



# **Travel to Japan Website**

**Report Paper | COMP 3670-01 | Professor: Jones Yu | Group: Absolute Zero  
Shamima Rahman, Xiaokai Hu, Jonathan Sudiaman**

## **Problem Description:**

Travelers need a way to access information about their potential destinations, such as transit, hotels, and places to check out.

## **Analysis:**

Most of the Japanese website is from Japanese language. So it is difficult to find the information for general people who does not know the Japanese language. I experienced this problem in my real life situation when I was living in Japan. I realized that the Japanese people does not want to use international language even though they know about it. They probably more focused to put emphasis on their nationality. So it is really a problem for the visitor in Japan to find out tourist places, find out hotel information and travel ticket information. Also some website not giving every information in one application, meaning some website giving tourist information but maybe not hotel information or not the travel ticket and purchase the ticket from the same web application.

## **Aim of of the Project:**

So the aim of our project was to solve all these problems. We wanted to implement the solution of these problems in one web application. That means the traveler won't need to go to other sites to get more information. Everything would be in one site. In one application.

- Firstly, the site will help user to find an attraction by city name.
- Secondly, the site will help users to find an attraction by specific time duration.
- Thirdly, the user can search a hotel first. Then find the attractions near the hotel.
- The application will also allow comments and ratings from people.

## Design:

## Prototype:

Attraction | Hotel | Flight

Enter City or Place

Date: MM/dd/YYYY From: AM/PM To: AM/PM

Search

Osaka Japan 10/29/2016 - 10/31/2016 4:00 to 6:00

Sort By: Most Popular

Please Wait

00000000000000

Photo (1)

Photo (2)

Osaka Japan 10/29/2016 - 10/31/2016 4:00 to 6:00

City/Place Day Date Time

Sort By: Most Popular

Image	Attraction	Rating	Total time
Image	A	★★★★★	Total time
Image	B	★★★★★	Total time
Image	C	★★★★	Total time
Image	D	★★★	Total time

State Rating

Detail Description of 'A'

Image Image Image

Photo (3)

photo (4)

### Prototype of attraction scenario

Our initial prototype was about the three scenarios. Which is (1) Attraction (2) Hotel and (3) Flight. But because of time constraint we were able to finish two scenarios which is Attractions and Hotels. For Attraction and Hotel we pretty much accomplished everything what we wanted to accomplish. But only the loading page we were not able to accomplish for the time constraint. But we searched for it and got the idea how to do it. Finally the recent prototype we accomplished for attraction is given below:

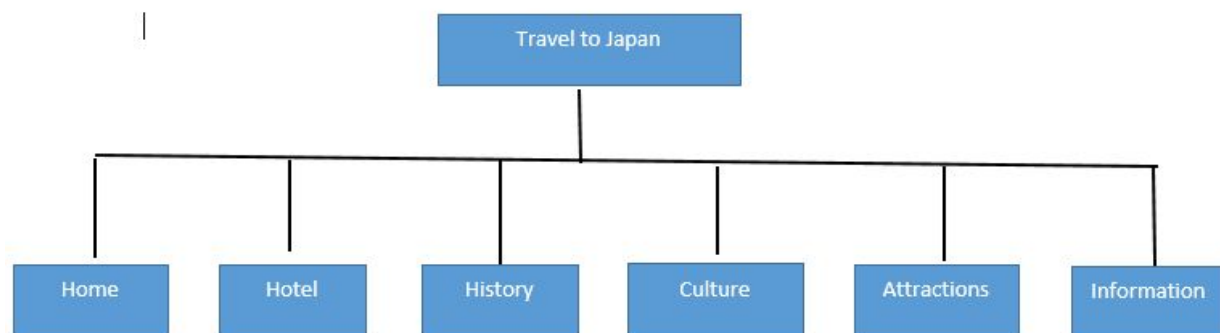
Click on Search will take to the search result page

Click on Detail will take to the detail information page of a specific tourist place



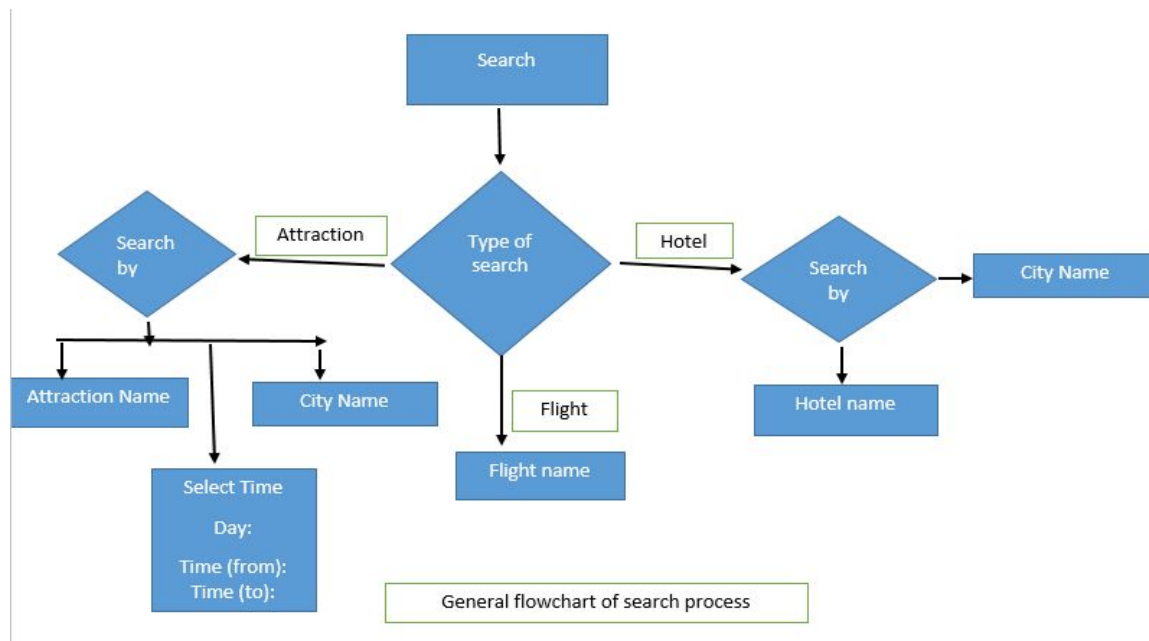
## Recent Prototype

### Flow Chart:



Flow Chart of Site Functionalities

So far in our web application we have 6 pages. 1) Home 2) Hotel 3) History 4) Culture 5) Attractions and 6) Information. They are interconnected with each other.



In the search process scenario, three types of search option. 1) Search type Attractions 2) Search type Hotel and 3) Search type flight. If the user choose Attraction search type then they can search by three ways. Which search by a) attraction name b) city name and by c) by time. If the user choose search type hotel then they can search by either Hotel name or by City name. For now the flight search type is not implemented.

### Attraction Searching:

One of the features of our service is the ability to search attractions within a certain time frame (e.g. Wednesday from 9am-5pm). To do this, we search for attractions in which the user's specified time frame overlaps with the business hours of the attraction. The following is pseudocode of our search algorithm.

```

foreach (attraction in attractions) {
    // These times are in HHMM format, e.g. 9am is '0900'.
    uFrom = (int) userFromTime;
    uTo = (int) userToTime;
    aOpen = (int) attraction.fromTime;
    aClose = (int) attraction.toTime;

    if (!(uFrom >= aClose || uTo <= aOpen)) {
        display(attraction);
    } else {
        hide(attraction);
    }
}

```

## Hotel Searching:

The other feature of our service is the ability to search hotels with given city name and hotel name. And also give user a quick search for attractions near this hotel.

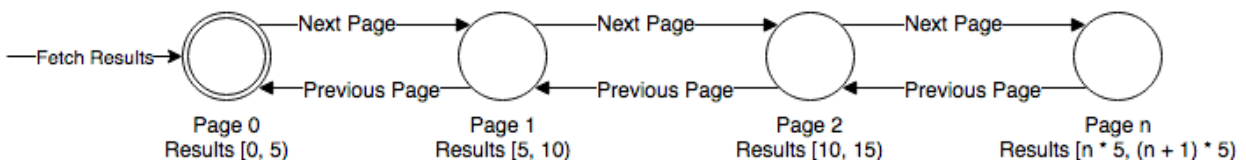
Hotels can be searched with two parameters, city and name of the hotel, name of the hotel can be empty, which will return all hotels in that city. If both are empty it will return all hotels in the database just like the hotel link does in the navigation bar.

## Search Result Pagination:

Now that we've defined two search-based scenarios, we need to figure out the best way to show the search results. The intuitive method would be to eagerly fetch results and list all of them down at once. For our purposes, this solution would probably work well enough as we don't store too many attractions or hotels in our database. However, it will not scale well if we decide to add, say 200 more attractions and hotels. This method will incur long load times and high server stress.

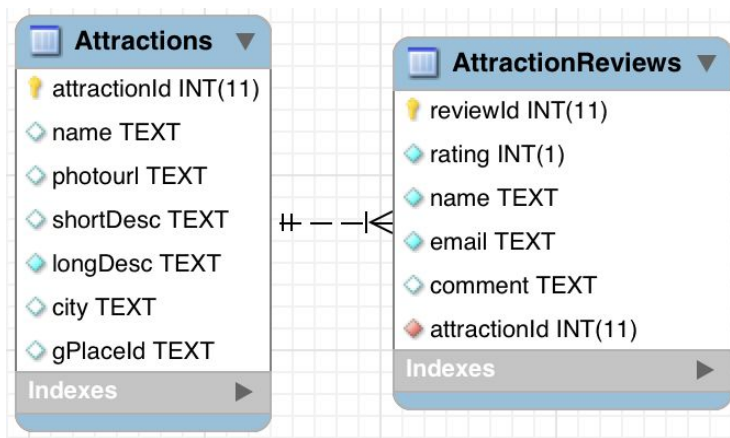
To solve this, we will split our results into pages such that each page contains 5 results at most. (Here, 5 is an arbitrary number, albeit somewhat low. As our application scales, we would most likely increase this limit.) We also introduce a **state** to our results page - namely the current page number. We use this page number, starting at zero, to control some `OFFSET` value, such that  $OFFSET = page * 5$ . For example, on page two, we would show results from 10 (which is  $2 * 5$ ) to 15 (which is  $(2 + 1) * 5$ ).

State transitions are fairly straightforward, and can be modeled by this diagram:



## Database:

Our system is backed by a relational database to store information about attractions and hotels.



Our **Attractions** table has a one-to-many relationship with the **Attraction Reviews** table. Each attraction has a name, photo URL (to be displayed in search results), short/long descriptions, city name, and place ID (for use with Google APIs). Each attraction review has a rating (from 1 to 5), name/email (of reviewer), optional comment, and a foreign key to refer back to the reviewed attraction.

To compute the overall rating of a particular attraction, we take the average rating of all its associated reviews, rounding to the nearest whole number (from 1 to 5).



In our **hotel** table, each hotel has a name, address, city name, phone number, photo URL, price, rating (1 to 5), and latitude/longitude coordinates (used to calculate distance from attractions).

# Implementation:

## Technology Stack:

Frontend uses HTML5, CSS, JavaScript, and jQuery. Backend uses PHP and MySQL. GitHub was used for collaboration and version control.

## Google Maps / Places APIs:

As previously mentioned, each attraction has a “place ID” used to query the Google APIs. We use Google APIs to perform the following:

- Obtain location data and show it on a map
- Find business hours
- Get up to nine additional photos of the attraction

### Example usage - Showing location on map

Attraction details page (JS):

```
var placeId = params['gPlaceId']; // From GET parameters.
var map = document.getElementById('map'); // On the details page.
...
// Connect to Places API
var service = new google.maps.places.PlacesService(map);
service.getDetails({ placeId: placeId }, function (place, status) {
    // place : An object given by the places API
    // status : Acts as a response code

    if (status === google.maps.places.PlacesServiceStatus.OK) {
        // Show Google map with exact location of the attraction
        map = new google.maps.Map(document.getElementById('map'), {
            center: {lat: place.geometry.location.lat(),
                    lng: place.geometry.location.lng()});
    }
    ...
}
```

## AJAX:

When user submits a review for an attraction, their browser uses AJAX to post it to our database. AJAX is similarly used to fetch existing reviews for attractions. We would have liked to use AJAX for our search functionalities as well, along with some sort of loading indicator. However, by the time we learned it, the deadlines were too close to perform all of the refactoring required.

### Example usage - Writing reviews

Review form (HTML):

```

<form id="writeReviews">
...
    <label>Name*<br><input name="name" type="text" required></label><br><br>
...
    <input type="submit" value="Submit Review">
...
</form>

```

### Review form submit listener (JS):

```

$('#writeReviews').on('submit', function (event) {
    // Merge form data with attractionId (database key)
    var data = $(this).serializeArray();
    data.push({name: "attractionId", value: params['id']});

    // Make AJAX call
    $.ajax({
        url: 'submitreview.php', // Target for request
        type: 'GET', // 'submitreview.php' uses GET parameters
        data: data, // Form data
        success: function () {
            // Clear the review form and thank the user
            $('#writeReviews').html("Thanks for your review!");
        }
    });

    // Prevent any further actions
    event.preventDefault();
    return false;
});

```

### Server-side submission (PHP):

```








// Insert review data (passed in through GET params) into database
$stmt = $pdo->prepare("INSERT INTO AttractionReviews (rating, name, email,
comment, attractionId) VALUES (?, ?, ?, ?, ?)");
$stmt->execute([$GET['rating'], $GET['name'], $GET['email'],
$GET['comment'], $GET['attractionId']]);

```

## Evaluation:

### Homepage:



# Welcome to Japan

Home


Hotel

History

Culture

Attraction

Information



In Japanese language the country Japan known as "Nihon" or "Nippon" is an island nation in East Asia. Japan is also known as the "Land of the Rising Sun", is a country where the past meets the future. Japanese culture stretches back millennia, yet has also adopted the latest modern fashions and trends. Although Japan adopted in modern fashion, their traditional lifestyle, culture, custom, language are completely different than modern fashion which is very impressive and enjoyable.

Attractions

Hotels

Flights

Name

City

Anytime  From  To

Search

Places to visit

Nikko: Tokogu Shrine


Tokyo: Harajuku

Hiroshima: Miyajima Island

Sources

[www.yunessun.com](http://www.yunessun.com)  
[www.jnto.go.jp](http://www.jnto.go.jp)  
[Top ten places](#)

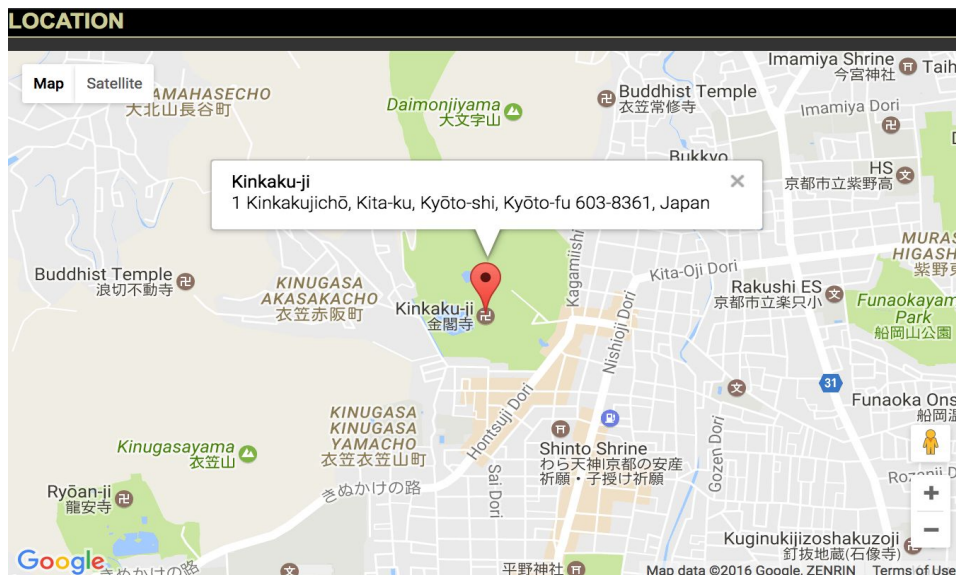
Japanese Language



The Japanese language "nihongo" is a combination of three different scripts: kanji, hiragana, and Katakana.

Japanese Currency

## Locating attraction on a map:



## Fetching images using Google Places API:



## Conclusion:

Working with web application with our group member was really interesting and enjoyable. Everybody was enthusiastic. Time management was very important for us. We also did so much research. For example, we used [travel.com](https://www.travel.com) as a basis for our homepage search box.

## Roles and Responsibilities:

### Shamima Rahman

- Home page: all of the design of home page. Header, Nav bar, table, image thumbnail, Background image, Search box form, link to another page for more information.
- History, Attraction, Culture and Information page
- HTML / CSS / JavaScript
- Mostly front end but also learned and participated on back end with group members and learned how to utilize php and database.

### Xiaokai Hu

- Hotel Page
  - Database
  - Search
  - Detail page

### Jonathan Sudiaman

- Home Page
  - Search Functionality
- Attractions Page
  - Database Tables
  - Search Functionality
  - Google Maps / Places API
  - Ratings and Comments