CPS630 Project Iteration I - II
Technical Report

Website: Plan Your Travel

Team 25

Angela Gavin 500761490
Nick De Santo 500686917
Alison Der 500815229

	Angela Gavin	**Nick De Santo**	**Alison Der**
Database	70%	20%	10%
Home Page	5%	15%	80%
Read More	100%	-	-
Shopping Cart	-	100%	-
Search	-	100%	-
About Us	-	-	100%
Contact Us	100%		

2. Describe briefly about the project objectives, languages and tools you have used. Describe briefly about the design and implementation of your database maintain mode; which MySql commands you have used to implement this mode in order to: create, add, delete, modify, and search through database (you should add drawings or screen-shots at this part).

Project Objectives

The objective of this project was to create a travel website that is both responsive and user-friendly. It features 20 famous attractions around the world from 5 different continents and two different countries per continent. On the homepage, users are able to select certain attractions that they can view more details about dynamically. The website also implements a shopping cart page which includes two different travel plans that show a map of the attractions featured in the selected plan.

Languages

HTML: used for website content
CSS: used to style whole website
JQuery: used for animations
Javascript: used to extract data from database
PHP: used to process requests for the forms
SQL: used to create, insert, modify, etc

Tools

MaterializeCSS: used for responsive design
Bootstrap: used for responsive design
phpMyAdmin: used to execute SQL statements
Heroku (Database): used to organize data

Methodology

AJAX: used to fill each dropdown dynamically

For our database, we used MySql commands such as CREATE TABLE to create a table for our attractions:

Table	Create Table
attractions	CREATE TABLE `attractions` (`a_id` int(10) unsigned NOT NULL AUTO_INCREMENT, `title` varchar(255) CHARACTER SET utf8 NOT NULL, `date_of_creation` varchar(255) CHARACTER SET utf8 DEFAULT NULL, `founder_name` varchar(255) CHARACTER SET utf8 DEFAULT NULL, `dimensions` varchar(255) CHARACTER SET utf8 DEFAULT NULL, `location` varchar(255) CHARACTER SET utf8 DEFAULT NULL, `country` varchar(255) CHARACTER SET utf8 DEFAULT NULL, `continent` varchar(255) CHARACTER SET utf8 DEFAULT NULL, PRIMARY KEY (`a_id`)) ENGINE=InnoDB AUTO_INCREMENT=42 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci

ADD DATA

```
INSERT INTO `photos`(`p_id`, `img_path`, `a_id`) VALUES
(17,'assets/img/ch1.jpg',5),
(18,'assets/img/ch2.jpg',5),
(19,'assets/img/ch3.jpg',5),
(20,'assets/img/ch4.jpg',5),
(21,'assets/img/ch21.jpg',6),
(22,'assets/img/ch22.jpg',6),
(23,'assets/img/ch23.jpg',6),
(24,'assets/img/ch24.jpg',6)
```

DELETE

```
1 DELETE FROM `photos` WHERE img_path LIKE '%fr%';
```

MODIFY

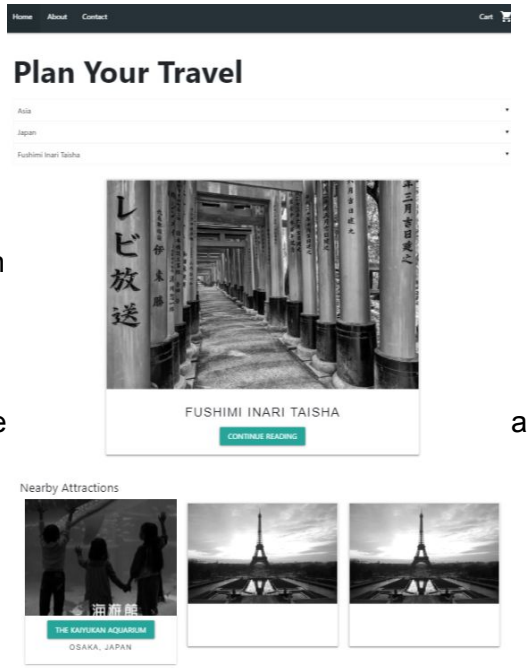
```
UPDATE `photos` SET `p_id`=4,`img_path`='assets/img/fr2.jpg' ,`a_id`= 2 WHERE a_id = 4
```

SEARCH

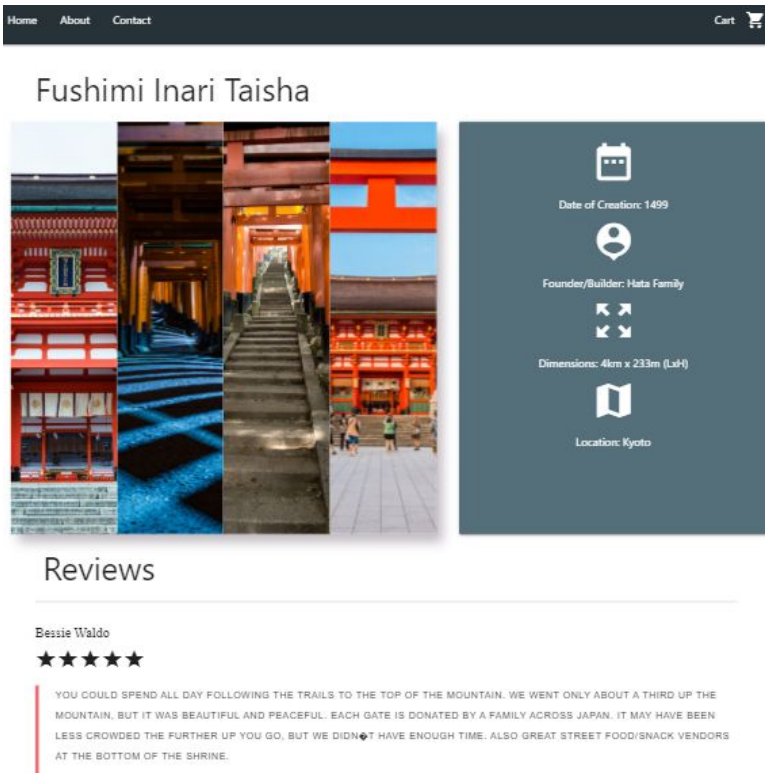
```
SELECT img_path FROM `photos` WHERE a_id = 20
```

3. Describe briefly about design and layout of your application user interfaces UIs: main page, second page and any other pages that can be opened while running application using a browser (add either drawings or screen-shots).

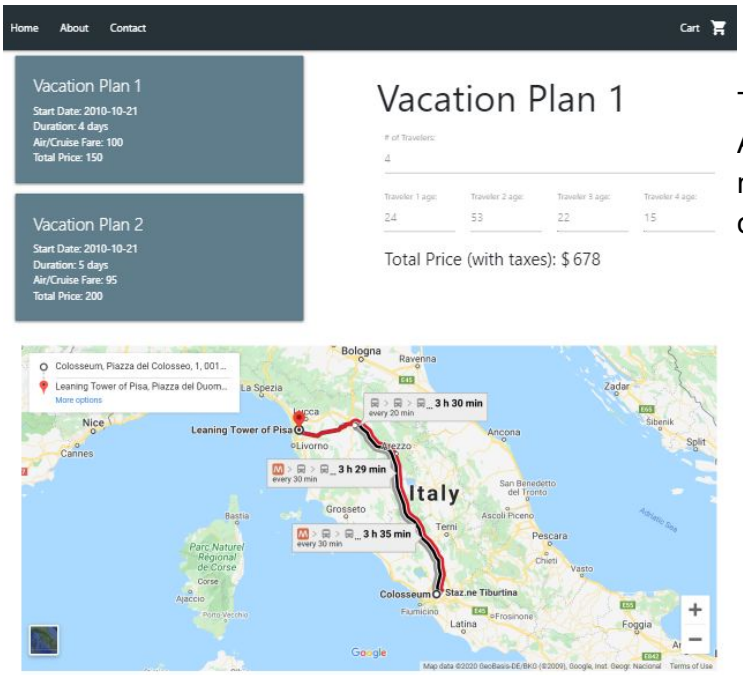
All pages implement Bootstrap and MaterializeCSS for UI design. The main page(homepage.php) features AJAX where the dropdown list of continents, countries and attractions options are filled in dynamically depending on the choice in previous dropdown. The attraction selected and its nearby attractions are hidden until the attraction dropdown has been selected for a better user experience.



Each page also includes a php code that lets us include a separate navigation header that is reusable on every page.



The attraction has a button ‘Continue Reading’ that will link it to the appropriate *readmore.php* page where it displays a gallery of attraction pictures, its details, and a few customer reviews.



The Shopping Cart also implements the AJAX methodology where it will change the map (google maps) and travel form based on the travel plan a user selects.

4. Describe briefly about the design and structure of your application Databases: name of tables, their fields, primary key (and any other keys), how each table is related to the other tables, how the tables can be accessed via their keys (you should add drawings or screen-shots at this part).Also add your database schema diagram here (similar to the diagrams in slides 57, 58, or 59 of "Lec4-3-Chap14-WorkingWithDatabases").

Attractions

a_id: each attraction's id, int(10), **primary key**
title: name of attraction, varchar(255)
date_of_creation: date of when the attraction was created, varchar(255)
founder_name: the builder or owner of the attraction, varchar(255)
dimensions: the size of the attraction, varchar(255)
location: the city that the attraction is in, varchar(255)
country: the country that the attraction is in, varchar(255)
continent: the continent the attraction is in, varchar(255)

Photos

p_id: each photos id, int(10), **primary key**
img_path: a string of the image's path in the project folder, varchar(255)
a_id: the related attraction id, int(10), **foreign key** referencing the a_id in the *attractions* table

Reviews

r_id: each reviews id, int(10), **primary key**
name: the reviewer's name, varchar(60),

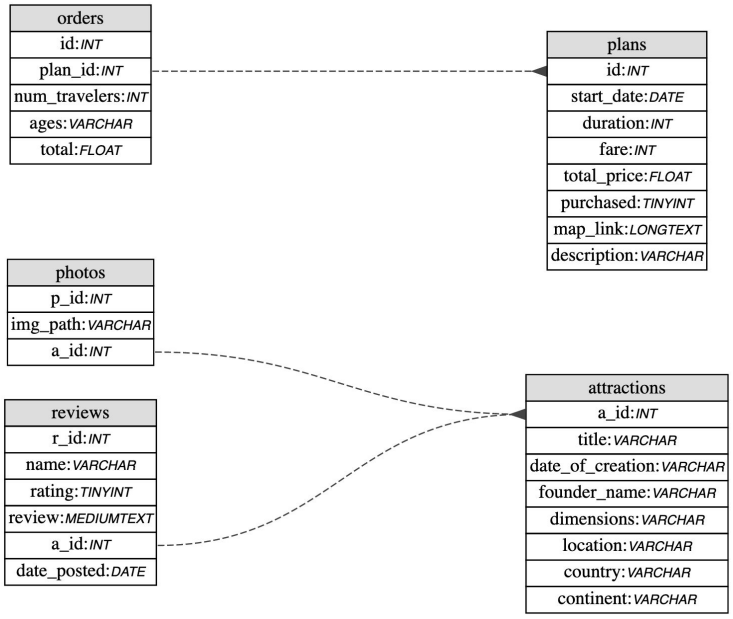
rating: a number 1-5 of a reviewer's rating of the attraction, tinyint(1)
review: the description of the review of the reviewer, mediumtext
a_id: the related attraction id, int(10), **foreign key** referencing the a_id in the attractions table

Orders

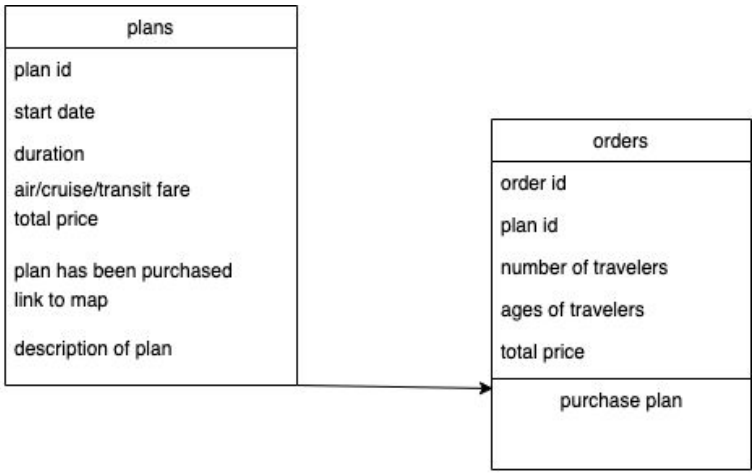
id: each orders id, **primary key**, INT(11)
plan_id: the related plan id, **foreign key** for plans id, INT(11)
num_travelers: number of travelers for the plan, INT(11)
ages: the ages of the travelers, VARCHAR(255)
total: the total price of the trip (plan total * number of travelers) + tax, FLOAT

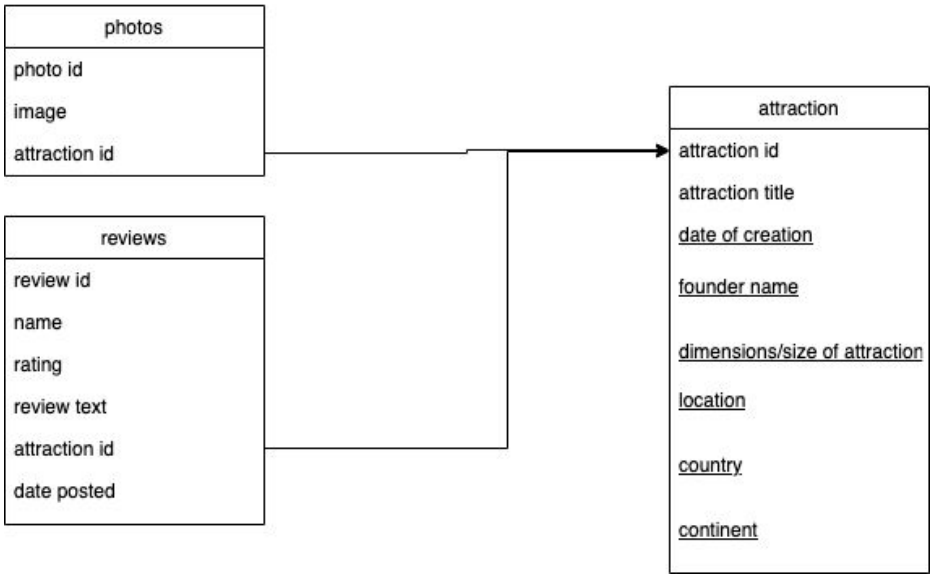
Plans

id: each travel plans id, **primary key**, INT(11)
start_date: the beginning date of the travel plan (yyyy-mm-dd), DATE
duration: number of days the plan is, INT(11)
fare: the price for one travelers air/cruise/transit fare, INT(11)
total_price: the price of the air/cruise/transit fare + any other fees of the plan, FLOAT
Purchased: if the plan has been purchased by user (currently one user, will need to be updated on a per user basis), TINYINT(1)
map_link: the link to the google map of the area with pin drops of the attractions, LONGTEXT
description: the description of the plan, VARCHAR(255)



5. Draw the whole architecture of your application through a class diagram including: the classes you identified at your design, and how these classes are related to each other. In this diagram, also indicate how MVC pattern is applied on your application.





The classes created in the design include **attractions, reviews, photos, plans, and orders.** The classes **photos** and **reviews** are related to **attraction**. Every **attraction** has photos and reviews associated with the specific attraction, this is through the shared key '**attraction id**'.

The class **plans** are related to **orders**, as a user must purchase **a plan** which then **creates an order** with the inputted values by the user.

Plans are contextually related to **attractions**, in that the attractions and locations are used within the plans, but there is no internal link through the database or code as it is not yet needed.

The **MVC** pattern is applied in our program as such:

Model -> The database structures of the program, which were created through PHP.

View -> The pages that display the info that can be seen within the database through forms and actions by the controllers (and other front-end pages that may not relate to the database)

Controller -> The requests on the database, through forms on the view, that get the data from the database (model). The middle man. Every server request to get data is through the separate controllers of the applications (which is what the classes are modeling).