Design Specification

Alexander Roan

May 6, 2014

Version 1.2 Draft

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to detail the design of the Online Carpooling Service describing the functional workings of the database and code.

1.2 Objectives

The objectives are to describe the overall system and its workings

2 Database Structure

The decomposition of the database describes each of the tables and their attributes.

Person :

ps_email character varying, not null, Primary Key

ps_first_name character varying, not null

ps_last_name character varying, not null

ps_password character varying, not null

ps_home_1 character varying

ps_home_2 character varying

ps_frequent_destination_1 character varying

ps_frequent_destination_2 character varying

ps_frequent_destination_3 character varying

ps_current_location character varying

ps_register_date date, not null, default = Now()

The 'Person' table holds the personal details of each of the website's users. the 'home_1' through to 'current_location' are not required to be filled in upon registration to the site but are required if the user wishes to receive suggested journeys. The primary key in this table is the user's email address.

Message :

ms_pk big serial, not null, Primary Key

ms_ps_sender character varying, not null, references Person(ps_email)

ms_ps_receiver character varying, not null, references Person(ps_email)

ms_title character varying

ms_body character varying

ms_sent_date date time

ms_read_date date time

The 'Message' table hold all message information between the users on the site. The primary key is generated as the next available primary key in the table.

Journey :

jr_pk big serial, not null, primary key

jr_ps_email character varying, not null, references Person(ps_email)

jr_origin character varying, not null

jr_destination character varying, not null

jr_etd date time, not null

jr_eta date time, not null

jr_spaces_available integer, not null

jr_total_spaces integer, not null

jr_description character varying

jr_register_date date time, default = Now()

jr_extra_distance real, default = 30.0

jr_origin_latitude real, not null

jr_origin_longitude real, not null

jr_destination_latitude real, not null

jr_destination_longitude real, not null

jr_total_distance real, not null

jr_is_cancelled small integer, not null, default = 0

The 'Journey' table is responsible for storing the majority of information relating to a journey posted by a user. The primary key is generated as the next available integer in the table. 'jr_ps_email' refers to the entry in 'Person' table which posted the journey i.e. The driver of the journey.

Journey_Step :

js_pk big serial, not null, primary key

js_jr integer, not null, references Journey(jr_pk)

js_step_order integer, not null

js_latitude real, not null

js_longitude real, not null

The 'Journey_Step' table stores data regarding each step of journey. the primary key is generated as the next available integer primary key. The 'js_jr' refers to the journey that the step belongs to by referencing the 'Journey' table's primary key. The 'js_step_order refers to the order of the step in the journey.

Journey_Step_Temp :

st_pk integer, not null, primary key

st_jr integer, not null, references Journey(jr_pk)

st_step_order integer, not null

st_latitude real, not null

st_longitude real, not null

The 'Journey_Step_Temp' table is identical to the 'Journey_Step' table except for the field names and the fact that the primary key is not generated, but rather inserted.

Hitch_Request :

hr_pk big serial, not null, primary key

hr_jr integer, not null, references Journey(jr_pk)

hr_ps_email character varying, not null, references Person(ps_email)

hr_request_date date time, not null, default = Now()

hr_response_date date time

hr_response small integer

hr_waypoint_1 character varying

hr_waypoint_2 character varying

The 'Hitch_Request' table stores data relating to a hitch request that a user makes to a journey. The primary key is generated as the next available integer in the field. 'hr_jr' references the journey that is being requested to hitch. 'hr_ps_email' refers to the user that made the request by referencing the 'ps_email' field in the 'Person' table.

3 PHP

3.1 Overview

The client side application is split into two structures of PHP files. One group, called the 'Database Controller' will act as the gateway of interaction with the database. This structure is an object oriented representation of the database itself, with controller classes which perform the complex tasks that the client side website will utilize.

The other group will be the website files. These files will be a collection of procedural PHP files which dynamically generate HTML data depending on the data received from the database controller classes.

The design is almost a twisted version of Model-View-Controller, where the controllers ad the models are the database controller, and the view is the website files.

3.2 Database Controller

Each of the tables in the database has a corresponding PHP class, a 'model' class, in the database controller. Each of these classes representing a table contain attributes of the same names as its table counterpart, and overwrite four functions from the 'Table' class they extend. These functions are named 'Create()', 'Load()', 'Update()', 'Delete()'.

Create() inserts a new record into the corresponding table using an SQL procedure and the values that the class' attributes are set to.

Load() takes a parameter as a string, and uses it as the primary key to the corresponding table to retrieve an entry from the table. If an entry is found, it will set the class' attributes equal to that of the corresponding values retrieved from the database.

Update() updates the table using the attributes in the class, of which will include the primary key to the table being updated. This method

assumes that changes have been made to the attributes in the class object before hand.

Delete() attempts to delete a record from the table where the primary key matched the same attribute stored in the class object. It then resets the values of all of the attributes in the object in case the object is used again.

As well as each of the model classes, controller classes instantiate and control them. The model classes are structurally very similar with the only difference being which tables they represent. Controller classes are structured depending on how they make use of the model classes. The 'Journey_Controller' class needs to perform many actions using the 'Journey' model class and interact with the 'Journey_Step_Controller', so there are far more methods in these classes that in the 'Message_Controller' for example.

3.2.1 API.php

All classes in the database controller classes extend 'API.php'

__construct() constructor

API() constructor

ConnectToDatabase()

CloseConnection()

Get()

Set()

GetAll() Returns All attributes from an object

3.2.2 Journey_Controller.php

extends 'API.php'

__construct() constructor

Journey_Controller() constructor

GetMyJourneys()

Private ReturnJourneys() Function used by other functions in the class to format the return structure as an array

FillSpace() Fills a space in a journey, decrements the amount of spaces available for a particular journey

LoadJourney() Loads a journey object to the controller

GetJourneyDataAll()

CancelJourney()

SearchJourney()

CreateJourney()

CreateJourneySteps()

ModifyJourneySteps()

Private MoveOldJourneySteps()

Private DeleteOldJourneySteps()

SetWaypointArray()

BuildDirectionsUrl()

Geolocate()

Private SetGeolocation()

Private BuildSearchString()

3.2.3 Journey_Step_Controller.php

extends 'API.php'

__construct() constructor

Journey_Step_Controller() constructor

MoveTempJourneySteps()

DeleteTempJourneySteps()

CreataeJourneySteps()

3.2.4 Message_Controller.php

extends 'API.php'

__construct() constructor

Message_Controller() constructor

SendMessage()

LoadMessage()

LoadMyMessages()

GetMessages()

GetNumberOfUnreadMessages()

3.2.5 Hitch_Request_Controller.php

extends 'API.php'

__construct() constructor

Hitch_Request_Controller() constructor

GetHitchRequestsForJourney()

GetNewRequests()

GetMyHitchRequests()

Private ReturnHitchRequests()

CreateHitchRequest()

LoadHitchRequest()

AcceptHitchRequest()

DeclineHitchRequest()

3.3 Website

The website files will be the files that are accessed by the user when interacting with the website. They comprise of 7 main files which display the main features of the site. These files are as follows:

index.php The main page that the user comes to when accessing the site. Has a register form and a log in form on it. If a session for the site already exists then the browser is redirected to 'home.php'

home.php The home page when logged on to the site. If there is no session already then the browser is redirected to 'index.php'. This page shows suggested journeys for the user.

activity.php Displays the user's journeys and hitch requests and prompts any changes that have occurred for them to respond to.

messages.php Shows the user's inbox, sent messages and enables them to send a new message.

profile.php Shows the user's profile details.

search_journey.php A form with search parameters for the user to fill in when searching for journeys.

post_journey.php A form page with fields which the user is required to fill in before sharing a journey on the site.

Each of these pages require other files to function properly. For example, the 'profile.php' page has a button on the bottom of the page labelled 'Edit Profile'. This button takes the user to 'edit_profile.php' where the static fields from 'profile.php' are editable, and can be submitted to the database using the 'Submit' button which activates database controller classes.