Design Document

Nameless.exe’s

Travel Planner

Purpose and Scope

This Design Document is meant to outline our project's files, dependencies, changes, and the overall design. This Design Document should be an updated resource for anyone to read and understand how this project is working at all times.

Functional Description

- A page that allows a user to Sign Up/Log In or Log out, so they have trips saved to their account
- A Homepage that shows users the available locations they can plan a trip to
- A page that allows the user to create and plan their trip, lets the user select where they want to go to, when they are going, where they are staying, what flight they are using, and how they plan to travel at their destination.
- A page that allows the user to view all the trip information on the trips they have taken in the past. The page also recommends them another trip.
- A tips and advice section that gives the user insight on what to bring on the trip
- A Expenses page that gives the user the costs of the trip based on the trip they have planned.

Files and Code

HTML

All html files will use Global.css to format their displays.
All html files also have a header UI bar that holds links to Home.html, Sample_Trip.html, Add_Trip.html, My_Trip.html, Sign_in.html as a baseline.

Home.html

Description: First page of the website, Will have the locations that are plannable by trips displayed as clickable links that send the user to a location's page

Dependencies: No Dependencies required.

Links to: lasvegas.html, reno.html, portland.html, phoenix.html, sanfrancisco.html, bellingham.html, seattle.html, losangeles.html

Code Layout: The layout is a table of 2 columns, with 4 rows. Each cell of the table holds a picture with a link to the respective cty page.

(Location).html

Files: lasvegas.html, reno.html, portland.html, phoenix.html, sanfrancisco.html, bellingham.html, seattle.html, losangeles.html
Description: The location pages of the website, Homepage will link to these pages. The page will display prominent landmarks of the location and possible hotels one can stay at. There are Functional web links can be clicked on for each location and hotel.

Dependencies: No Dependencies required.

Links to: No outgoing Links.

Code Layout: A column on the left will hold Hotel links and a column on the right will attractions specific to the city

Sample_Trip.html

Description: This page has been created for mainly a potential user who is not part of the database. This page allows a non-user (or user) to 'get a feel' for how to add a trip. This can also be a huge aid for our older users who are not as 'tech-savvy'. If someone has a question regarding how to add a potential trip, they can visit our Sample Trip and generate a random trip. Upon generating a random trip, a valid data value is displayed for a user to physically see and potentially use for their future trips.

Dependencies: Sample_Trip.js holds functions the page uses.

Links to: No outgoing Links.

Code Layout: The layout of the page holds the description of what you need to enter for adding a trip on the left hand side (in table format). On the right hand side, there are set text describing what needs to be updated/generated. Upon selecting 'Randomize', the right hand side is updated/generated with randomized, valid values that can be used for the actual creation of a trip on the Add Trip page (users only).

Add_Trip.html

Description: This page is for users to add trip(s) to our database (planning for themselves). This is the core of our web app. Upon planning a trip, they now have access to a multitude of data regarding their trip(s). At the same time of them submitting data, we can now populate statistics for future users (sometimes current users in the future as well) to view, in regards to their trip planning.

Dependencies: Must be a current user of our database. Add_Trip.js holds functions the page uses. Global_JS.js holds functions the page uses. firebase.js holds functions the page uses.

Links to: tipsform.html

Code Layout: There are entry fields with descriptions on the left hand side of each them, respectively. A user must input valid data in all fields in order for a users trip to be added. Any incorrect/invalid data causes the page to give an error message and reset all boxes to empty.

My_Trip.html

Description: This page is for users to gauge what their statistics are that we have for them in our database. The user has the option to see their past trips (all trips in the database), our recommendation for them based on their past trip data, or both. The can select what they want to see via a dropdown menu.

Dependencies: The current user must be a user in our database and currently logged in. In addition, they must also have past trip data. My_Trip.js holds functions the page uses.

Links to: This page does not link to other pages.

Code Layout: Dropdown menu at the top with 3 options to select past trips, recommended trips, or both. Past trips will display a table of the past trips in each row, with information displayed being Trip number, name, location, departure time, return time, hotel, airline, transportation, and the total cost. Recommendation will display 2 tables. One for a trip giving the location, airline, hotel, and transportation. The other table displaying tips and advice that is based on the global counts of people who have gone to the same location.

Sign_In.html

Description: This page is to allow a user to log in/log out/create an account in our database. This is also how we verify users and allow for entries and viewing of data on other pages. This is the central hub for interacting with our database for users, without having to know anything about the database.

Dependencies: A user must have an account to sign in/sign out and a user cannot use a previously used email (still in database) to create a new account. firebase.js holds functions the page uses.

Links to: This page links to no other pages.

Code Layout: One main section to hold the sign in option. Sign in will hold one field for email and a field right below it for password. A second section to hold the sign up option. Sign in will hold one field for email and a 2 fields right below it for password and retyping their password.

Tips.html

Description: This page takes the data location from the Add Trip page that a current user just filled out and uses that location to provide suggestions to the user based on the trip they are going to take. This page also allows for a user to enter information regarding what they want to do/take when they go on their trip. This data is then appending to our global counts to be used for others users in the future.

Dependencies: Must have filled out the Add Trip form successfully and been redirected to this page. This is because the location from that page is appended to the URL and used to pull data from the database, as well as, push data to the database upon submission of the form. tipsform.js holds functions the page uses.

Links to: Expenses.html

Code Layout: An initial display at the top telling the user what location they are going to. Below that is a box that shows what other users recommend bringing or doing in preparation for the trip complete with number of votes for every tip. Last is a table which is composed of all the possible tips on the first column, Yes radio button in the 2nd column, and No radio button in the 3rd column.

Expenses.html

Description: This page is used to display all the information to a user on the current trip that they just added with details and prices. This is formatted for viewing pleasure and also has an image of the hotel that they have selected to stay at during their current trip. On this page, the user is given the option of whether or not they want to receive an email with all the details of their trip as a notification/reminder.

Dependencies: user must have been redirected to this page from the tipsadvice.html page with no errors. The current user must be signed in. Expense_data.js holds functions the page uses.

Links to: This page does not link to other pages.

Code Layout: Title noting the page as the expenses page, followed by a warning that the page can't be navigated back to. The next object is a full page detailing all the trip details with all the costs for airline, hotel, transportation, followed by a subtotal, and the actual total. All the trip details organized in a 2 column table, with the details on the left column and costs on the right column. Below the table is finally a dropdown option with text for selecting yes/no if they would like to opt in for an email notification or not.

<u>Javascript</u>

All javascript files leads off with the firebase configuration and initialization.

Expenses_data.js

Description: This file is holding the data for the Expenses page. This file syncs up with the database to verify who the current user is and gets the email. This page also breaks apart that email and pulls data for the specific trip based on what was passed in via the URL. This data allows for pulling of all data for the most recent Add Trip for the user and populates HTML fields with the database data. This file also allows for the updating of the email notification data, as well as, allowing an email to be sent out to a user's currently logged in email address.

Code Layout: Firebase authentication to check for user login
function getInputVal(id) {grabs the html value from the html "id" tag}
function pull_data_Expenses(email_front) {using email username, look up their database entry and pull all relevant trip data then pushes data to disp_Expenses function }
function disp_Expenses(ap, a, d, dt, h, hp, loc, nn, re, re_t, st, tax, tot, tra, trap, em) { takes input data and pushes it to html id's, html's will update and display the data}
function update_email_send() {grabs dropdown value on the html page, and sends email with trip information using function send_email if yes. Pushes trip information to database in both cases using function writeNewPost. }
function writeNewPost(ap, a, d, h, hp, l, nn, r, sub, tax, tot, tr, trap, ue, se) {takes input data and pushes them all to firebase}

function send_email(un, nn, h, hp, a, ap, t, tp, email, image_url, image_name, dep_d, dep_t, ret_d, ret_t, tax, st, tot, loc) {sends email with hardcoded string message, use the input data to fill in the blanks}
function get_loc_data(hotel_choice){use input data to determine what location, filename, and URL link to send to html page for display}

Global_JS.js

Description: This file encompasses a variety of updates to fields via multiple pages. This page has no database connectivity and is used just for updating elements from a collection of locally-stored data.

Code Layout: Initial run that hard codes in all the hotel options for each location. Information is sent to Addtrip.html
function disp_email(){ print to console whatever the value of html id “email” is}
function fn1(){ print to console whatever the value of html id “text1” is}
function email_record(){simple function that prints to console if grabbing a succesful chosen value from html id “e_r” }

My_Trip.js

Description: This file allows for updating of a table on a linked HTML page (My_Trip.html) of data for all past trips that are in our database for the currently signed in user. This file also keeps counts of all past trip data to make a recommendation to the user. Another part of the database returns the highest count for a location’s recommendations. These recommendations are also displayed for the user to see and use if they wish. This file is also is in charge of keeping track of what to display based on the user input on the linked HTML page.

Code Layout: Firebase authentication to check for user login
function get_user_past_data() {uses user’s login to grab ALL of the user’s past trip data, the function then parses through all every trip to and pushes the information into a table on the html side. In addition each trip updates an internal array to count the location, airline, hotel, and transportation preferences. At the very end, once all trips have been pushed to the table, we check every array and take the highest count in order to recommend the trip they like taking the most. The function then ends on reading the location we recommended and grabbing from database the tips and advice for that location and pushing those that are positively recommended to the html table for our recommended tips}
function noDisplay(){hides all html elements on the page from view }

function display(){read the dropdown option and depending on what's selected, display only the related past trips information, display only the related recommended information, or both of the information at the same time with past trips on top then recommendations below}
function gotData(data) {grabs the data object and parses the objects values into separate arrays}
function errData(err) {error function, prints error when called}
function get_updated_loc_for_db(loc) {takes input "data" location and outputs the accurate location string}

Sample_Trip.js

Description: This file holds all the possible entries that a user can enter on the Add_Trip.html page. This data is linked in a way where a random number generator can select 'randomized' data and update different fields on the linked HTML page.

Code Layout: function get_sample_trip_data(){ Uses hard coded strings in arrays which all hold trip information specific to their purpose(location, airline, hotel, departure, return, transportation, trip name)Randomly generate numbers to select indexes from the arrays. The selected strings corresponds to actual trip information, and the trip is then pushed to overwrite a table in html. }
function getRandom_whole(min, max) {randomly selects number between min and max}

firebase.js

Description: This file holds the initial push to the database. This file allows for reading/writing to the account section of the database.

Code Layout: Firebase authentication to check for user login
function logout() {logs user out on firebase, and reset the email and password fields on the html page}
function sign_up() {takes the email, password, and retype password values that the user enters on the html page, verify the passwords match, and creates a new user using the email name section of the email address}
function login() {takes the email and password values that the user enters on the html page, and attempts to sign in or authenticate on firebase with the given username and password, throw an error if it fails }
function getInputVal(id) {return the html value from the html page given the html element's id}
function SendVerification() {checks with firebase that the current user matches that who is logged in on firebase, and sends a verification email to the email address}

function submit_Add_Trip() {grabs all the values on the html page using where the values are stored in the id's, push all values to database}
function forgot_password() {grabs value from html value field and checks database for a password under the said value which should be the user's username. Then send the password by email to the user, return an error if any error occurs }

firebase_push.js

Description: This file saves the Add Trip data to the Realtime Database from the successful 'Add_Trip.html' page.

Code Layout: Firebase authentication to check for user login
function submitForm(e) {grabs the html values that the user inputed for their trip. Check if the user is logged in, and if they are proceed to run function saveMessage with the trip details.}
function getInputVal(id) {grabs the html value from the html "id" tag}
function saveMessage(user_email, name, loc, dep, ret, hot, air, tra) {generate random values based on the departure and return dates; the random values are used for determining the cost. Generate and calculate the tax rate, subtotal, and total costs after as well. Parse the user's email address to grab just the name leaving out the address. Push all trip information with costs, under the user's name to the database.}
function getRandomArbitrary(min, max, days) {get a random number between the min and max, and multiply it by days}
function date_diff_indays(date1, date2) {calculates and returns the exact difference in days between the two dates }

recommend.js

Description: This file sets up the values for the tips/advice form. It sets up the values of each answer choice (or lack thereof) and gets the values ready for pushing to the correct spot in the database.

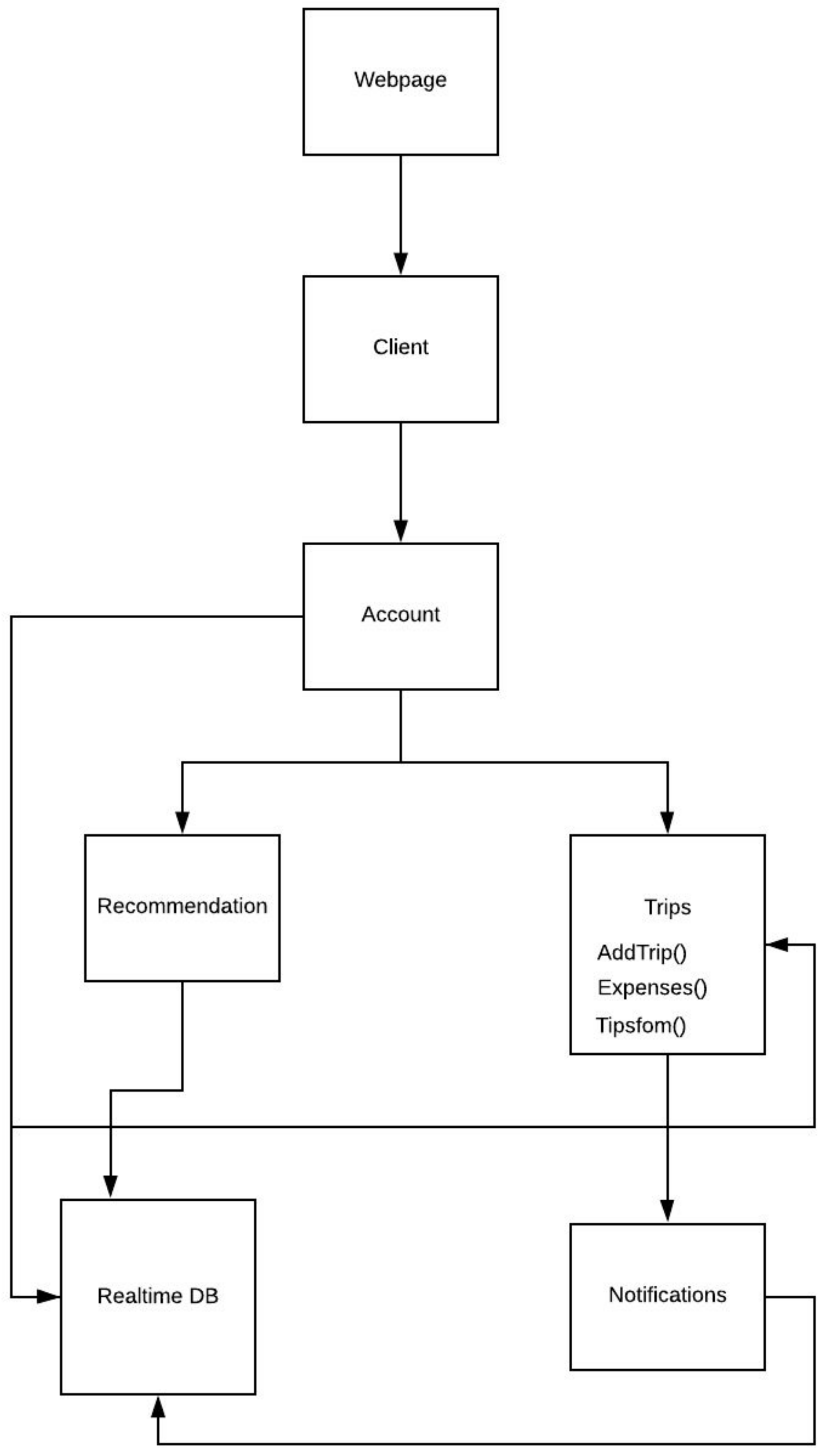
Code Layout: Firebase authentication to check for user login
function Data(data){ loads in location from database based on the user that is currently signed in. pushes the location to an html id named "local". Then picks a random hotel at said location. Function getRandom_whole is used for the randomization. A random airline and transportation are chosen and pushed as well.
function getRandom_whole(min, max) { gets a random number between min and max}

tipsform.js

Description: This form pulls data from the database based on the selected location from the Add_Trip page. It displays this to the user before giving them the option to pick what they would like to do/bring on their trip. This data is then added to a global count and pushed back to the database for later usage in other files/pages.

Code Layout: Firebase authentication to check for user login
function pull_data_Expenses(email_front) {uses the email to check who's logged in and pull from database all the users past trip data. Given the past trip data we know what location to push to tips.html. Using the location we can also pull from database the saved values for each recommendation. Once pulled, we push the positive values to the html page as well}
function dataGather() { takes the values that the user as entered in the radio values in the html page. We have to grab every radio value, so we make sure the radio values that have a value are counted and the ones that aren't don't get counted at all. Save which tips change, and arrange it to an array that matches the database, and push it to update the database values.}
function writeNewPost(ap, a, d, h, hp, l, nn, r, sub, tax, tot, tr, trap, ue, se, rec) {push the updated trip details to database, it's an update because we push back to the same trip using the same trip key}
function writeNewPostRec(loc, arr) {push the updated recommendations to database, using the same recommendation key, so we update the right recommendation }
function get_updated_loc_for_db(loc) {takes input "data" location and outputs the accurate location string}

UML Diagram


Webpage
Client
Account
Recommendation
Trips
AddTrip()
Expenses()
Tipsfom()
Realtime DB
Notifications