



CMP 416 – Internet and Network Computing

Dr. Gerassimos Barlas

Project Report

Mohammad Asif Towheed – 66375

The Proposal

The Application:

The project will involve creating a website serving as a platform for freelancing photographers. Photographers as well as potential customers will be required to register to become a member. Once a member, they would have to log in every time in order to use the website's services. Thus, we will have two sets of web pages:

- i) A photographer's website
- ii) A customer's website

A slightly more detailed idea about the set of services offered by each of the different user's website is given below.

Photographer's website:

In general, the photographer will be able to create and update his 'portfolio'. This will include their essential details such as **name**, **DOB**, **nationality** (*if required*), **address**, **education**, **experience** and **service charges**. In addition, the photographer can upload **albums** demonstrating his sample photos (*for example, an album for pictures he captured at a wedding and another album for pictures he captured on a trip to Athens*). Furthermore, their website will also allow the photographer to view a **list** of all the customers they have had in the past, and their personal **history** with those customers (*number of times served, what the project was, et.al*). Lastly, if contacted by a customer for a **reservation**, the photographer may choose to accept or decline the customer's request (*the reservation would mainly include the address, date(s), time and a proposed payment amount – acceptance and rejection of the reservation is solely dependent on the photographer's personal appeal towards the project and is not subject to any other factors*). If the photographer accepts the reservation, the project is added to their history with that particular customer.

Customer's website:

The customer's website will be different from the photographer's website. The customer will have a 'profile' including their essential details too, such as **name**, **DOB**, **nationality** (*if required*), **address**, et. al. The customer will be able to browse for photographers. They can either view the entire **list** of photographers, or filter their search based on the photographer's name, address, service charge range, etc. The customers can also view the photographer's details, their albums, and their personal **history** with the given photographer (*if they have worked together earlier*). Finally, the customer will be able to contact the photographer and make **reservations** with the photographer for a given date(s) and propose a payment amount. If accepted by the photographer, the given project will be added to the history.

P.S.: The entire application is intended to work as a single *web application*. When using the term 'website' for the two types of users (*photographers and customers*), I am referring to the different sets of web pages generated for each of them.

DB Schema

Synopsis:

Photographer:

- Id (*integer*) – *auto-generated*
- Username (*varchar*) *unique*
- Password (*varchar*) *md5 hash*
- First Name (*varchar*)
- Last Name (*varchar*)
- Date of Birth (*date*)
- Nationality (*varchar*)
- Address (*varchar*)
- Education High school (*boolean*)
- Education Bachelor's (*boolean*)
- Education Master's (*boolean*)
- Experience (*varchar*)
- Service charges (*double*)

Customer:

- Id (*integer*) – *auto-generated*
- Username (*varchar*) *unique*
- Password (*varchar*) *md5 hash*
- First Name (*varchar*)
- Last Name (*varchar*)
- Date of Birth (*date*)
- Nationality (*varchar*)
- Address (*varchar*)

Album:

- Id (*integer*) – *auto-generated*
- Album name (*varchar*)
- Photographer Id (*integer*) *Foreign Key: Photographer (Id)*

Image:

- Id (*integer*) – *auto-generated*
- File name (*varchar*)
- Directory Path (*varchar*)

- Album Id (*integer*) *Foreign Key: Album (Id)*

Reservation:

- Id (*integer*) – *auto-generated*
- Customer id (*integer*) *Foreign Key: Customer (Id)*
- Photographer id (*integer*) *Foreign Key: Photographer (Id)*
- Date (*date*)
- Time (*time*)
- Venue (*varchar*)
- Proposed amount (*double*)
- Status (*varchar*) – [*‘requested’*; *‘accepted’*; *‘declined’*]

We do not need another table for storing the history between a certain photographer and customer. The ‘reservation’ table is suitable enough for this purpose. At the end, every reservation created will be some sort of history between a customer and a photographer.

photofreelancing.sql

```
drop table reservation;
drop table image;
drop table album;
drop table photographer;
drop table customer;

create table photographer (
    ID int primary key generated always as identity,
    Username varchar (30) NOT NULL UNIQUE,
    Password varchar (100) NOT NULL,
    Fname varchar (30) NOT NULL,
    Lname varchar (30) NOT NULL,
    DOB date,
    Nationality varchar (20),
    Address varchar (200),
    EduHS boolean,
    EduBS boolean,
    EduMS boolean,
    Experience varchar (500),
    Rate double)
;

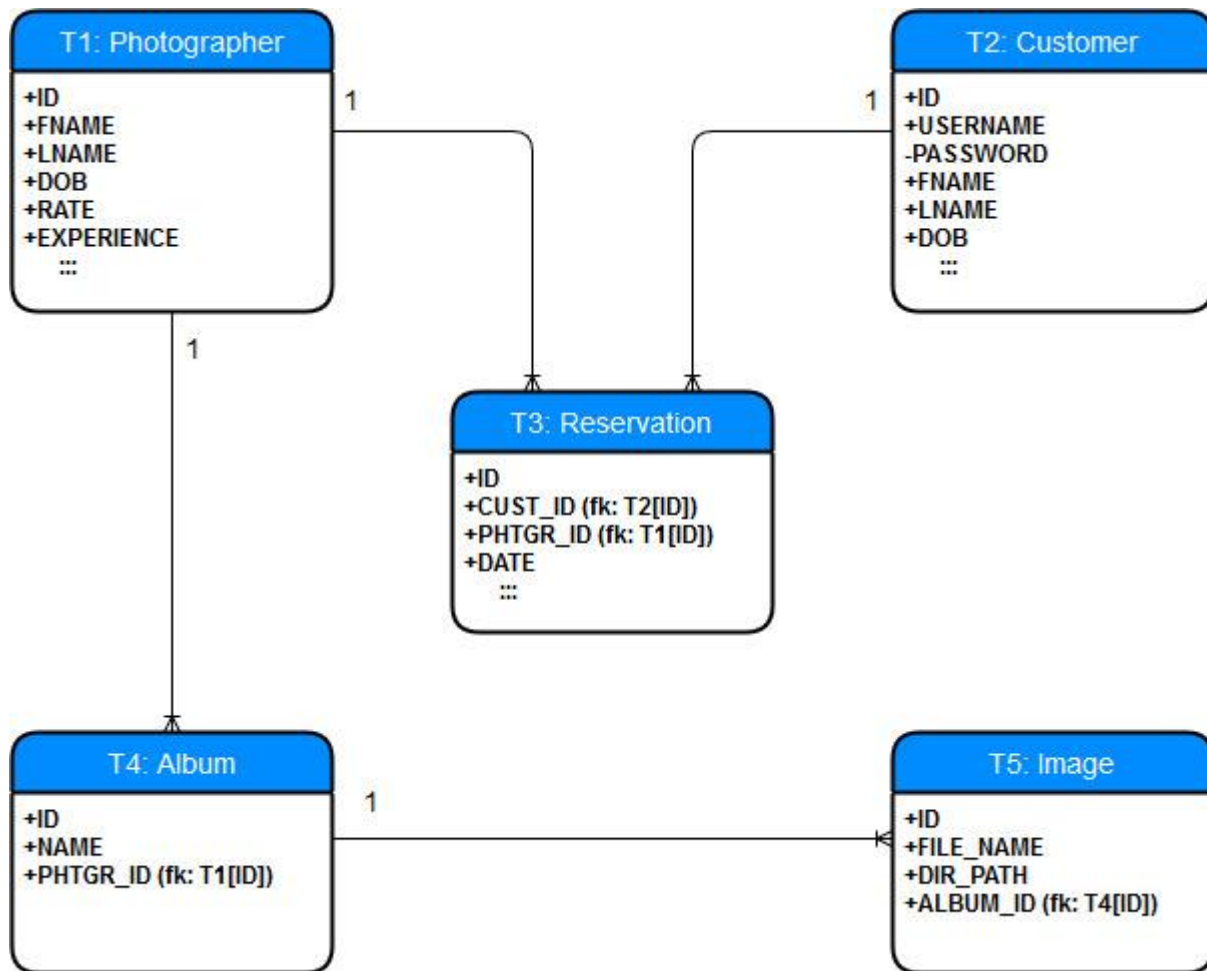
create table customer (
    ID int primary key generated always as identity,
    Username varchar (30) NOT NULL UNIQUE,
    Password varchar (100) NOT NULL,
    Fname varchar (30) NOT NULL,
    Lname varchar (30) NOT NULL,
    DOB date ,
    Nationality varchar (20),
    Address varchar (200))
;

create table album (
    ID int primary key generated always as identity,
    Name varchar (100) NOT NULL,
    Photographer_id int NOT NULL,
    constraint fk_album_phtgr foreign key (Photographer_id)
        references photographer (ID)
)
;

create table image (
    ID int primary key generated always as identity,
    File_name varchar (100) NOT NULL,
    Dir_path varchar (100) NOT NULL,
    Album_id int NOT NULL,
    constraint fk_img_album foreign key (Album_id)
        references album (ID)
)
;
```

```
create table reservation (  
    ID int primary key generated always as identity,  
    Customer_id int NOT NULL,  
    Photographer_id int NOT NULL,  
    Date_reserved date NOT NULL,  
    Time_reserved time NOT NULL,  
    Venue varchar (200) NOT NULL,  
    Proposed_amount double NOT NULL,  
    Status varchar (10) NOT NULL,  
    constraint fk_reserve_customer foreign key (Customer_id)  
        references customer (ID),  
    constraint fk_reserve_phtgr foreign key (Photographer_id)  
        references photographer (ID)  
);
```

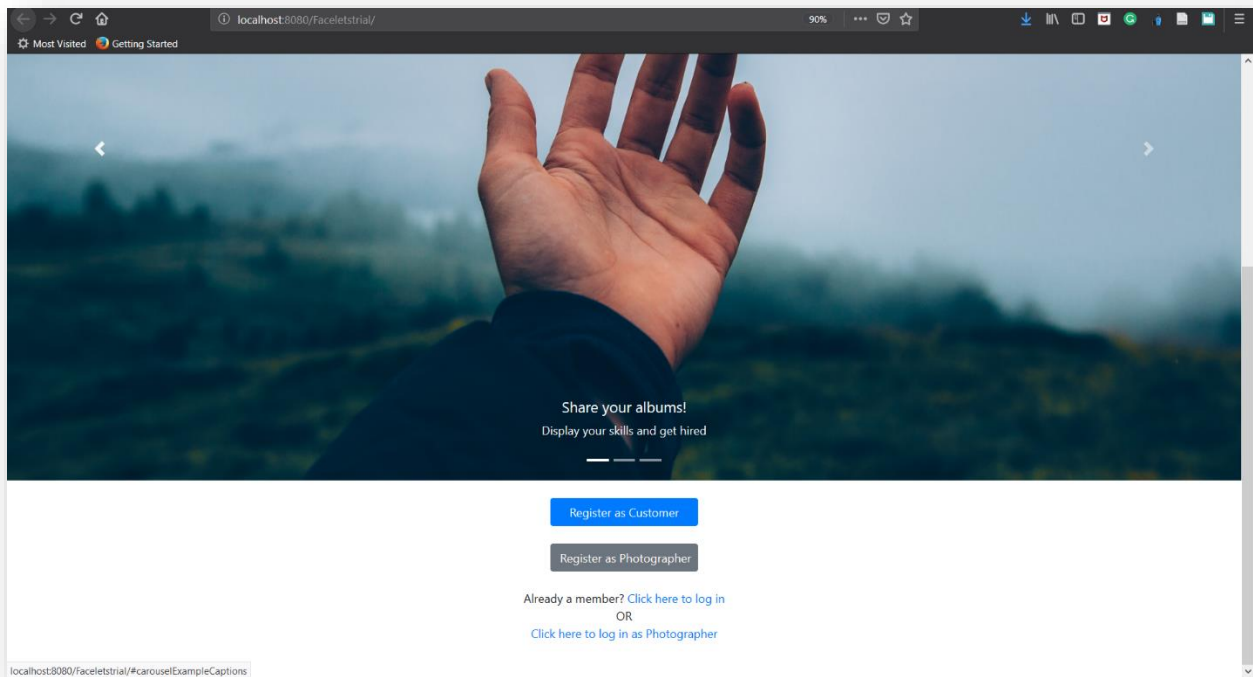
ER Diagram:



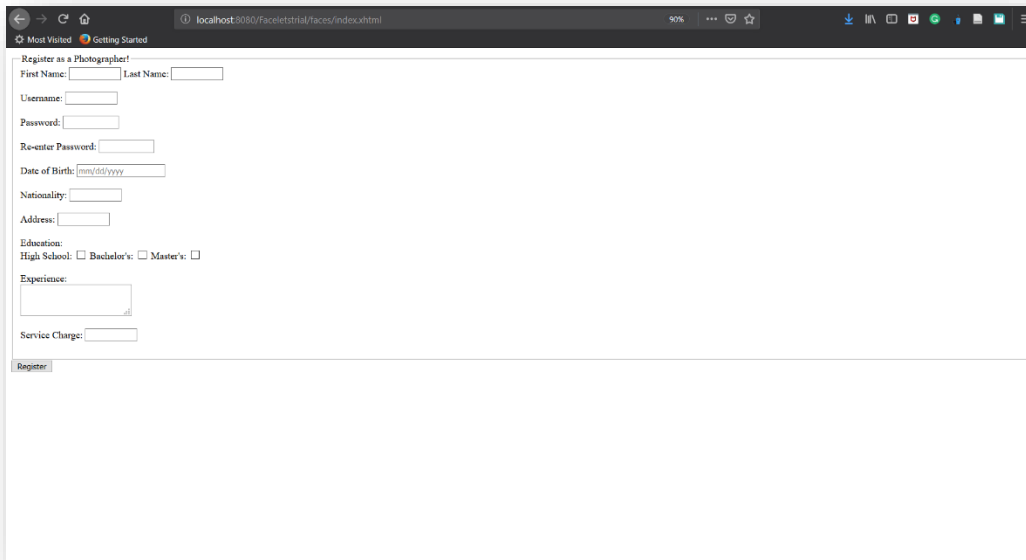
Architecture and Page-flow

The first page is the classic index.html which not only serves the purpose of a warm welcome page but also serves as a switchboard so that the user can select one of 4 options

- Register as a photographer
- Register as a customer
- Log in as a photographer
- Log in as a customer



Upon choosing any of the *Register* options, the website is redirected to their respective forms (photographer / customer). The form consists of input fields with decent security measures, ensuring that the user *has* to input their name, a username and password (*which must match with a re-entry*). Some of the other fields are specific to the type of user and differ.



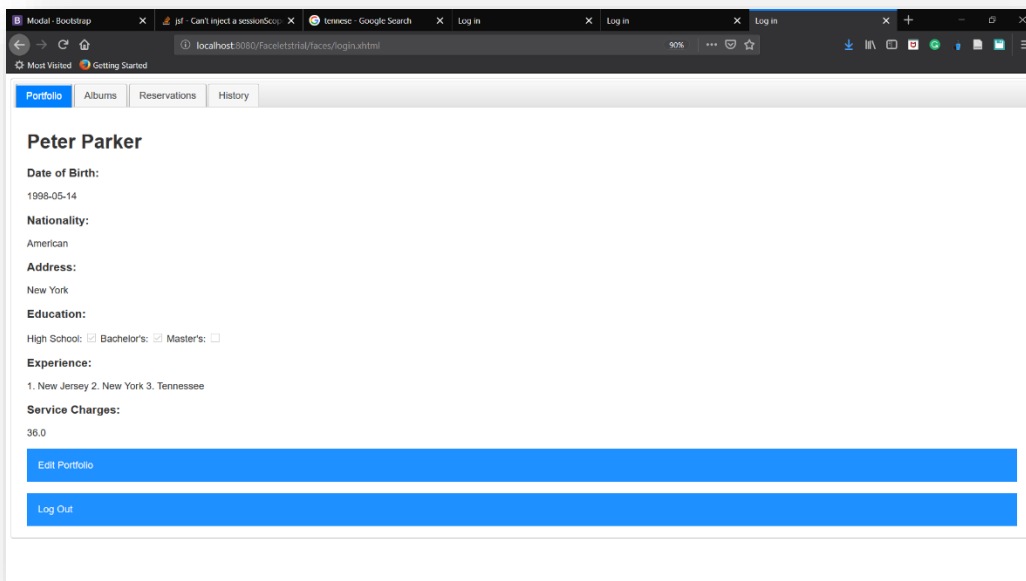
The screenshot shows a web browser window with the address bar displaying 'localhost:3080/Facietrial/faces/index.html'. The page title is 'Register as a Photographer!'. The form contains the following fields and options:

- First Name:
- Last Name:
- Username:
- Password:
- Re-enter Password:
- Date of Birth:
- Nationality:
- Address:
- Education: High School: ☐ Bachelor's: ☐ Master's: ☐
- Experience:
- Service Charge:

A 'Register' button is located at the bottom of the form.

Upon selecting the *Log in* buttons, the user is allowed to access their portfolio (*if a photographer* – portfolio.xhtml) or their profile (*if a customer* – profile.xhtml). The overall working of the portfolio or profile is similar, which provides the UI with 4 collapsible tabs with a set of functionalities.

The portfolio's first tab is called the portfolio itself, which displays the photographer with their details and provides 2 buttons, one for editing their details and one for logging out.

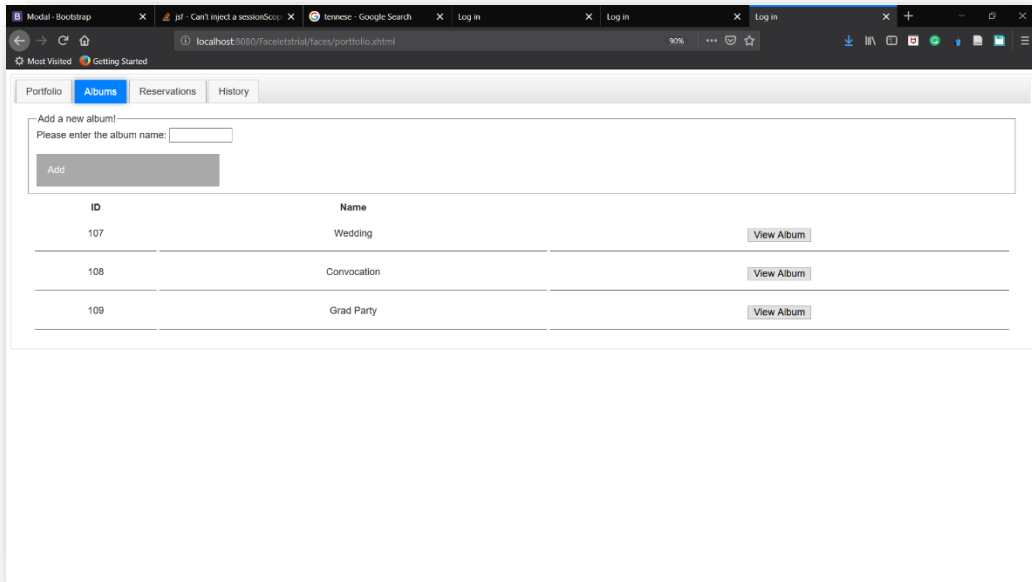


The screenshot shows a web browser window with the address bar displaying 'localhost:3080/Facietrial/faces/login.html'. The page has a navigation bar with tabs: 'Portfolio', 'Albums', 'Reservations', and 'History'. The 'Portfolio' tab is selected, displaying the details for 'Peter Parker':

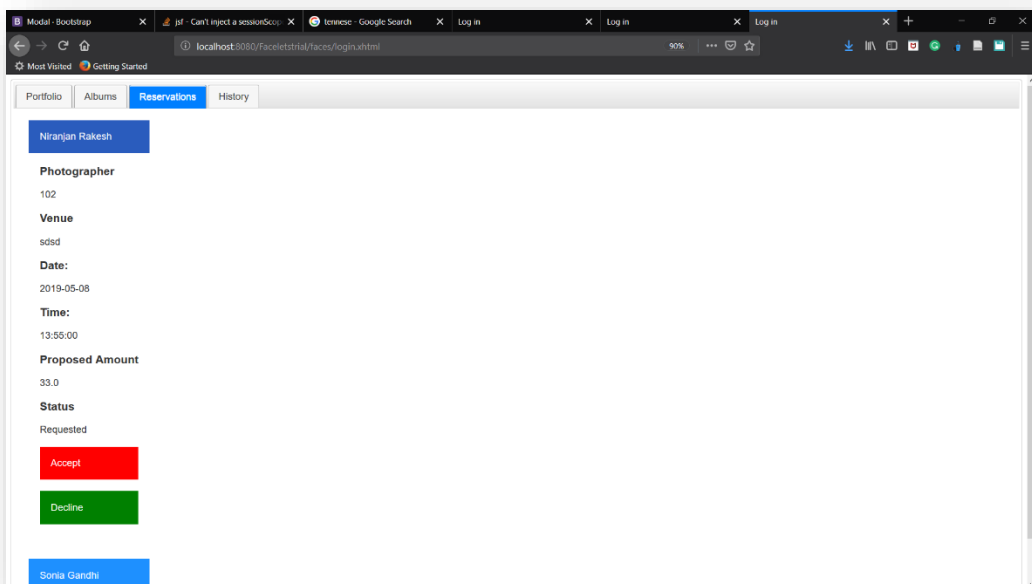
- Date of Birth:** 1998-05-14
- Nationality:** American
- Address:** New York
- Education:** High School: ☐ Bachelor's: ☒ Master's: ☐
- Experience:** 1. New Jersey 2. New York 3. Tennessee
- Service Charges:** 36.0

At the bottom of the page, there are two blue buttons: 'Edit Portfolio' and 'Log Out'.

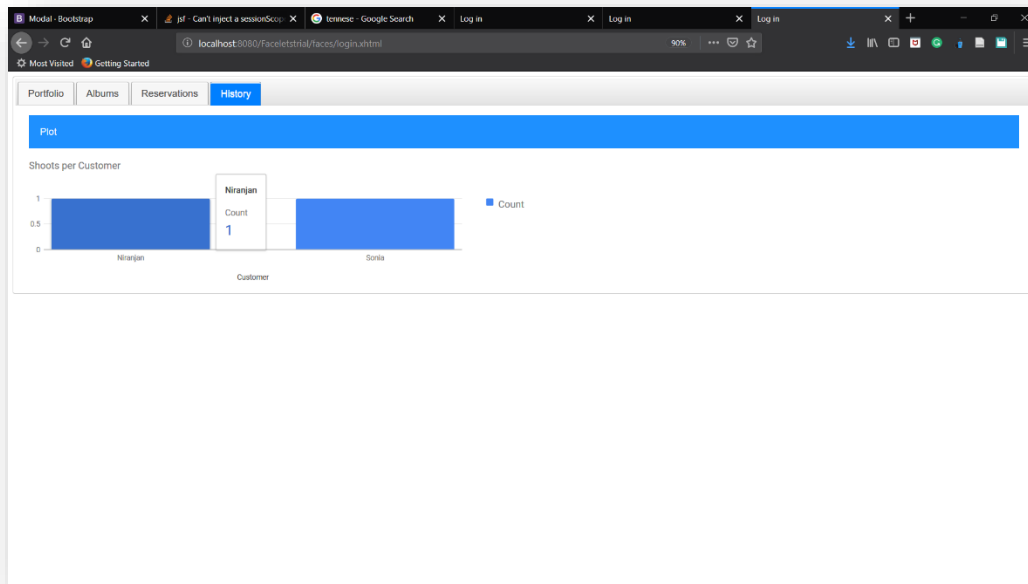
The second tab displays the photographer with a list of their existing albums and allows provides them with an option to add a new album by giving it a name.



The third tab in the portfolio allows the photographer to view all reservations related to him (*whether they are requested, accepted by him already, or declined by him*). By selecting a reservation, the accordion expands and the photographer has the option to accept or decline these reservations.

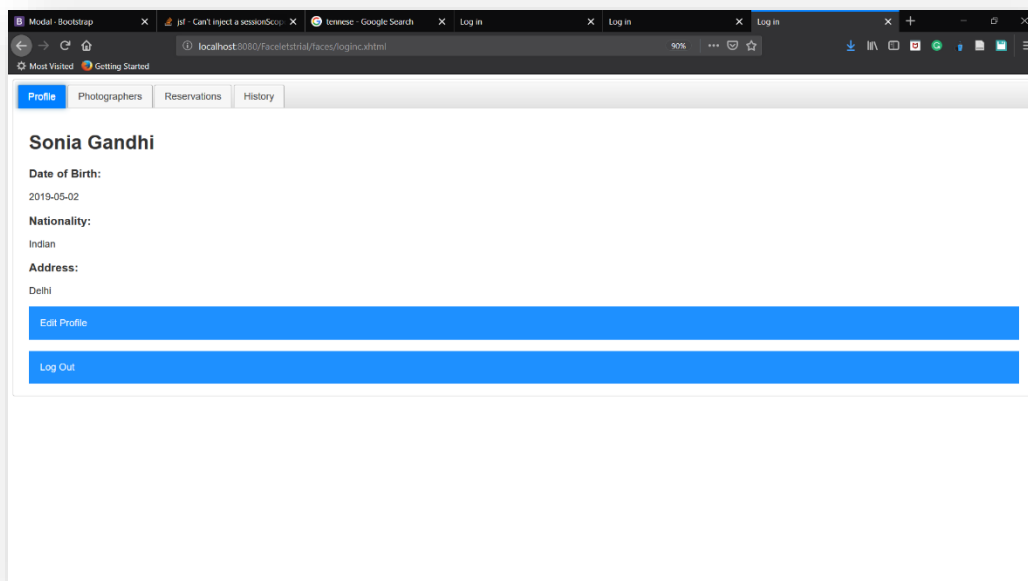


The fourth tab allows the photographer to get some basic insights about his activities, by plotting a bar graph showing the number of interactions (*reservation requests*) he has had with customers.

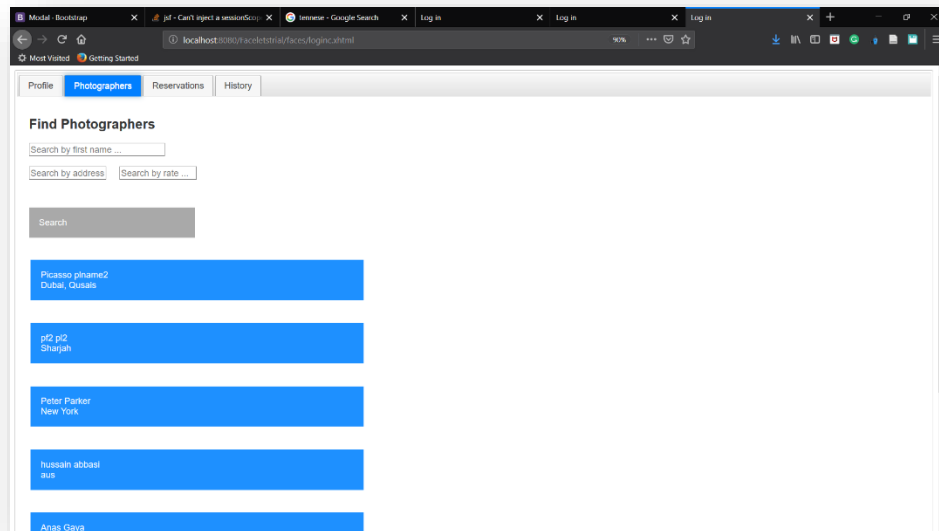


The profile.xhtml is displayed instead of a portfolio, if a customer would have logged in. The profile page, similar to the portfolio consists of 4 collapsible tabs.

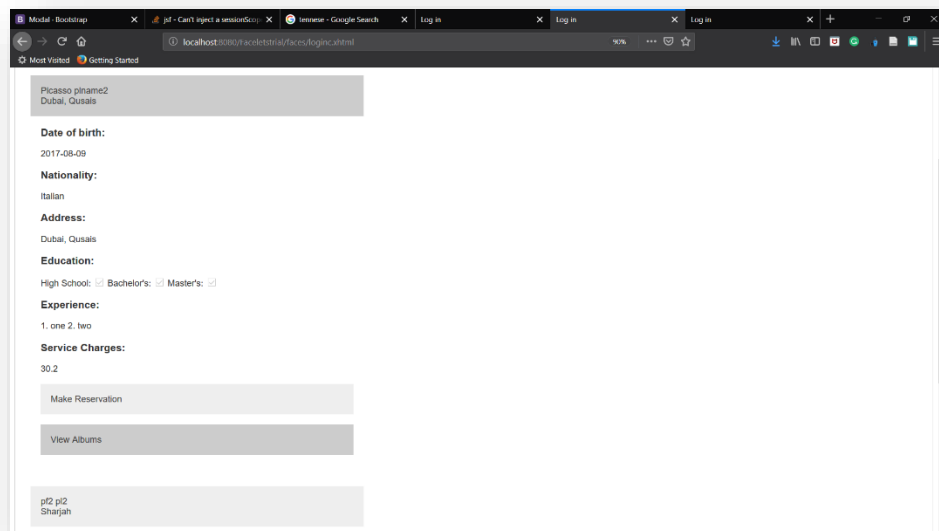
The first tab, similar to the portfolio, is the profile itself, that displays the basic information about the customer and has 2 buttons, one to allow the customer to edit his details and another to allow him to log out.



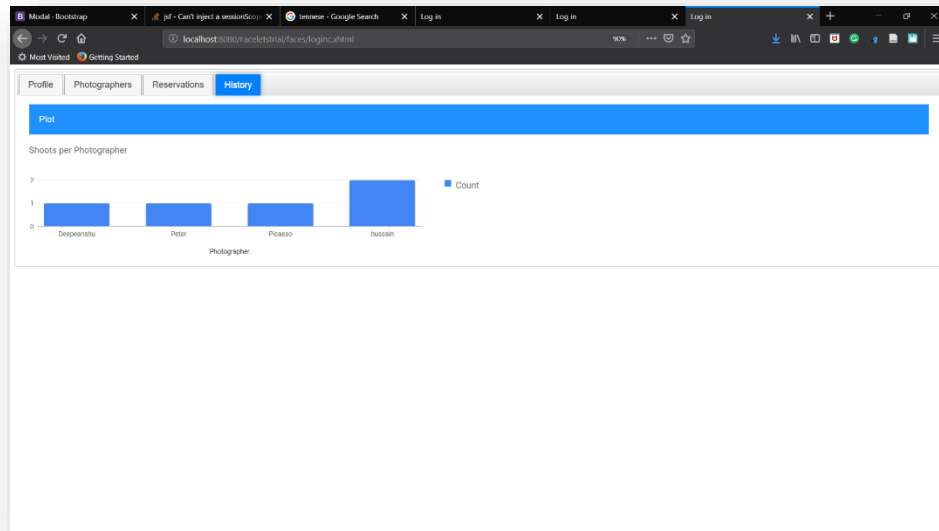
The second tab in the profile.xhtml is the tab that allows the customers to view all photographers or filter their search based on the photographer's first name, address, service charge, or a combination of these attributes.



Upon selecting a photographer, the accordion expands and the customer can see the details of the photographer, view their albums or make a reservation.



The fourth tab in the profile is the same as the portfolio and allows the customer to plot a graph showing the number of interactions he has had with different photographers.



If the customer would have chosen to view the albums for one of the photographers, he is taken to the `viewalbums_for_customer.xhtml` that simply displays a list of albums under the selected photographer. He can select an album to view its images.

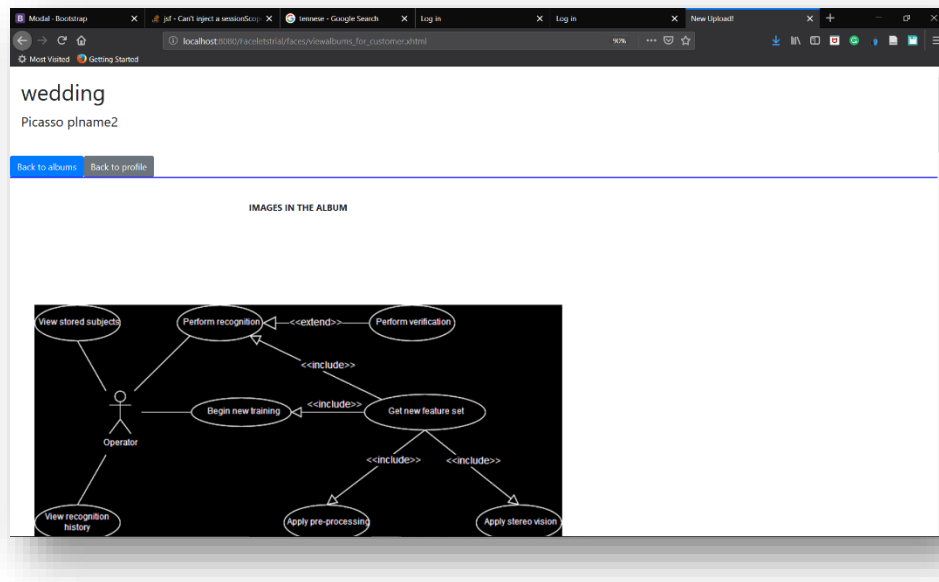
The screenshot shows a web browser window displaying a list of albums. The table has columns for ID, Name, and a 'View Album' button. Below the table is a 'Back to profile' button.

ID	Name	
2	wedding	View Album
3	convocation	View Album
5	testalbum p1	View Album
6	newalb	View Album

[Back to profile](#)

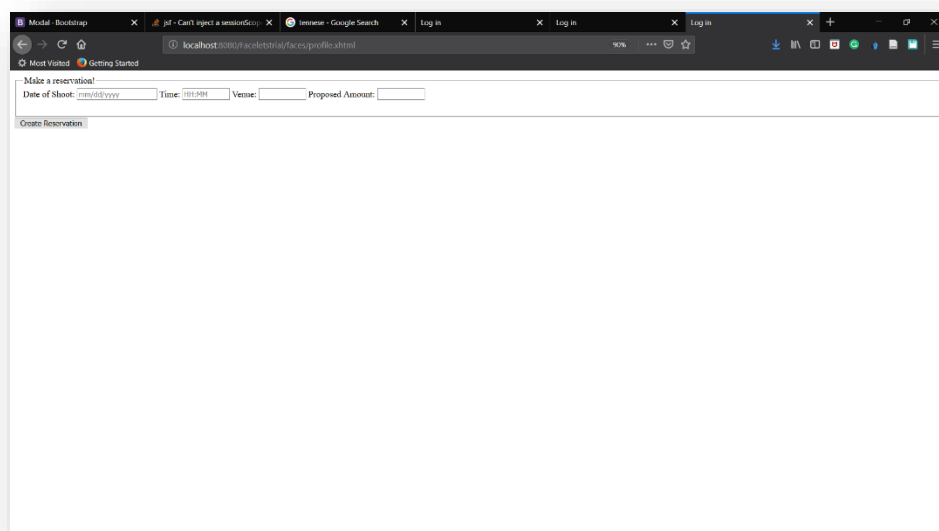
He also has the *Back to profile* button to go back to his profile.

Upon selecting an album, the `viewalbum_for_customer.xhtml` (the previous one was `albums`) is loaded, which allows the user to view all of the images in that album.

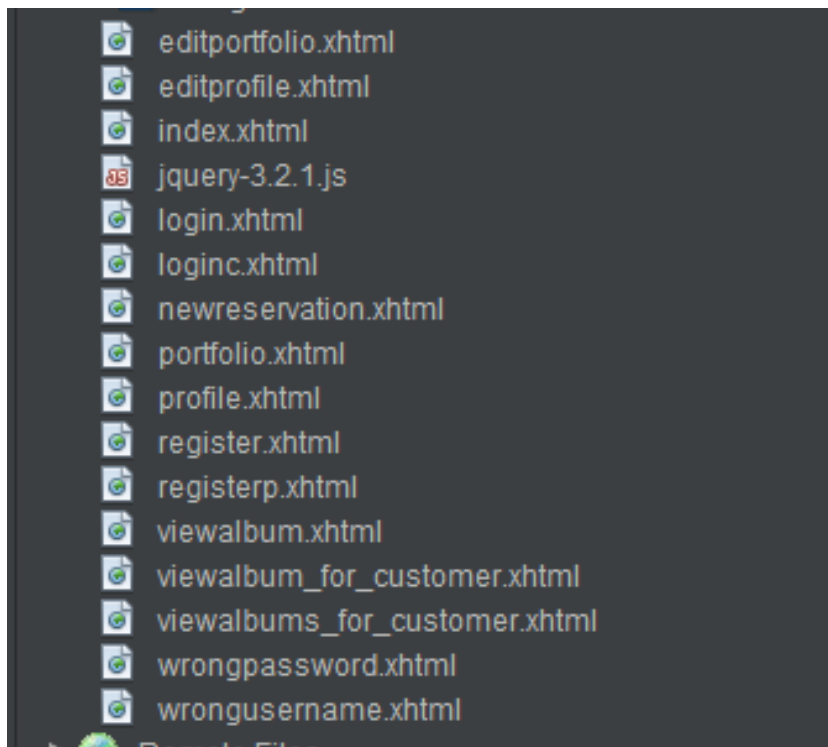


They also have to buttons *Back to albums* and *Back to profile* to redirect them back to those respective pages. The corresponding `viewalbum.xhtml` for a photographer is similar but also has options to add/delete images from the album.

Lastly, if the customer would have selected *Make reservation* for a photographer on his profile, he would have been redirected to the `makereservation.xhtml` that presents him with a basic form to input the details about his reservation.



In general, these are the pages used in the project (*the names are intuitive*):



PS: All the functionalities explained above are fully implemented and tested.