Table of Contents

[1 Introduction 1](#_Toc434415165)

[2 Programming Languages and SoftwareUsed and Their Underlying Concepts 2](#_Toc434415166)

[2.1 Ruby 2](#_Toc434415167)

[2.2 Ruby on Rails 3](#_Toc434415168)

[2.3 Git and GitHub 3](#_Toc434415169)

[2.3.1 Software Framework 3](#_Toc434415170)

[2.3.2 Model-View-Controller (MVC) 4](#_Toc434415171)

[3 Data Sources 4](#_Toc434415172)

[3.1 National Oceanic and Atmospheric Administration (NOAA) 4](#_Toc434415173)

[3.2 California Department of Transportation (Caltrans) 4](#_Toc434415174)

[3.3 Google Maps 6](#_Toc434415175)

[4 Creating the Ruby on Rails Framework and its key elements 7](#_Toc434415176)

[4.1 Creating the web framework 7](#_Toc434415177)

[5 Conclusion 10](#_Toc434415178)

# Introduction

There is no doubt that the United States is a car culture. The automobile is daily relied on for transportation to and from work, soccer practice, the doctor, etc. There is one element of this culture that is special and unique; that most all of us go through and usually at a younger age. This is the road trip. You pile into a car with one or more people, one of which is a friend, and head out onto the highway to go camping, see another friend, go to some ones birthday party, or some other event requiring a long drive typically four or more hours. During this journey unexpected events occur friction in the car, a flat tire, closed freeways. Use your imagination or personal experience lots of things can happen on a road trip.

In the days before the internet and massive amounts of real time data you had an address a paper map, and maybe some directions from the place you are trying to find. The person in the passenger seat was your defacto navigator; hopefully they could read the map. In inevitably you got lost and had to stop for directions, ended up in a bad part of town, or missed your destination an hour ago. The culture in the car during a road trip is usually quite distracting with playing music, sharing stories, and other things.

Today the road trip is a different thing. With very high quality mapping software like Google Maps, In Dash Navigation Systems, Streets & Trips by Microsoft, Street Atlas, and other software the road trip has become less of a navigational challenge. Smart phones have excellent mapping applications with turn by turn directions and many other features. Some of these applications use crowd sourced data, Maze is one of these applications. Other driving apps like Along The Way point out the places to eat and sights to see for a day trip. Others include Hotel Tonight, Priceline, Songza, Animoto, Glympse, Gas Buddy, SitOrSquat, and many more. However most of these offer services not related to the actual road travel to get from point A to point B. In addition, there are better and well established apps to address the issues of finding food, gas, lodging, or any other need during your trip. Like Yelp, Priceline, GasBuddy, Musement, and others. What is missing is a program that will tap weather conditions along your route, access the current road conditions and the road conditions ahead as well as and provide images of the route ahead.

When travelling long distances weather may significantly change necessitating a layover, delay of your trip, bumping up your departure time, or all out cancelling of the trip. For example a trip from Bakersfield, CA to Incline Village, NV goes over two mountain passes. In the winter and, though less common, even in the summer months blizzards, car damaging hail, and flash floods can occur. The usual route to Incline Village is to take Highway 99 to Sacramento, to Interstate 80 (I-80) to Highway 267 to Incline Village. This trip takes approximately 6 ½ hours. Many different climate zones are traversed necessitating up to date and accurate weather forecasts days in advance as well as on the travel day. There is a joke in the Sierra Mountains that there are two seasons winter and road work season. Knowing of road work, lane closures, and especially night time road closers is vital for many a road trip goes long into the night. It is so often the case that 11:00 PM is when the closure begins. So up to date road conditions are key. It could be raining in Sacramento and dry in the mountains and if you time your drive right you can miss the snow or rain in the mountains.

# Programming Languages and SoftwareUsed and Their Underlying Concepts

## Ruby

Ruby was developed in the mid 1990’s by Yukihiro “Matz” Matsumoto. It is an object oriented language that was designed for productivity and implementing a quality user interface. Ruby is a dynamic language. One of the philosophies of Ruby is to focus on the human coding and not obsess on how fast or efficient the machines run. Mats followed the principle of least astonishment (POLA) intending that the code should behave in a manner that experienced users could understand. Ruby is written in the C language. Background C code can be seen and, though highly not recommended, modified. The following is C code for reversing an array in Ruby and demonstrates how C is the underlying language of Ruby and how Ruby used this code to create simpler methods to call.

rb\_ary\_reverse\_m(VALUE ary){

long len = RARRY\_LEN(ary);

VALUE dup = rb\_ary\_new2(len):

If (len > 0){

Const VALUE \*P1 = RARRY\_CONST\_PTR(dup) + len – 1

VALUE \*p2 = (VALUE \*)RARRY\_CONST\_PTR(dup) + len – 1;

do \*p2-- = \*p1++; while (--len > 0);

}

ARRY\_SET\_LEN(dup, RARRY\_LEN(ary));

Return dup;

}[[1]](#footnote-1)

For this project Ruby version 2.1.5 is used.

Because this is a web based application Embedded Ruby (ERB) needs to be used to distinguish its code from regular HTML. The ERB comes between the characters <%.......%>. ERB for example that prints the first name of someone takes the format <%= user.first\_name %>. This is powerful because it creates a template that can be used for many different pages and in many locations. This process is executed on the server side.

## Ruby on Rails

Ruby on Rails also known as just Rails is an open-source web application framework. It was written by David Heinemeier Hansson. Rails is a model-view-controller (MVC) framework. It is written in Ruby and includes structures for web pages, databases, and web services. Common web formats like JSON, XML, HTML, CSS, and JavaScript are all supported. Rails favors a convention over configuration development philosophy. Rails makes assumptions on what every website needs to aid in its deployment of a framework. Convention over configuration tries to reduce the number of decisions a developer needs to make gaining simplicity but not losing flexibility in the code. Some well-known companies use Rails for example GitHub, Shopify, and initially Twitter. Twitter began on Rails but became too large to continue running on this platform. This is often a source of criticism of Rails but the reality is that Twitters extremely large member base, data repository, and huge user interaction simply outgrew Rails, which was designed for small to medium loads. For this application Ruby on Rails 4.1.8 will be used.

## Git and GitHub

To manage the files in this project the open source programs of Git and GitHub will be used. Git is a distributed version control system used widely for software development. Only the command line interface is supported by Git. Git’s core constructs include data integrity, speed, and the ability to have a distributed non-linear workflow. Repositories are central to Git’s functionality it is a collection of files, directory structure, and metadata that is stored on the disk. Copies of these repositories can be disseminated to others working on the same project. Git is useful in that it tracks changes to the repository. This is accomplished by the following command line input *git commit –m “put a message here”*. This command creates a unique point that is maintained with a hash. The user is able to “revert” to one of these “commit” positions if future versions are lost, corrupted, or any other reason. These actions are stored locally. The user is able to work with Git files offline and sync later. With the “git push” command the documents can be pushed to a cloud storage company called GitHub. Git is criticized for being very complex to work with and difficult to properly manage projects over different computers/users, and a poor merging process. Its competitor also has its weaknesses and it is a bit of personal preference on which one to use, although Git is more powerful.

GitHub hosts Git repositories on the web via a GUI. There are private and public repositories on GitHub. The Git command line also works on GitHub. The public repositories can be accessed by anyone and are frequently used for communal projects. GitHub also offers some social functions like feeds, followers, a social network graph, and other elements. The core elements of GitHub are the fork, pull, and merge commands. This allows individual users to check out the code by creating a branch to modify and merge back into the main branch.

### Software Framework

Frameworks are a beginning skeletal structure of a program that can then be modified for customization. They consist of the code most often needed for its particular niche. The classes and methods most often used in the specific framework area are already in place. In this case a web framework such as Ruby on Rails. Frameworks are beneficial to the coding process because they speed up application development, use reliable code, they are more secure, and they are modular.

### Model-View-Controller (MVC)

The format of MVC is quintessential to Rails. The software is divided into three interconnected elements: the model, the view, and the controller. Initially the MVC methodology was used for GUIs and currently it is a key format for the design of web pages. Models are the stored data like a database, csv file, or other formats of data storage. The view is how the data is presented to the user; frequently in HTML. The controller is the connection between the Model and the View. It says how to get the model or data and turn it into a view the user can read. The controller can also update the model.

# Data Sources

## National Oceanic and Atmospheric Administration (NOAA)

NOAA is a very large organization running under the U.S. Department of Commerce. They manage satellites, buoys in the ocean and weather stations on land, set up tornado warning as well as tsunami warnings. They forecast the weather on land and in the ocean. NOAA has collected a very large quantity of data over a wide range of subjects and time period. NOAA has many APIs and works hard to share their data. This program will connect with the National Oceanic and Atmospheric Administration API for the National Digital Weather Forecast Database. NOAA uses a Simple Object Access Protocol (SOAP) to generate XML files that can be parsed for the desired information.

For this project the data from NOAA is accessed through their SOAP protocol. This protocol has nine formats to choose from point data, data for a list of points, latitude and longitude sub grid (rectangle of data points), line function, latitude and longitude zip code, latitude longitude city name, a time period with latitude and longitude, and other searchable formats. For this project the latitude longitude city name is being used to capture the weather report for the specific city. This format was chosen because cities are easily identified along a driving route and typically have NOAA weather data. Even the smaller cities with no NOAA data will default to the closest location with NOAA data. NOAA converts city names into latitude and longitude coordinates. This takes ambiguity out of the search for there are many cities called Springfield in the United States (US). This is why for this program city name and state are required to determine the desired city. From this data Google Geocoding is used to convert the name of the city (or an address) into latitude and longitude coordinates that NOAA can then calculate the closest point where they have a weather station.[[2]](#footnote-2)

There will be the option for multiple cities weather data. Because of this the data from the initial requests must be saved for later display. A session variable is used to keep track of the requests. Rails stores user information as a cookie in the user’s browser but this is limited to 4k of data. This is too small to contain the weather data pulled for this application. Other options for storage include storing the information in a database or in RAM. The database has a slower retrieval time but is stable if there is a crash. RAM has a faster retrieval but if there is a crash all data will be lost. The storage format for all of these is a hash. The first line below is the original code for saving session data as a cookie and the second line contains the code to change the session storage to the RAM.

# Rails.application.config.session\_store :cookie\_store, key: '\_road\_trip\_website\_session'

Rails.application.config.session\_store :cache\_store, key: '\_road\_trip\_website\_session'

The purpose of saving the sessions is to ultimately save the data so it can be displayed later on a master page with all of the road and cam information. This brings up the question of what is the best way to store the data. This project has implemented the RAM storage methodology favoring speed and the belief that crashes will be rare and the necessary repeat of data entry is small.

## California Department of Transportation (Caltrans)

Caltrans has a variety of tools to help the driver navigate from point A to point B. Some of these include road maps, highway conditions, construction alerts, road closer alerts, chain restrictions, and other useful data relating to roads in California. Currently Caltras has “Caltrans Earth” which uses Google Earth API but it will be retiring on 12.12.2012 because of security flaws. Caltrans does not have an API but does have very stable web pages that contain plain text regarding the road conditions throughout the state. Below is an example of the text reporting style for Caltrans and its importance when travelling. The closure of I-80 the major east west Interstate Highway is very important for the traveler.

“

I 80

[IN THE SAN FRANCISCO BAY AREA - SOLANO CO]

NO TRAFFIC RESTRICTIONS ARE REPORTED FOR THIS AREA.

**[IN THE NORTHERN CALIFORNIA AREA &; SIERRA NEVADA]**

**IS CLOSED TO WESTBOUND TRAFFIC 0.5 MI WEST OF DONNER SUMMIT (NEVADA CO) -**

**DUE TO AN ACCIDENT - MOTORISTS ARE ADVISED TO USE AN ALTERNATE ROUTE**

MOTORISTS ARE SUBJECT TO LANE REDUCTIONS IN BOTH DIRECTIONS FROM

SIERRA COLLEGE BLVD /IN ROCKLIN/ TO NEWCASTLE RD /IN NEWCASTLE/ (PLACER CO)

FROM 2000 HRS EACH NIGHT TO 0900 HRS EACH MORNING SUNDAY THRU FRIDAY THRU

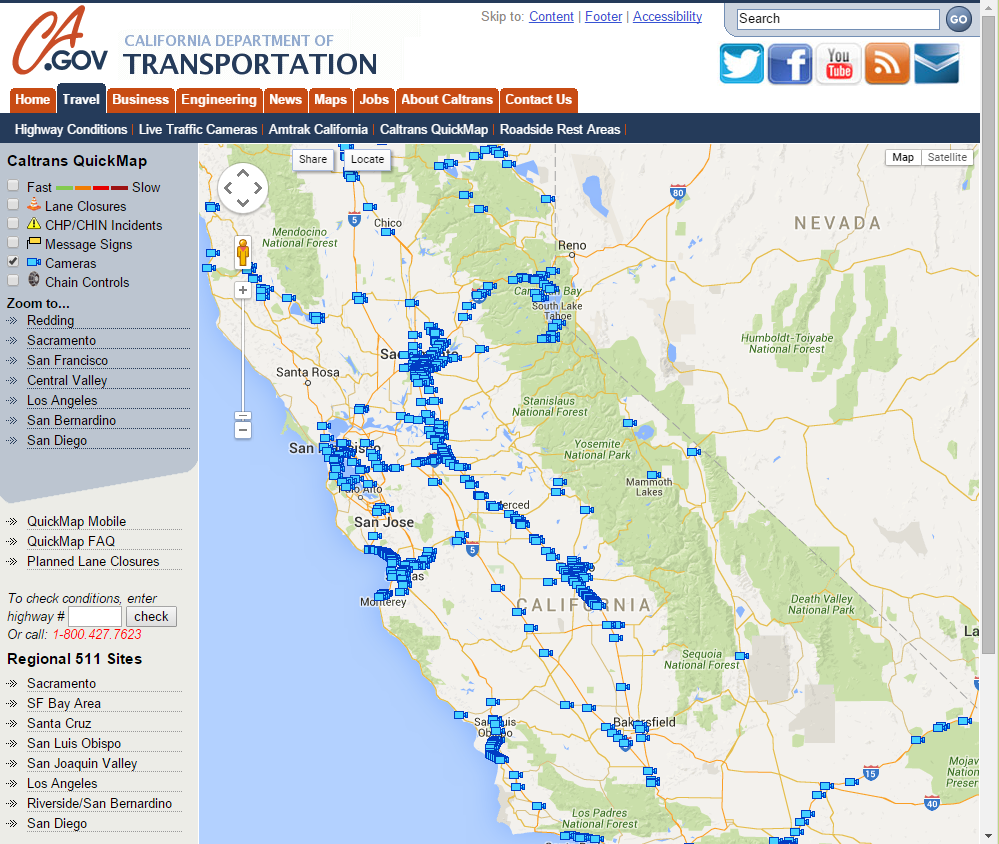
10/16/15 - DUE TO CONSTRUCTION

“

The webpage where the data will be accessed is called “Highway Tables” and can be found at <http://www.dot.ca.gov/hq/roadinfo/hwytables.htm>. These tables list all of the highways in the state and list the road conditions for each of them. It is a great resource for the web page. There are a couple of draw backs though; one is that it relies on webpages for the data and web pages can change easily creating problems. The setup of these links appears to be a permanent location for this data however. The I-5 information page appears as follows <http://www.dot.ca.gov/hq/roadinfo/i5> and for US highway 50 its address is <http://www.dot.ca.gov/hq/roadinfo/us50>. Each highway has its own terminating values that relate to that specific highway so it seems that this is a stable format unlikely to change soon or often. To capture the data from the Caltrans Highway Tables the “Inspect Element” function was used in Chrome and the area containing the Highway Table links was found; in this case it was an entire <div>. The <div> was copied into the “roads.html.erb” file in the project. It can be found at “Trip-Details-Planner” >> “road\_trip\_website” >> “app” >> “views” >> “pages” >> roads.html.erb.

Images from road cams will be a second source of data from Caltrans. There are cams throughout the state highly concentrated in urban areas and sparser on less traveled roads. They offer still images and live video. One challenge with this data is that in rural areas the data is still sent via a dialup connection and is updated only once or twice a day. In other parts of the state especially in urban areas there are live traffic cams. Initially not setup for sharing with the public Caltrans is making changes to their system to allow access for the public. In Figure 1 the location of cams is displayed over a Google Maps map. Each one of the blue cameras has a live feed or still image. The cams are set up so that the image appears in a popup window. This is a challenge to include in the Road Trip Info page, the page that contains all of the information weather, road conditions, and cam images. Also shown in the picture is the Caltrans QuickMap options for additional data layers. The CHP/CHIN Incidents and Chain Controls are very important information sources that currently are beyond the scope of this project but their incorporation in the future will add value to the application. Also a significant source of information but a step too close to having too much data is the QuickMap Fast/Slow and Lane Closures data layers.

Figure 1: Caltrans cams in the central California area and other data layer options



## Google Maps

Google Maps is a very powerful application that is widely known and used. Many applications and webpages use Google Maps a large variety of ways. There are many APIs offered by google for developers. Some of them include Directions API, Distance Matrix API, Elevation API, Geolocation API, Places API, Roads API, Time Zone API, and more. For this webpage the Geolocation API is used. Geolocation is where addresses and other text based locations are converted into latitude and longitude coordinates. This process can also be reversed going from latitude and longitude to addresses or place names. Google requires users of their Geolocation service to register and obtain a session key. The free service has a limit of 2,500 requests per day and no more than 10 requests per second. For more traffic a fee is required. The data is accessed via a HTTP request. See below for the code that accesses the Geolocation data from Google.

geocoding\_api\_key = ENV["GEOCODING\_API\_KEY"]

google\_url = "https://maps.googleapis.com/maps/api/geocode/json?"

google\_params = "address=" + params["city1"].gsub(" ", "%20") + "&" +

"key=" + geocoding\_api\_key

session[:weather][:geo] = @geo = HTTParty.get(google\_url + google\_params, verify: false)

The first two lines of code place the Google Geocoding key into a variable geocoding\_api\_key and assigning the URL for Google Maps into the variable google\_url. The parameters for the search are placed in the variable goggle\_params. At the end of this you can see the API key required by Google. Lastly everything is put together into an instance variable @geo. Using HTTParty, a Ruby Gem (library), the request is returned as a .json file (why .json will be discussed later). This file is then parsed for the desired data.

# Creating the Ruby on Rails Framework and its key elements

## Creating the web framework

The powerful framework of Ruby on Rails is its ability to set up a working web page and database very quickly, within a few short minutes for those who are versed in its setup. The table below contains files automatically created by Rails on setup of a webpage. The most important one is app/ where the models, views, and controllers are contained. The config/ file is another key file for setting up the applications.

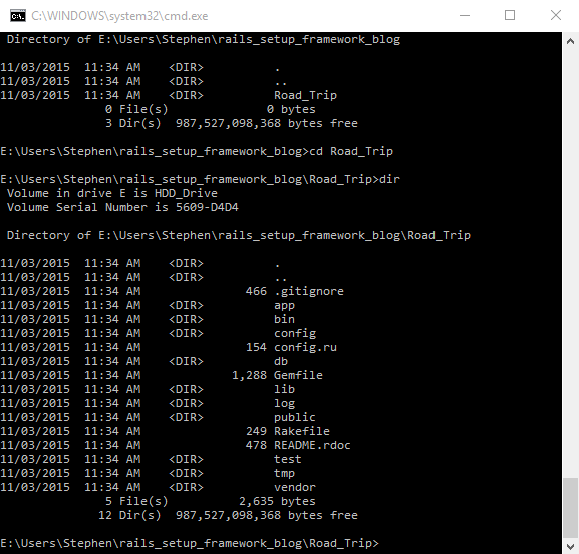
Table 1: Default files automatically generated on creation of a Rails application[[3]](#footnote-3)

| **File/Folder** | **Purpose** |
| --- | --- |
| app/ | Contains the controllers, models, views, helpers, mailers and assets for your application. You'll focus on this folder for the remainder of this guide. |
| bin/ | Contains the rails script that starts your app and can contain other scripts you use to setup, deploy or run your application. |
| config/ | Configure your application's routes, database, and more. This is covered in more detail in [Configuring Rails Applications](http://guides.rubyonrails.org/configuring.html). |
| config.ru | Rack configuration for Rack based servers used to start the application. |
| db/ | Contains your current database schema, as well as the database migrations. |
| Gemfile Gemfile.lock | These files allow you to specify what gem dependencies are needed for your Rails application. These files are used by the Bundler gem. For more information about Bundler, see the [Bundler website](http://bundler.io/). |
| lib/ | Extended modules for your application. |
| log/ | Application log files. |
| public/ | The only folder seen by the world as-is. Contains static files and compiled assets. |
| Rakefile | This file locates and loads tasks that can be run from the command line. The task definitions are defined throughout the components of Rails. Rather than changing Rakefile, you should add your own tasks by adding files to the lib/tasks directory of your application. |
| README.rdoc | This is a brief instruction manual for your application. You should edit this file to tell others what your application does, how to set it up, and so on. |
| test/ | Unit tests, fixtures, and other test apparatus. These are covered in [Testing Rails Applications](http://guides.rubyonrails.org/testing.html). |
| tmp/ | Temporary files (like cache, pid, and session files). |
| vendor/ | A place for all third-party code. In a typical Rails application this includes vendored gems. |

The first step for a web page is to create a new application by using the, in this case, the Windows command line and typing “rails new Road\_Trip”. Figure 1 represents all of the files, or framework, created by Ruby on Rails for the new webpage. The red arrows point out the Model-View-Controller format. There is a fourth arrow that points to the routs.rb file this is how the model, view, and controller are linked for the webpage. Also of importance is that Figure 2 and Table 1 have the same folders, the table had the definitions and Figure 2 contains the output of running the “rails new <project name> command to begin a new webpage . At this point a web page has been created and a local server needs to be set up to display data.

There are several options for storing and moving the data around. The ones discussed here include JSON, XML, and SOAP. JSON is a less verbose and faster method of storage compared to the others. It supports the ability to translate objects. It is considered simple, predictable and easy to read. It’s down sides are that it is too limiting and that there are possible security issues. XML allows for a broader set of data formats to be modeled. It is human and machine readable. Its tenants are simplicity, generality, and usability across the internet. XML’s focus is on documents but can also be used for other data structures. The criticisms of XML include that it is too verbose and complex. If the data is very structured then it is difficult to map the tree. Simple Object Access Protocol (SOAP) is another option to exchange data across the web and computer networks. It is a XML based protocol that uses the HTTP protocol to transmit its information. It has the same disadvantages as the XML protocol. For this project JSON will be used because of its simplicity and readability.

Figure 1: The auto generated files by Ruby on Rails for a web page



To get a Rails server up and running. WEBrick, a server, comes with Rails automatically. This server is designed for development only. WEBrick is a single threaded and single process application meaning that it is slow. For example if two request come in for processor time the first one there is executed and the second must wait for the completion of the first. Using a production server can handle multiple requests at the same time and thus handle a higher workload that will be encountered in the real world. In the command line “rails server” or “rails s” is used to start the server. Once the server is up and running by going to <http://localhost:3000/>, you can see the default ruby web page. From here additional web pages can be added.

Adding additional pages is as simple as creating a new file in the “Pages” folder and creating a method for the new page in the

Figure 2 Automatically created files

# Putting Together the Weather Data Request

Figure 2 Rails initial framework for a web page

## Linking Google Maps and NOAA

To obtain the weather information desired both Google Maps API and NOAA’s Digital Forecast Database API must be used. First on the Weather page the city name and state are entered into the text boxes. When the “Search” button is clicked

# Conclusion

All of this information can help a traveler navigate to their destination. However there is still the human aspect of interpretation of the data. Does heavy traffic through Sacramento as you drive from Bakersfield to Oregon warrant the change of routes? If you’re driving over Donnar Pass or other Sierra Mountain passes and the weather report says the big blizzard will hit 4 hours after you plan on passing over the mountains is this a guarantee? Can you forgo carrying chains? How old are the still images and do they represent current conditions or conditions that will occur in the future. It may be raining now but when the sun sets and the temperatures drop to below freezing what happens? If this application is made accessible for handheld devices is there enough bandwidth for streaming video and what about areas of the state without data coverage.

A driver and/or their navigator must use their knowledgebase to understand the data returned from this application. The past conditions, present conditions, and future conditions are all important for safe navigation. This application provides some essential data for travel but it is no substitute for a drivers judgement of the conditions past, present, and future. Many drivers no longer listen to the radio as the travel using their smart phones or IPods to listen to music, podcasts, or eBooks. They are not tuned to a local radio station that would normally transmit significant road and weather conditions. Because of this isolation from travelling route data a mobile application is needed.

1. Accessed from <http://ruby-doc.org/core-2.2.0/Array.html#method-i-reverse> on 10/15/2015 [↑](#footnote-ref-1)
2. See Google Maps section 3.3 for more information on how this API works [↑](#footnote-ref-2)
3. Rails Guides Webpage by rubyonrails.org, <http://guides.rubyonrails.org/getting_started.html> downloaded on 11.3.2015 [↑](#footnote-ref-3)