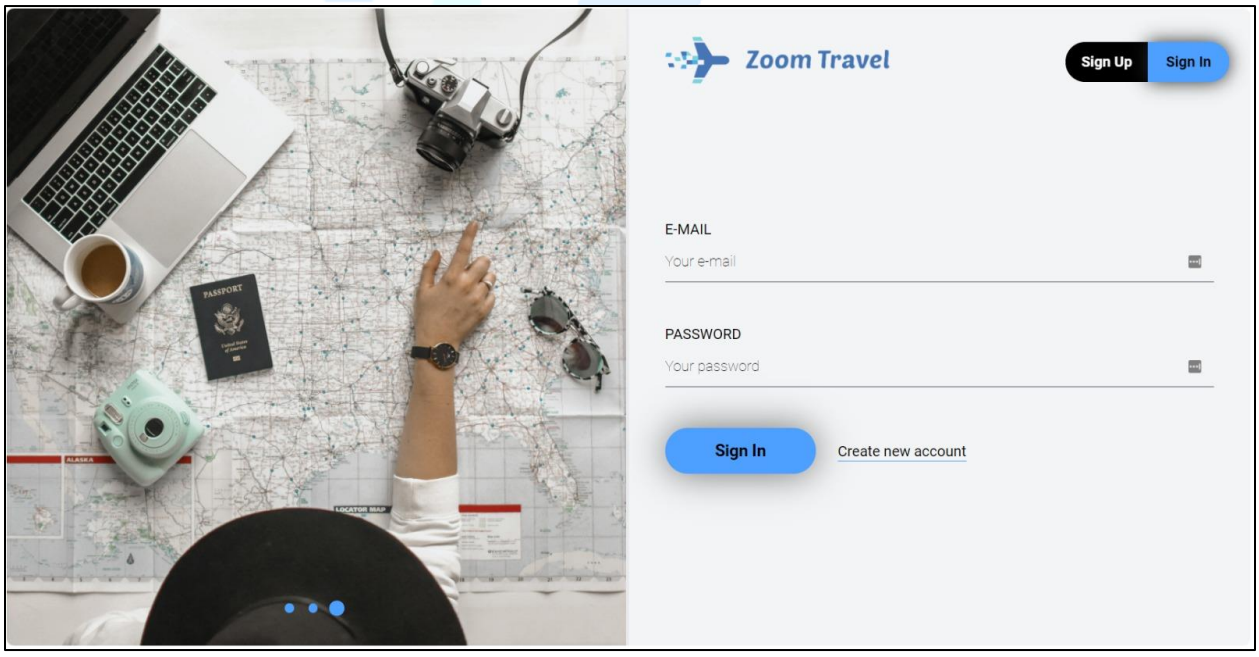
After Signing up when you already have an account

Responsiveness

Desktop Version

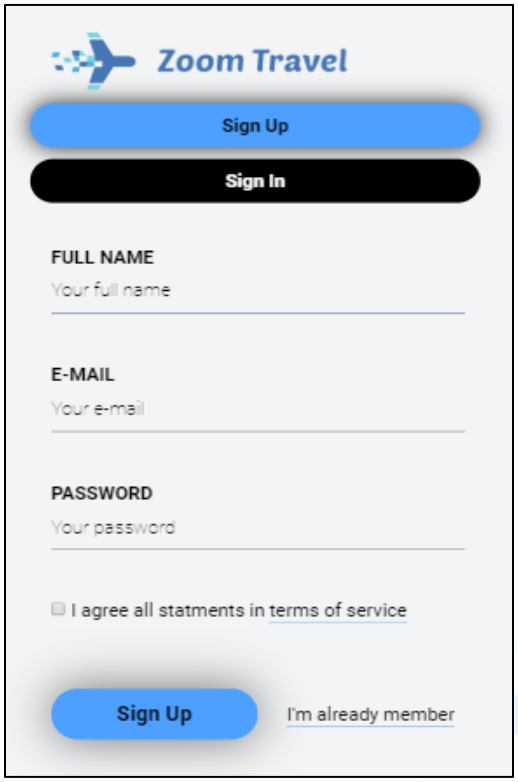


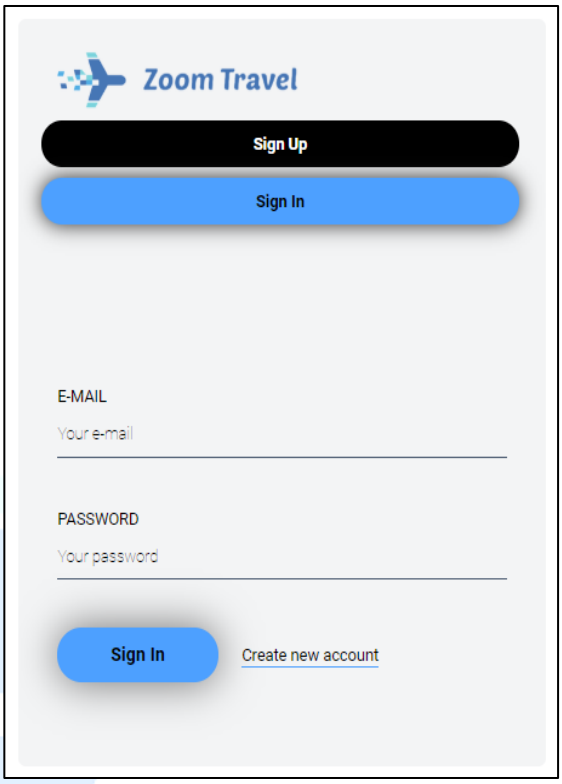
Sign-Up Page



Sign-In Page

Mobile version

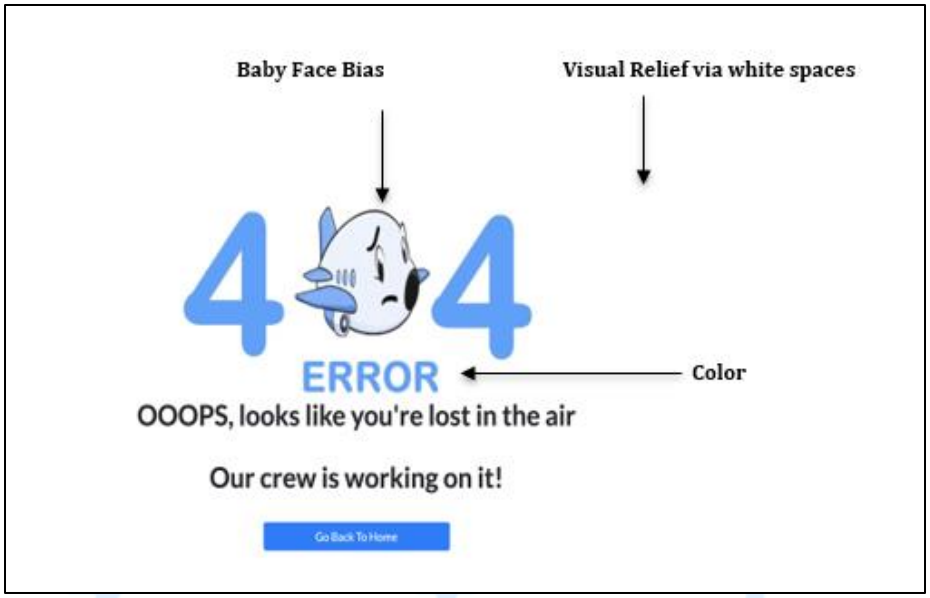




Error page

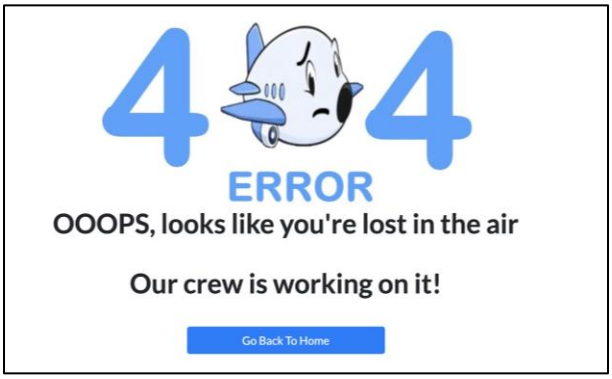
When the user reaches this page, it might be because one of our pages is being maintained or because the user typed in random parameters to our page. We give the users the option to get back to the home page. The engagement method here is the babyface bias from our mascot. The page fits our fun but not distracting and informal but not childish brand traits so that the targeted users from elderly travelers, first-time travelers to non-frequent travelers don't get upset when they see the error page.

Design Principles

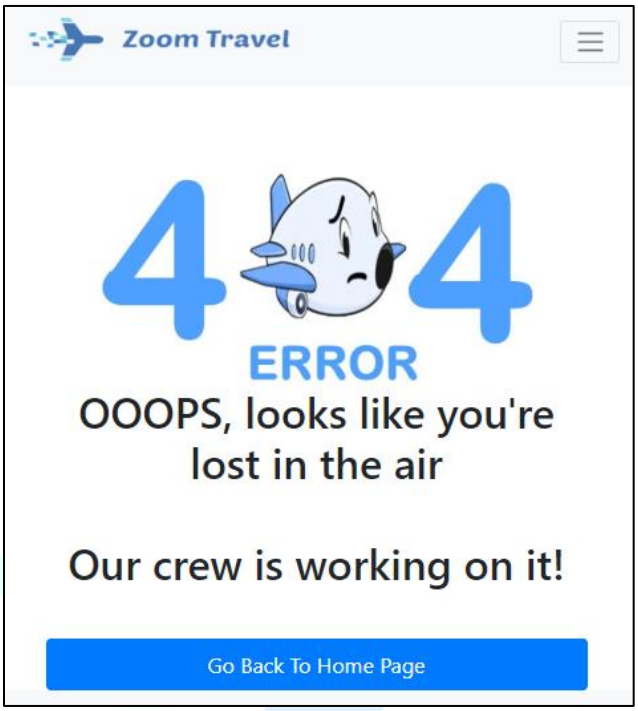


Error Page of the Website

Responsiveness



Error Page on Desktop



Error Page on Mobile

Website Development

Software and Tools

We are using the following tools and software for developing the web application

- IDE: Visual Studio code
- Version control: Git
- Learning and Training: W3school
- Presentations: PowerPoint

Technology Stack

- Front-end: HTML, CSS, JavaScript, Bootstrap CSS
- Back-end: Python, Flask

Database

- AWS - RDS

Host Suite

- AWS Elastic Beanstalk

Updates since Design Mock-up

Database Connectivity –

After completing the HTML pages, our next step was to implement the database connectivity. We hosted our database on Amazon Relational Database Service (RDS). It is a distributed relational database service by Amazon Web Services. This web service runs on the cloud designed to simplify the setup, operation, scaling of a relational database for the use in applications. [1]

Hosting the Web Application –

We hosted our web application on AWS Elastic Beanstalk. This service by Amazon Web Services for deploying applications.

Sign-Up and Sign-In –

The first functionality which required interaction with the database was Sign-Up and Sign-In functionality. After creating an account, the user needs to sign in to his/her profile using the same credentials. These credentials are cross-checked with the credentials entered by the user when creating the account. This feature displays the first interaction with the backend

Home Page –

This is the first web page which the user view after creating and logging into the account on our website. The home page mimics the landing page with the difference in the features on the navigation bar. Moreover, when a user signs in, Flappy (the mascot) greets the user with the following message:

"Hello *[UserName]*

Welcome onboard

You are all set to fly high"

Conclusion

For the prototype phase, we have only focused on implementing two functionalities completely and aiming at achieving our objective of making the website friendly and super-easy to use.

Learning expectations

- Recognizing the importance of the interface design towards the success of the system
- How to make a website functional, usable as well as understandable
- Combine different elements of design to enhance the overall user experience
- Understand how business requirements can be incorporated into the project

References

[1] https://en.wikipedia.org/wiki/Amazon_Relational_Database_Service

FINAL REPORT

Zoom Travel

Website Link:

http://zoomtravel.us-east-2.elasticbeanstalk.com/

Overall Description

Zoom Travel (ZT) guides travelers throughout their flight journey, enabling them to have a stress free and step-by-step seamless travel experience. The face of Zoom Travel, Flappy, believes in travelers having experienced the same as his name, which is, Fly Happily. Its cartoony face helps nervous fliers feel at ease and make their flying experience seamless.

The key functionalities of ZT are:

- **User Registration:** A user can register on the website and can make changes to their profile
- **Add Flight Details:** A registered user can add flight travel details
- **View Checklist:** Based on the trips added and the type of travel (national/international), the user can complete the checklist (the steps to be followed throughout the flight journey). The traveler would be able to view checked as well as unchecked steps.
- **View Past Trips:** A registered user can also view the past trips if any
- **Pro-Tips:** Providing pro-tips to all the users

ZT wants to create a better user web experience for all its travelers. The main purpose of the website is to help first-time flight travelers to have hassle-free flying experience.

Navigating the Website

Site Link: http://zoomtravel.us-east-2.elasticbeanstalk.com/

Test Credentials:

Username: test@gmail.com

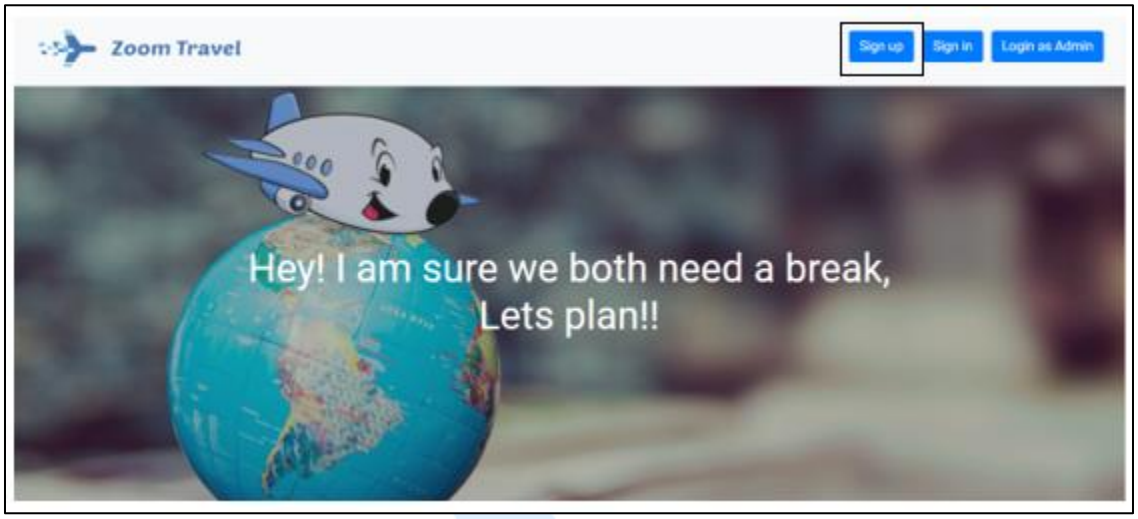
Password: team13test@123

General guidelines to navigate the website:

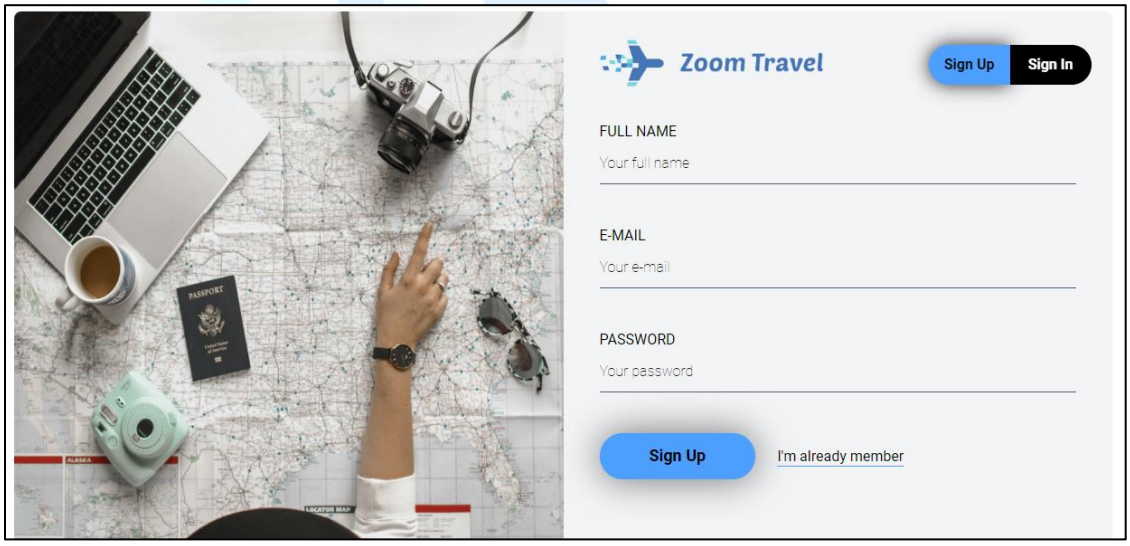
- **User Registration**
  To be able to add trip details or view previous trip details, the user needs to create an account on the website. The user can access the services offered by ZT by logging into the account.

  Steps:
  - ✓ Create a user account by clicking Sign Up on the Home page

✓ Fill out the sign-up form as shown below



Click on Sign Up to Register



Sign Up form

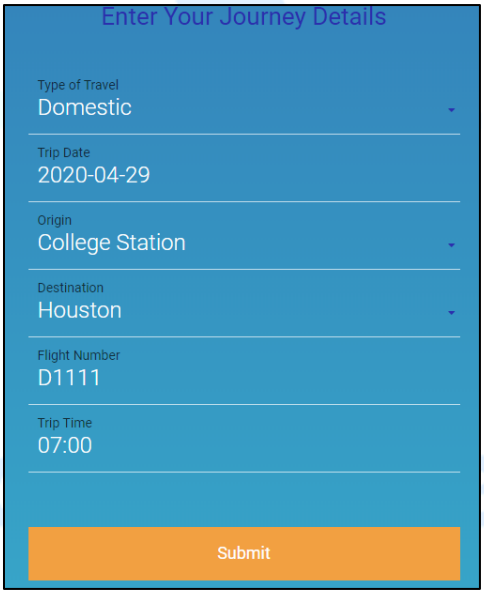
- **Add Flight Details**

  Steps:

  ✓ After signing in, the user can enter their trip details by clicking the *Add Trips* option visible on the navigation bar.

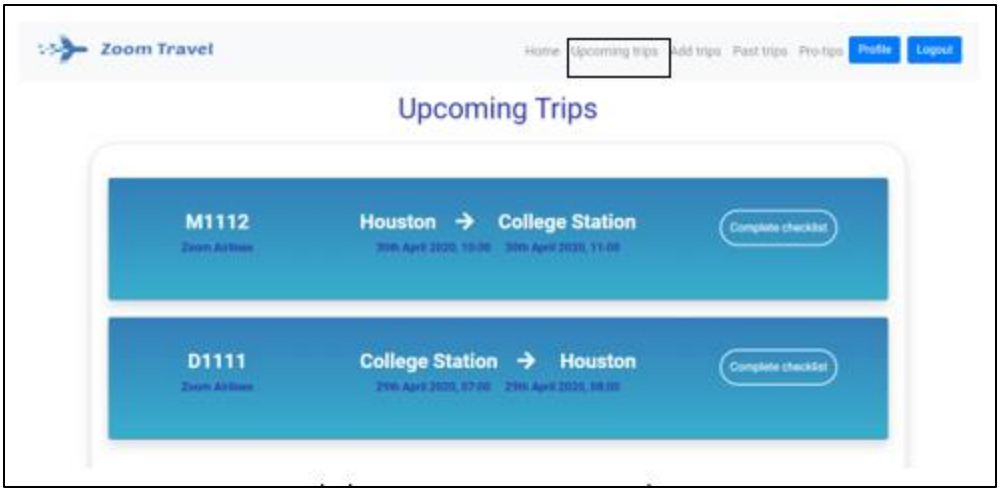
Click on Add Trip to add journey details

✓ Enter Your Journey details



Add Trips Form

✓ After you submit, your upcoming trips will be visible by clicking on the *Upcoming Trips* tab



Click on Upcoming Trips tab to view your trips

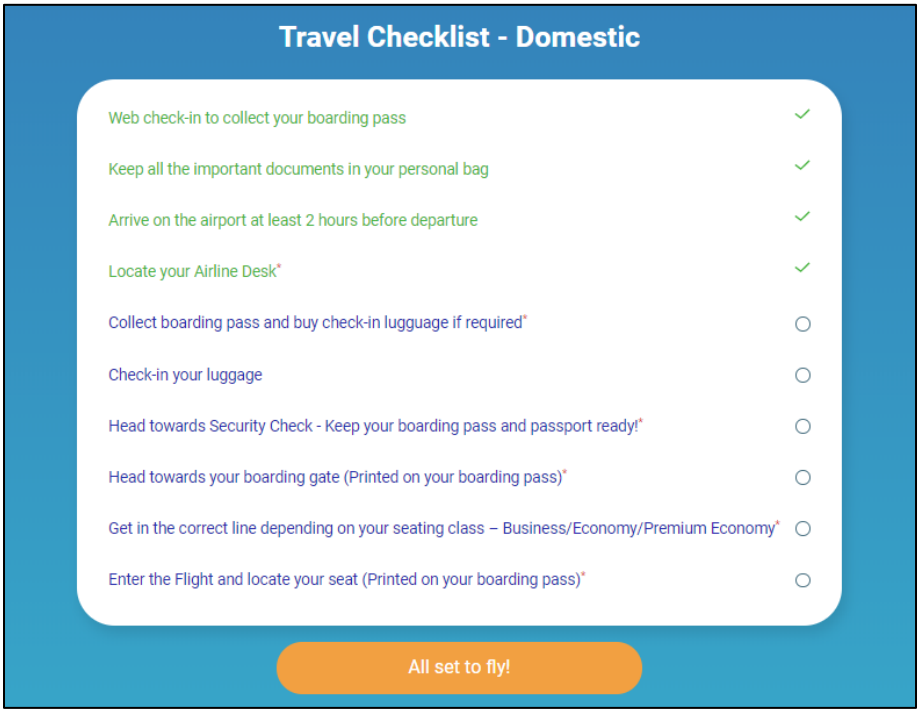
- **View Checklist**

  The most important functionality of ZT is to create a checklist for the users so that we can guide them throughout their journey

  Steps:

  ✓ You can view the *Complete Checklist* option beside your upcoming trips

  ✓ The checklist will be displayed, and the user can check the items off the list as shown below

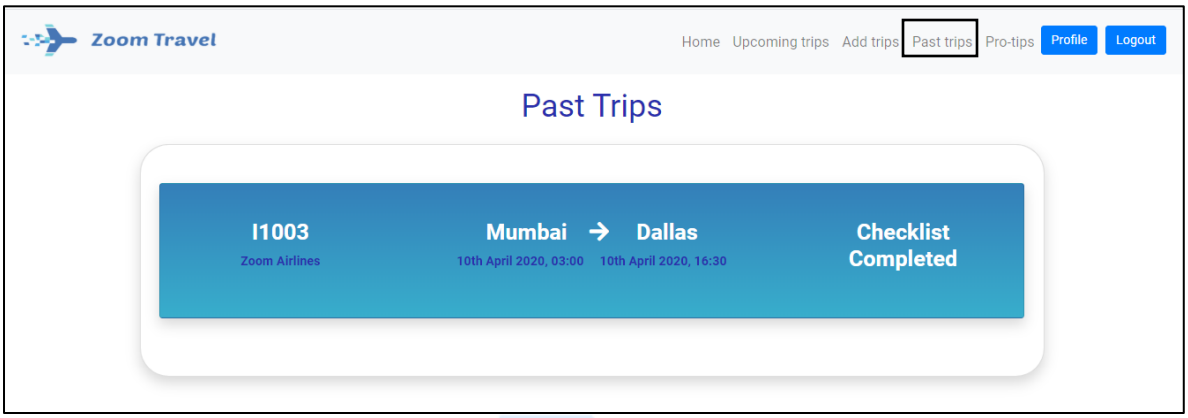


View Checklist

- **View Past Trips**

  Steps:

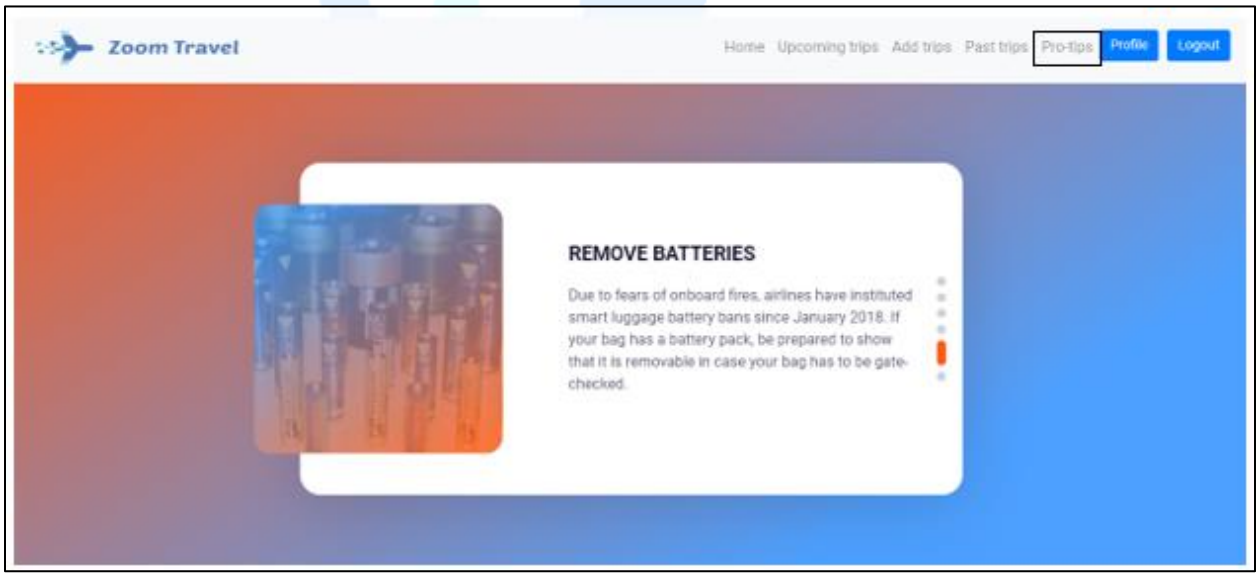
  ✓ Click on *Past Trips* on the navigation bar as shown below



Click on Past Trips to view your completed Journey details

- **Pro-Tips**

  Pro-Tips are displayed to keep the users best prepared before their air travel. The users can access the page as shown below



Pro-Tips Page

Elements of Design

Baby Face Bias

We have our mascot, Flappy, which is a representation of the idea of babyface bias. This design element makes the view the baby-faced feature of the mascot as pure and honest

Color

The blue color is the central theme of the website. It is the color of the sky and sea. As our website is related to all the activities before sky travel, blue color resonates with it. It symbolizes trust, loyalty, and builds confidence, which we want to instill in our customer's minds as a long-lasting impression.

Typography

Zoom travels website is going to be designed in a way that is easy to read, understand, and visually appealing and engaging for the customers. To achieve this, typography plays an important role. To enhance the usability of the website and user experience, we will be using **Roboto** font of sans-serif typeface family for the content which will make it look stylish, exceptional, and easy to read.

This feature has been added to ensure that if the user's device does not support our primary font, Roboto, the website would still be visible in other acceptable font faces.

The following shows the implementation of adding multiple fonts.

```
*{
   font-family: 'Roboto', sans-serif;
   margin: auto;
}
```

Golden Ratio

The golden ratio has been implemented on the website to make it more appealing to the human eye. The user will be naturally attracted to the website because of the golden ratio. The GIF covers 33.33% of the page, while the content covers 66.67% of the front page.

There are other examples of implementation of Golden Ratio in the webpage

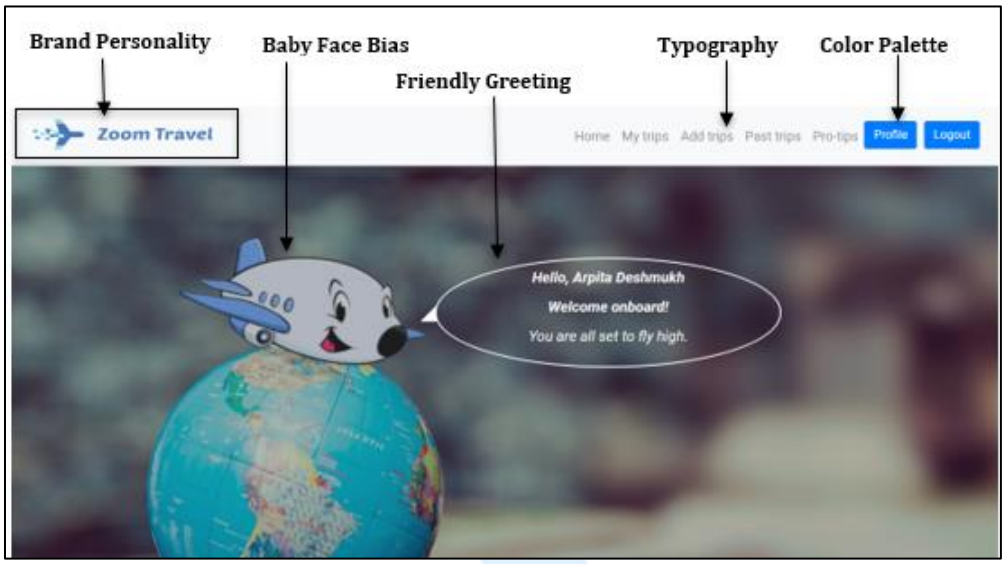
Surprise & Delight

The error message is friendly and ties to our travel themed application.

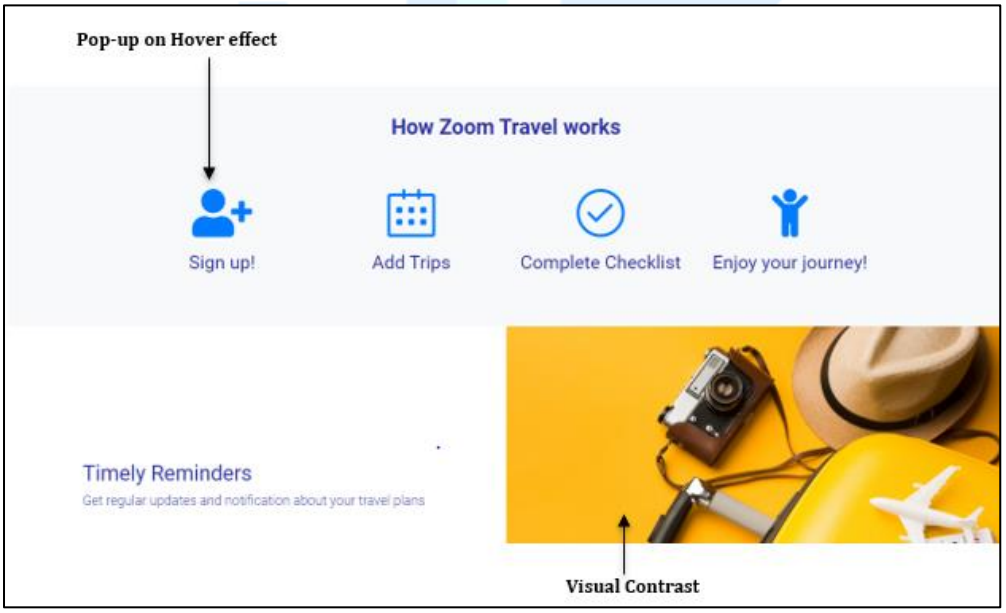
Visual Contrast

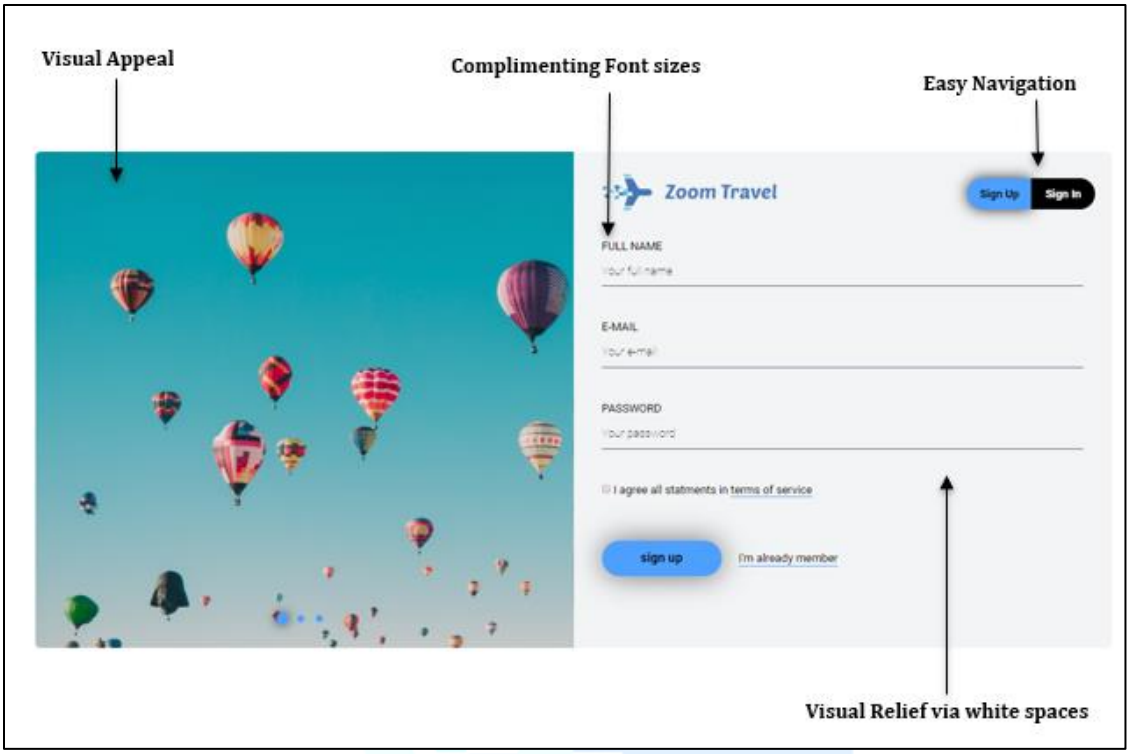
We have used two colors in combination to display visual contrast on the website. This is exhibited in the image below.

The design principles incorporated in our web application are as shown below

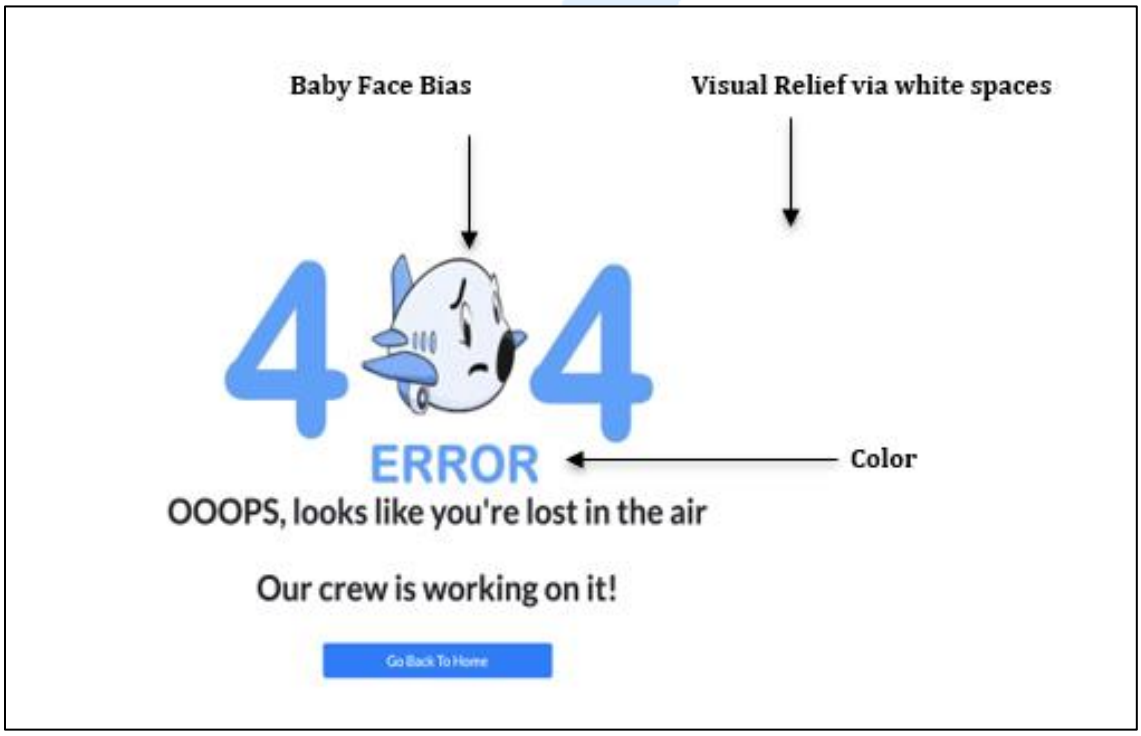


Design Elements on Home Page





Design Elements on Sign Up Page



Design Elements on the Error Page

Updates since Prototype

Database Tables

Added an admin button. Using this button, the user can see all the database tables and the real-time changes made into the database.

My Trips (Retrieving Trips from the Database)

Users can view all the upcoming trips that were added. For every upcoming trip, depending upon the nature of the trip (national/international), there is a checklist associated with it which can be viewed by the user.

Add Trip (Adding Trips from the Database)

Users can add trip details to the database.

Past Trips (Retrieving Trips from the Database)

Users can view all the past trips that were added through the website.

Pro-tips Page

We have added a pro-tips page so that first-time users can have some important information handy. This can be visited by all the users.

Database Design

For our application, we decided to implement our relational database using Amazon Relational Database Service (RDS). Every time a user creates an account, the data is stored in the Traveler Record table, when the user creates a trip, the Itinerary table then stores the information entered by the user as well as information from the Flights table. Every time a user's checklist is modified, the correspondence data in the Itinerary table changes too. Some of the data from the Flights table are created manually by the developers for now, but it can also be loaded from sources like flight information from different airlines.

Traveler Records

When a person is traveling with Zoom, they are required to sign up with the Zoom Travel for their travel. The user needs to enter some basic information such as Name, Email address, Date of Birth, Gender, Passport Number, and Country of Residence. This information associated with the travel will be stored in the traveler_record table.

Traveller Record		
PK	Customer_ID	VARCHAR(20)
	Name	VARCHAR(50)
	Email	VARCHAR(60)
	Password	VARCHAR(50)
	DOB	DATE
	Passport Number	VARCHAR(10)
	Country of Residence	VARCHAR(50)

Traveler Record Table

Traveler's Travel Information

For seamless navigation for travelers on the day of their travel, ZT is required to store the travel information of all their upcoming and past travels. Once in the system, a person can provide their travel details like International or Domestic travel, Date of Travel, Origin, Destination. The form will then look up this information from the database and display flight time and flight number automatically. This information provided is recorded in the Flights table.

Travelers sometimes make some changes to their travel plans. Any cancellations of travels, the record will be deleted from the main table and would be stored in another table. Any updates on the travel, necessary changes will be reflected in the main database table.

Flights		
PK	flight_num	VARCHAR(10)
	orig_code	VARCHAR(5)
	dest_code	VARCHAR(5)
	origin	varchar(25)
	destination	varchar(25)
	departure_time	time
	arrival_time	time
	departure_gate	varchar(5)
	arrival_gate	varchar(5)
	flight_duration	float
	departure_term	varchar(5)
	arrival_term	varchar(5)

Flights Table

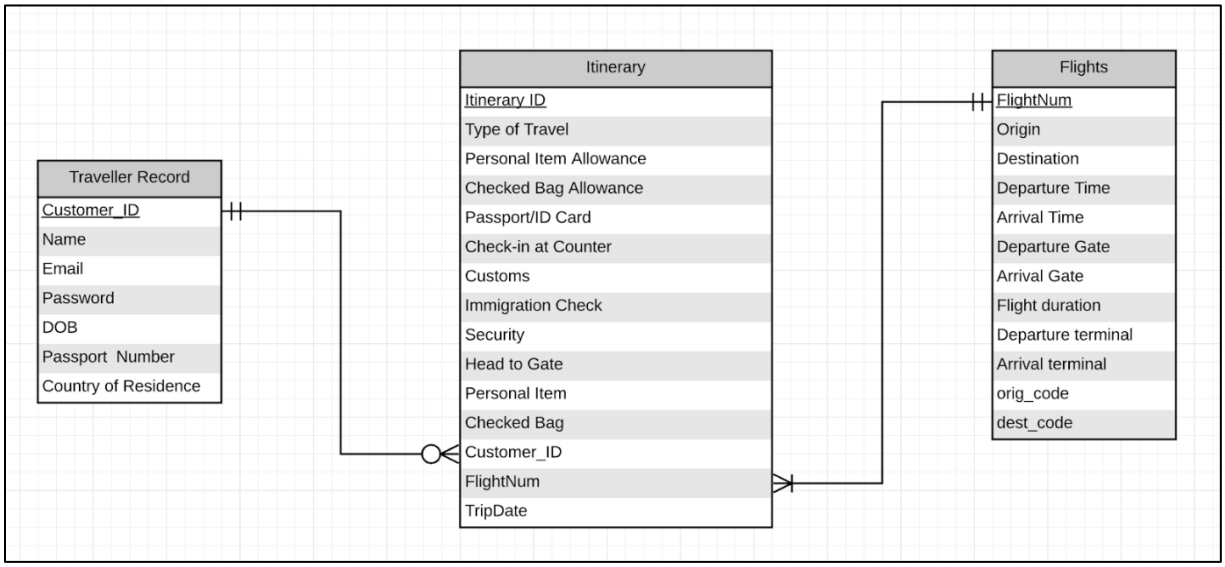
Travel Checklist service

ZT's main motive is to provide seamless navigation on the day of the flight. This website will collect the required details of the flight when he/she logs in and provide a checklist of items on the day of travel. The traveler will be provided exact steps that he/she needs to follow during their entire flight journey

Itinerary		
PK	itinerary_id	varchar(20)
	type_of_travel	char(1)
	personal_item_allw	int
	checked_bag_allw	int
	is_id	char(1)
	is_checkin_counter	char(1)
	is_customs	char(1)
	is_immigration	char(1)
	is_security	char(1)
	is_head_to_gate	char(1)
	is_personal_item	char(1)
	is_checked_bag	char(1)
FK	customer_id	VARCHAR(20)
FK	flight_num	varchar(10)
	TripDate	date

Itinerary Table

Entity Relationship Diagram



Entity Relationship Diagram

Records from the tables

Below are the screenshots of the records for each table:

```
20 •  select * from itinerary;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

itinerary_id	type_of_travel	personal_item_allw	checked_bag_allw	is_id	is_checkin_counter	is_customs	is_immigration	is_security
1	International	0	0	F	F	F	F	F
2	Domestic	1	0	T	T	F	F	T
3	International	1	2	F	F	F	F	F
4	Domestic	1	0	F	F	F	F	F
5	Domestic	1	0	F	F	F	F	F

Records for the Itinerary Table

```
20 •  select * from flights;
21
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

flight_num	orig_code	dest_code	origin	destination	departure_time	arrival_time	departure_gate	arrival_gate
B1012	HOU	BLR	Houston	Bangalore	15:00:00	02:30:00	F26	W7
B1016	HOU	DEL	Houston	Delhi	14:00:00	02:00:00	G56	N9
D1111	CLL	HOU	College Station	Houston	07:00:00	08:00:00	A2	D6
D1114	CLL	AUS	College Station	Austin	06:00:00	07:30:00	A2	F8
D1118	CLL	DAL	College Station	Dallas	11:00:00	12:00:00	A2	T8
D1200	LAS	HOU	Las Vegas	Houston	13:00:00	16:30:00	M8	S6
D1202	DAL	LAS	Dallas	Las Vegas	06:20:00	08:50:00	T6	G11
D1203	LAS	DAL	Las Vegas	Dallas	05:30:00	10:00:00	F5	J8
D1207	LAS	AUS	Las Vegas	Austin	03:30:00	07:00:00	M7	S4
D1949	DAL	SAT	Dallas	San Antonio	06:00:00	07:05:00	B87	E10

Records for the Flights Table

```
20 •  select * from traveler_record;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	name	email	gender	DOB	password	passport_num	country_of_res
1	Sneha Bhat	b123@gmail.com	Female	1993-02-22	abcd	M1234	India
2	Gaurav Agrawal	gaurav@gmail.com	Male	0002-02-22	gaurav	123456	Australia
3	mary	abc@gmail.com	NULL	NULL	asd	NULL	NULL
4	Shilpa Chanshetti	shilpa@gmail.com	Female	2018-03-01	shilpa	None	India
5	shilpa c	shilpa1@gmail.com	NULL	NULL	shilpa123	NULL	NULL
6	Shilpa	shilpac@gmail.com	Female	1994-11-11	team13	None	India
7	Arpita Deshmukh	arpitapd@gmail.com	NULL	NULL	arpita123	NULL	NULL
8	KeerthanTanhaji	kbantwal@gmail.com	NULL	NULL	weloveshilpa	NULL	NULL
9	Arpita	arpitapd@tamu.edu	NULL	NULL	arpita123	NULL	NULL
10	Santholli Sneha Bhat	abc@zyx.com	NULL	NULL	iiii	NULL	NULL
11	qqqqqq	s@s.com	NULL	NULL	1234	NULL	NULL

Records for the Traveler Record Table

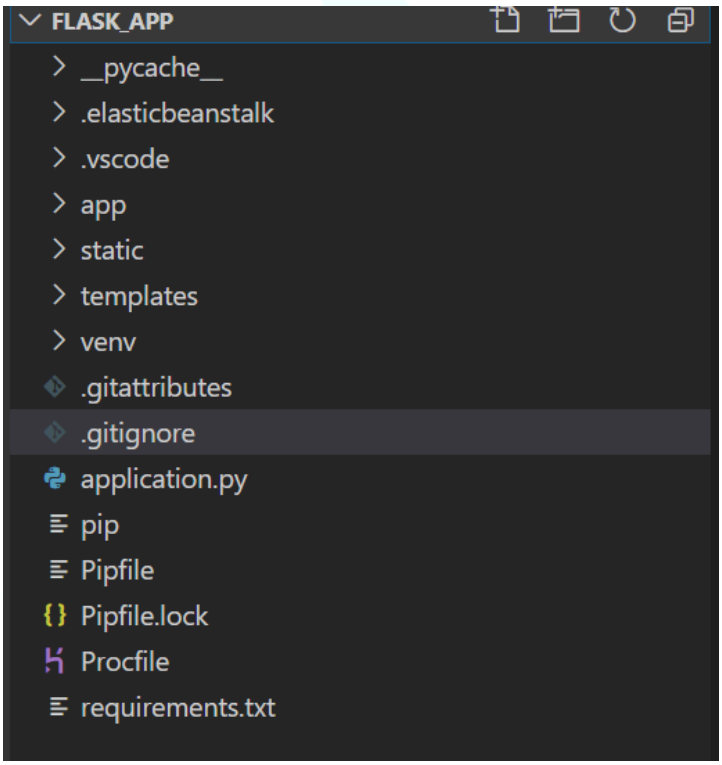
Code Implementation

For the development of our code, we have used Python Flask as our backend technology. Python Flask is a framework in Python that helps with URL routing and page rendering. With the help of Flask, we are connecting our Database, which is a MySQL Database, front-end HTML pages. Using Flask, we have developed certain APIs or routes that help us in the transition from one HTML page to another.

Our front-end pages have been developed using Bootstrap 4.3.1 extensively. We have also incorporated our HTML, CSS for any other styling purpose. We have also used jQuery and AJAX to perform some of our form validations.

Backend structure

The folder structure of our project in flask looks like the following:



Python Flask Folder structure

As we can see in the folder structure mentioned above, we have created a virtual environment on which flask application runs (venv). The templates folder stores all the HTML templates and pages developed for our website. The static folder stores the CSS files for templates, images, and other related files required to run the templates seamlessly.

The main file in the folder structure is the **application.py.** This file is used to create all the URL routes and APIs required for our application.

In application.py, we start by importing the Flask framework and other related frameworks to run the application. Post the import, our code mentions the database connection details of our application. Our database is hosted on AWS and tables on AWS are created using AWS RDS.

```
from flask import Flask, render_template,request, url_for, redirect, session, jsonify
from datetime import datetime, timedelta, date
from flask_mysqldb import MySQL
import MySQLdb.cursors
import datetime
import re
application = Flask(__name__)

# Secret Key
application.secret_key = 'ZoomTravel'

# Database connection details below      You, a few seconds ago • Uncommitted changes
application.config['MYSQL_HOST'] = 'flaskapp.crmt6c0dobbu.us-east-2.rds.amazonaws.com'
application.config['MYSQL_USER'] = 'team13'
application.config['MYSQL_PASSWORD'] = 'team132020'
application.config['MYSQL_PORT'] = 3306
application.config['MYSQL_DB'] = 'flaskapp'

# Intialize MySQL
mysql = MySQL(application)
```

application.py import and database connection details

In the above figure, the connection details of the database which is hosted on AWS is mentioned in connection.

To run the application.py, we use the following statements:

```
if __name__ == '__main__':

    application.run()
```

application.py run statement

API's or URL ROUTES

To transition from one page to another, we have created certain routes that help us perform some URL routing from one page to another. The routes that we have created are:

1. **index()**: This method routes to the first page(index.html) when the application is opened on a web browser. The following screenshot provides the information on the route.

```python
@application.route('/')
def index():
    return render_template('index.html')
```

index() route

2. **aboutUs(), contactUs(), page_not_found(e):** The aboutUs() method routes to the about us page of our application, contactUs() method routes to the contact page and page_not_found() navigated to 404 page if any error in the URL or page requested is not available.

```python
#aboutUs route
@application.route('/about')
def aboutUs():
    return render_template('about-us.html')

#contactUs route
@application.route('/contactUs')
def contactUs():
    return render_template('about-us.html', msg="contactUs")

#404 error route
@application.errorhandler(404)
def page_not_found(e):
    #expliciting setting 404      You, a few seconds ago • U
    return render_template('error.html'), 404
```

aboutUs, contactUs and 404 error Code

3. **login_form() and signup_form():** These routes are called when the sign-in or login button is clicked from index.html. We have a single HTML file for our login and sign-up but are different tabs. Therefore, there are 2 different routes with type (Sign-in or Sign-up)

```python
@application.route('/signin', methods=['GET', 'POST'])
def login_form():
    if request.method == 'POST':
        #redirect to index if method is post or any error
        return redirect(url_for('index'))

    # show the login form with signin type
    return render_template('Login.html',sType="signIn")

@application.route('/signup', methods=['GET', 'POST'])
def signup_form():
    if request.method == 'POST':
        #redirect to index if method is post or any error
        return redirect(url_for('index'))

    # show the login form with signup type
    return render_template('Login.html',sType="signUp")
```

login_form() and signup_form()

4. **signin_auth(), admin_auth() and signup_save():** These routes provide relevant functionalities when the user is trying to sign in, signup and admin. While the user tries to sign-up, the API first checks if the user has already signed up using the same email previously, else signup succeeds. Similarly, in signin, the API first checks if any user exists with the email entered by the user. If the user exists, the signin succeeds else fails. In admin auth, the admin user will be redirected to the admin page without any authentication.

```python
@application.route('/signin/', methods=['GET', 'POST'])
def signin_auth():
    msg = ''
    msg_type = ''
    class_type = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        # Variables to extract form data
        email = request.form['email']
        password = request.form['password']
        # Check if account exists using MySQL
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM traveler_record WHERE email = %s AND password = %s', (email, password,))
        # Fetch one record and return result
        account = cursor.fetchone()
        # If account exists in accounts table in out database
        if account:
            # Create session data, we can access this data in other routes
            session['loggedin'] = True
            session['id'] = account['email']
            session['email'] = account['email']
            # User is loggedin show them the home page
            return redirect(url_for('home', account=account['name']))      chioujiin, 18 days ago • resovle
        else:
            # Account doesnt exist or username/password incorrect
            msg = 'Hmmm, the information you entered does not match our records. Please try again!'
            msg_type = 'Error'
            class_type = 'sadFlappy'
    return render_template('Login.html', msg=msg, msg_type=msg_type,sType="signIn", class_type=class_type)
```

Signin_auth()

```python
@application.route('/admin/', methods=['GET', 'POST'])
def admin_auth():      You, a day ago • Changes for Final project
    msg = ''
    msg_type = ''
    class_type = ''
    print("yes")
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELECT * FROM traveler_record WHERE email = "admin@tamu.edu" AND password = "admin" ')
    # Fetch one record and return result
    account = cursor.fetchone()
    print(account['email'])
    # If account exists in accounts table in out database
    if account:
        # Create session data, we can access this data in other routes
        session['loggedin'] = True
        session['id'] = account['email']
        session['email'] = account['email']
        #fetch flight data
        cursor.execute('SELECT * FROM flights')
        flights = cursor.fetchall()
        #fetch traveler record data
        cursor.execute('SELECT * FROM traveler_record')
        travelRecords = cursor.fetchall()
        #fetch itinerary data
        cursor.execute('SELECT * FROM itinerary')
        itinerary = cursor.fetchall()
        # User is loggedin show them the home page
        return render_template('admin.html', flights=flights, travelRecords=travelRecords, itinerary=itinerary)
    return render_template('Login.html', msg=msg, msg_type=msg_type,sType="signIn", class_type=class_type)
```

admin_auth()

```python
@application.route('/signup/', methods=['GET', 'POST'])
def signup_save():
    msg = ''
    msg_type = ''
    class_type = ''
    # Check if "username", "password" and "email" POST requests exist (user submitted form)
    if request.method == 'POST' and 'name' in request.form and 'password' in request.form and 'email' in request.form
        name = request.form['name']
        password = request.form['password']
        email = request.form['email']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM traveler_record WHERE email = %s', (email,))
        account = cursor.fetchone()
        if account:
            msg = 'Hmm, I think you already are a Zoomer. Please try logging in again!! '
            msg_type = 'Error'
            class_type = 'sadFlappy'      chioujiin, 18 days ago • resovle
        elif not name or not password or not email:
            msg = 'Looks like the sky is empty! Please fill out the form!'
            msg_type = 'Error'
            class_type = 'sadFlappy'
        else:
            cursor.execute('INSERT INTO traveler_record (name, email, password) VALUES (%s, %s, %s)', (name, email, p
            mysql.connection.commit()
            msg = 'Yippe!! You just subscribed for an easy travel. You are all set to fly high'
            msg_type = 'Success'
            class_type = 'happyFlappy'
    elif request.method == 'POST':
        # Form is empty... (no POST data)
        msg = 'Please fill out the form!'
        msg_type = 'Error'
        class_type = 'sadFlappy'
    # Show registration form with message (if any)
    return render_template('Login.html', msg=msg, msg_type=msg_type, class_type=class_type)
```

signup_save

5. **home() and logout():** These APIs allow navigating to the home page from any other page on our website. Similarly, once the user is logged in, he can log out and the logout() API is called in the process.

```
@application.route('/signin/home')
def home():
    # Check if user is loggedin
    if 'loggedin' in session:
        # User is loggedin show them the home page
        return render_template('home.html', account=request.args.get('account'))
    # User is not loggedin redirect to index page
    return redirect(url_for('index'))

@application.route('/logout')
def logout():
    # Remove session data
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('email', None)
    # Redirect to index page
    return redirect(url_for('index'))
```

home() and logout()

6. **profile_form() and profile_submit(): These** are profile form related APIs. The profile_form opens up when the user clicks on the Profile button after logging into the application. Once, the profile form is submitted, profile submit performs validations and checks and updates the traveler record table.

```
@application.route('/profile', methods=['GET', 'POST'])
def profile_form():
    if 'loggedin' in session:
    # We need all the account info for the user so we can display it on the profile page
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM traveler_record WHERE email = %s', (session['id'],))
        record = cursor.fetchone()
        # Show the profile page with account info
        return render_template('profile.html', record=record)
    # User is not loggedin redirect to login page
    return redirect(url_for('login_form'))
```

Profile_form()

```python
@application.route('/profile/', methods=['GET', 'POST'])
def profile_submit():
    msg = ''
    msg_type = ''
    class_type = ''
    if request.method == 'POST' and ('name' in request.form and request.form['name']!='')  and ('email' in request.form and request.form['email']!=
        #defined variables
        name = request.form['name']
        email = request.form['email']
        dob = request.form['dob']
        formatted_date = datetime.datetime.strptime(dob, "%Y-%m-%d")
        gender = request.form['gender']
        passport = request.form['passport']
        country = request.form['country']
        dt = datetime.date.today()
        today = datetime.datetime.combine(dt, datetime.time())
        if (formatted_date < today):      You, a day ago • Changes for Final project
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('UPDATE traveler_record SET name = %s, gender = %s, DOB = %s, passport_num = %s, country_of_res = %s where email = %s',
            mysql.connection.commit()
            msg = 'Yay! Profile is updated'
            msg_type = 'Success'
            class_type = 'happyFlappy'
        else:
            msg = 'I see that the date you entered is incorrect. Please enter past date'
            msg_type = 'Error'
            class_type = 'sadFlappy'
    elif request.method == 'POST':
        # Form is empty... (no POST data)
        msg = 'Uh oh! Please fill out the form!'
        msg_type = 'Error'
        class_type = 'sadFlappy'
    # Show registration form with message (if any)
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELECT * FROM traveler_record WHERE email = %s', (session['id'],))
    record = cursor.fetchone()
    # Show the profile page with account info
    return render_template('profile.html', record=record, msg=msg, msg_type=msg_type, class_type=class_type)
```

profile_submit()

7. **addTrips(), populateOrig(), populateDest(), populateFlightDetails():** The addTrips() route is called when a user is adding a new trip. The add trip form has few validations
   a. The first validation is to check with user if the trip type is International or Domestic
   b. Based on Trip type, populateOrig() makes a call to DB to fetch all the origin
   c. Based on the origination selected, the destinations are showed using popluateDest()
   d. The rest of the flight details like Flight num and Departure time is populated using populateFlightDetails()

```python
@application.route('/addTrips', methods=['GET', 'POST'])
def addTrips():
    if 'loggedin' in session:
        # Show the profile page with account info
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM traveler_record WHERE email = %s', (session['id'],))
        record = cursor.fetchone()
        # Show the profile page with account info
        return render_template('addTrips.html', record=record)
    # User is not loggedin redirect to login page
    return redirect(url_for('login_form'))
```

addTrips()

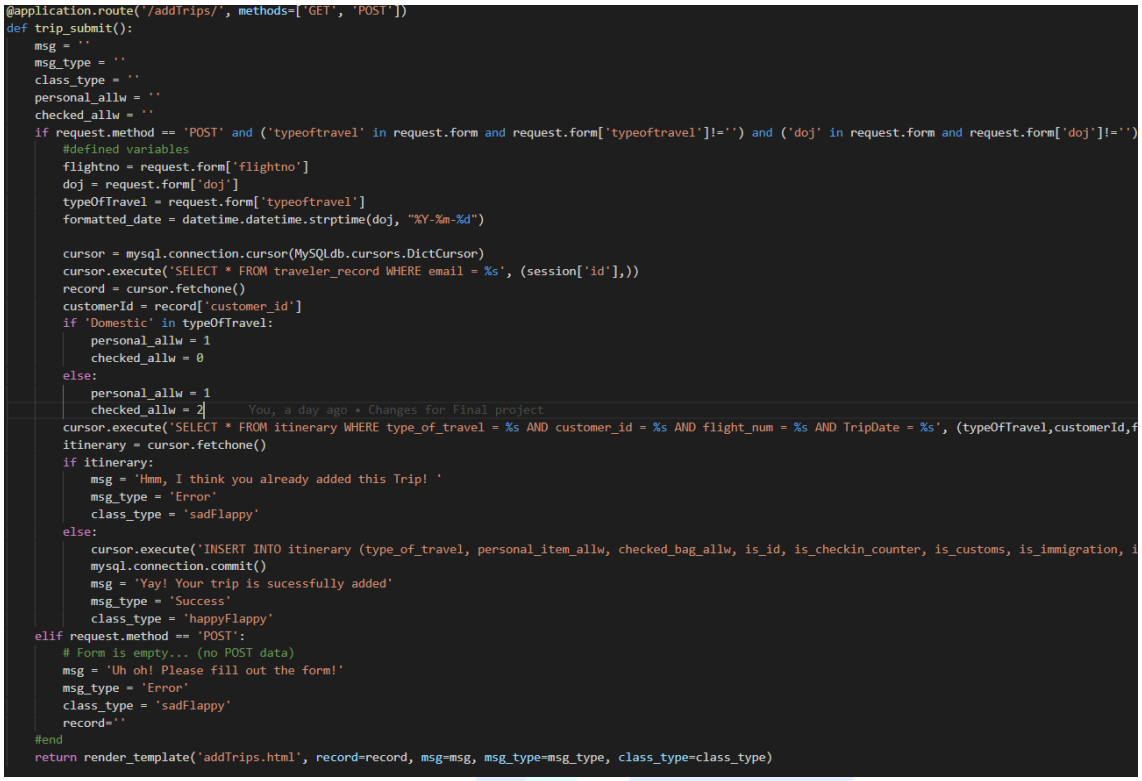
```python
@application.route('/getOrigin', methods=['GET', 'POST'])
def populateOrig():
    if 'loggedin' in session:
        typeoftravel=request.form['travelType']
        print(typeoftravel)
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute("SELECT distinct origin FROM flights where travel_type=%s", (typeoftravel,))
        origins = cursor.fetchall()
        return jsonify(origins)
    return redirect(url_for('login_form'))

@application.route('/getDest', methods=['GET', 'POST'])
def populateDest():
    if 'loggedin' in session:
        typeoftravel=request.form['travelType']
        origin = request.form['origin']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute("SELECT distinct destination FROM flights where travel_type=%s AND origin= %s", (typeoftravel,origin,))
        destinations = cursor.fetchall()
        #return jsonify({'name':records[0]['origin']})
        return jsonify(destinations)
    return redirect(url_for('login_form'))

@application.route('/getFlightDetails', methods=['GET', 'POST'])
def populateFlightDetails():
    if 'loggedin' in session:      You, a day ago • Changes for Final project
        origin = request.form['origin']
        destination = request.form['destination']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute("SELECT flight_num, date_format(departure_time, '%%h:%%i') as departure_time FROM flights where  origin= %s AND destination
        flightDetails = cursor.fetchone()
        return jsonify(flightDetails)
    return redirect(url_for('login_form'))
```

populateOrig(), populateDest(), populateFlightDetails()

8. **Trip_submit():** This is one of the main functionality API in our project. The trip_submit() method validates when a user tries to add a trip to the database.

   Few validations are:

   a. When a user adds a trip, the API checks if the user has added the same record before. If a trip exists, then it prompts an error.
   b. The user adds the trip (past and upcoming). The past trips act as a history of trips for the users. The upcoming trip is the trip that the user travels on a later date. On successful submission, a new record is inserted in the itinerary table with all the checklist attributes set to 'F' (false)



```python
@application.route('/addTrips/', methods=['GET', 'POST'])
def trip_submit():
    msg = ''
    msg_type = ''
    class_type = ''
    personal_allw = ''
    checked_allw = ''
    if request.method == 'POST' and ('typeoftravel' in request.form and request.form['typeoftravel']!='') and ('doj' in request.form and request.form['doj']!=''):
        #defined variables
        flightno = request.form['flightno']
        doj = request.form['doj']
        typeOfTravel = request.form['typeoftravel']
        formatted_date = datetime.datetime.strptime(doj, "%Y-%m-%d")

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM traveler_record WHERE email = %s', (session['id'],))
        record = cursor.fetchone()
        customerId = record['customer_id']
        if 'Domestic' in typeOfTravel:
            personal_allw = 1
            checked_allw = 0
        else:
            personal_allw = 1
            checked_allw = 2      You, a day ago • Changes for Final project
        cursor.execute('SELECT * FROM itinerary WHERE type_of_travel = %s AND customer_id = %s AND flight_num = %s AND TripDate = %s', (typeOfTravel,customerId,f
        itinerary = cursor.fetchone()
        if itinerary:
            msg = 'Hmm, I think you already added this Trip! '
            msg_type = 'Error'
            class_type = 'sadFlappy'
        else:
            cursor.execute('INSERT INTO itinerary (type_of_travel, personal_item_allw, checked_bag_allw, is_id, is_checkin_counter, is_customs, is_immigration, i
            mysql.connection.commit()
            msg = 'Yay! Your trip is sucessfully added'
            msg_type = 'Success'
            class_type = 'happyFlappy'
    elif request.method == 'POST':
        # Form is empty... (no POST data)
        msg = 'Uh oh! Please fill out the form!'
        msg_type = 'Error'
        class_type = 'sadFlappy'
        record=''
    #end
    return render_template('addTrips.html', record=record, msg=msg, msg_type=msg_type, class_type=class_type)
```

Trip_submit()

9. **Complete_checklist() and checklist_submit():** Once the trip is added, the trips can be seen in the upcoming trips page. Each trip is showed as a card with all details and the 'Complete Checklist' button. The button navigates to checklist forms (Domestic or International) to be completed by the user. Once on the checklist form, the user can follow the checklist and submit to populate data to the database.

```python
@application.route('/checklist', methods=['GET', 'POST'])
def complete_checklist():
    # Check if user is loggedin
    if 'loggedin' in session:
        # User is loggedin show them the home page
        print(request.args.get('itineraryId'))
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM traveler_record WHERE email = %s', (session['id'],))
        record = cursor.fetchone()
        if 'International' in request.args.get('travelType'):
            return render_template('InternationalChecklist.html', travelType=request.args.get('travelType'), itineraryId=request.args.get('itinerary
        else:
            return render_template('DomesticChecklist.html', travelType=request.args.get('travelType'),itineraryId=request.args.get('itineraryId'),
    # User is not loggedin redirect to index page
    return redirect(url_for('login_form'))
```

complete_checklist()

```python
@application.route('/checklistSubmit', methods=['GET', 'POST'])
def checklist_submit():
    # Check if user is loggedin
    if request.method == 'POST':
        itineraryId = request.args.get('itineraryId')
        header = 'Upcoming Trips'
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        if request.args.get('travelType') == 'International':
            print('inside')
            isId = request.form.get("item-2")
            isCheckinCounter = request.form.get('item-4')
            isCustoms = request.form.get('item-7')
            isImmigration = request.form.get('item-9')
            isSecurity= request.form.get('item-10')
            isHeadToGate = request.form.get('item-12')
            isPersonalItem = request.form.get('item-16')
            isCheckedBag= request.form.get('item-7')

            isCheckinCounter = 'T'
            isImmigration = 'T'
            isSecurity = 'T'
            isHeadToGate = 'T'
            isPersonalItem = 'T'

            if isId == 'on':
                isId = 'T'
            else:
                isId = 'F'
        You, a day ago • Changes for Final project
            if isCustoms == 'on':
                isCustoms = 'T'
                isCheckedBag = 'T'
            else:
                isCustoms = 'F'
                isCheckedBag = 'F'
```

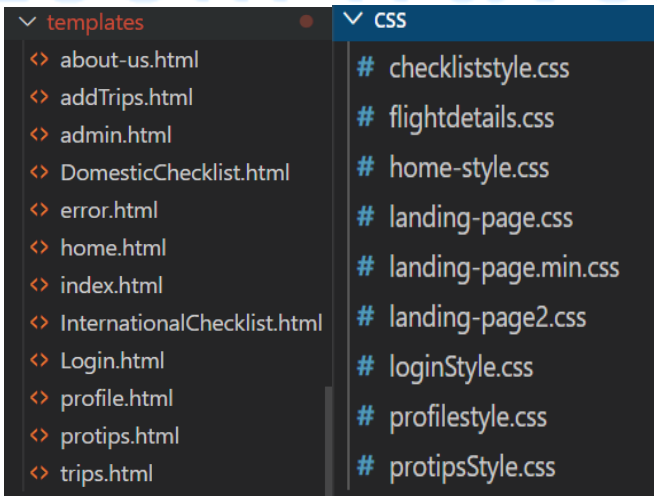
checklist_submit()

```
            cursor.execute ('UPDATE itinerary set is_id = %s, is_checkin_counter = %s , is_customs = %s, is_immigration
            mysql.connection.commit()
    else:

        isId = request.form.get("item-2")
        isCheckinCounter = request.form.get('item-4')
        isSecurity= request.form.get('item-7')
        isHeadToGate = request.form.get('item-8')
        isPersonalItem = request.form.get('item-5')
        isCheckedBag= request.form.get('item-6')

        isCheckinCounter = 'T'
        isSecurity = 'T'
        isHeadToGate = 'T'
        isPersonalItem = 'T'

        if isId == 'on':
            isId = 'T'
        else:
            isId = 'F'

        if isCheckedBag == 'on':
            isCheckedBag = 'T'
        else:
            isCheckedBag = 'F'

        cursor.execute ('UPDATE itinerary set is_id = %s, is_checkin_counter = %s , is_security= %s , is_head_to_gate=%s
        mysql.connection.commit()

    cursor.execute('SELECT * FROM traveler_record WHERE email = %s', (session['id'],))
    record = cursor.fetchone()
    customer_id = record['customer_id']
    #Get all trips
    cursor.execute('SELECT itinerary_id, customer_id, type_of_travel, f.flight_num, DATE_FORMAT(TripDate, "%%D %%M %%Y")
    flights = cursor.fetchall()
    return render_template('trips.html', record=record, flights=flights, header= header)      You, a day ago • Changes
# User is not loggedin redirect to index page
return redirect(url_for('login_form'))
```

Checklist_complete contd.

Front-end HTML

All our templates are produced mostly using bootstrap CSS and jQuery. The following structure pages are developed for our website.



Templates and CSS

We have a common header and footer across the website. The following code shows the navigation links in the header and footer. The Navigation on the website links and the mobile view it compresses to the Hamburger menu.

```
<!-- Navigation -->
<nav class="navbar navbar-expand-xl navbar-light bg-light" style="background-color: #ffffff;">
 <img class="navbar-brand logo" src="{{url_for('static', filename='img/logo2.png')}}" width="300" height="65"

 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#mobile_menu" aria-controls=
   <span class="navbar-toggler-icon"></span>
 </button>
   <div class="collapse navbar-collapse" id="mobile_menu">
    <ul class="navbar-nav ml-auto font ">
    <li class="nav-item-active"><a class= "nav-link" href="#">Home <span class="sr-only">(current)</span></a>
    <li><a class= "nav-link"  href="{{url_for('upcomingTrips')}}">Upcoming trips </a></li>
    <li><a class= "nav-link" href="{{url_for('addTrips')}}">Add trips</a></li>
    <li><a class = "nav-link" href="{{url_for('pastTrips')}}">Past trips</a></li>
    <li><a class = "nav-link" href="protips">Pro-tips</a></li>
    <ul class="navbar-nav navbar-right">
      <li><a href="{{url_for('profile_form')}}" class="btn btn-primary" role ="button" style = "margin-right:
      <li><a href="{{url_for('logout')}}" class="btn btn-primary" role ="button" style = "margin-right: 10px;
    </ul>
     </ul>
   <!-- </div> -->
 </div>
 </nav>
```

Navigation Bar across the web application

```
<!-- Footer -->
<footer class="footer bg-light" style= "font-size: 16px;">
  <div class="container">
    <div class="row">
      <div class="col-lg-4 h-100 text-center text-lg-left">
        <ul class="list-inline mb-2">
          <li class="list-inline-item">
            <a href="/about">About</a>
          </li>
          <li class="list-inline-item">&sdot;</li>
          <li class="list-inline-item">
            <a href="{{url_for('contactUs')}}">Contact</a>
          </li>      You, 18 days ago • Final prototype changes
        </ul>
        <p class="text-muted small mb-4 mb-lg-0">&copy; Developed by Team 13.</p>
      </div>
      <div class="col-lg-4 h-100 text-center text-lg-center my-auto">
        <div class="scroll-up">
          <a href="#"><i class="fa fa-angle-double-up"></i></a>
        </div>
      </div>
      <div class="col-lg-4 h-100 text-center text-lg-right my-auto">
        <ul class="list-inline mb-0">
          <li class="list-inline-item mr-3">
            <a href="#">
              <i class="fab fa-facebook fa-2x fa-fw"></i>
            </a>
          </li>
          <li class="list-inline-item mr-3">
            <a href="#">
              <i class="fab fa-twitter-square fa-2x fa-fw"></i>
            </a>
          </li>
          <li class="list-inline-item">
            <a href="#">
              <i class="fab fa-instagram fa-2x fa-fw"></i>
            </a>
          </li>
        </ul>
      </div>
    </div>
  </div>
</footer>
```

Footer code across web application

Technologies Used

Software and Tools

We are using the following tools and software for developing the web application

IDE: Visual Studio code

- Version control: Git
- Learning and Training: W3school
- Presentations: PowerPoint

Technology Stack

- Front-end: HTML, CSS, JavaScript, Bootstrap CSS
- Back-end: Python, Flask

Database

- AWS - RDS

Host Suite

- AWS Elastic Beanstalk

Project Learning Outcomes and Challenges

Learning Outcomes

- Recognized the importance of the interface design towards the success of the system
- Understood how to make a website functional, usable as well as understandable
- Combine different elements of design to enhance the overall user experience
- Understood how business requirements can be incorporated into the project

Challenges

- It took a lot of time to host the website as we decided to host the website once all the functionalities were implemented.
- Initially, it was a bit difficult to coordinate with all the team members with all the online meetings

Appendix

The code implemented is too large and has a lot of files, so it can be accessed using the below GitHub link:

https://github.com/ShilpaChanshetti/Zoom-Travel

- The templates folder stores all the HTML templates and pages developed for our website.
- The static folder stores the CSS files for templates, images, and other related files required to run the templates seamlessly.

- The main file in the folder structure is the **application.py.** This file is used to create all the URL routes and APIs required for our application.

Our Team

Arpita Deshmukh

Mary Huang

Shilpa Chanshetti

S. Sneha Bhat