**ASK FINANCE TRACKER**

**Ruby on Rails SWE 610 1(Spring ’16)**

****

**INTERNATIONAL TECHNOLOGICAL UNIVERSITY**

**San Jose**

Date: 08th May, 2016

**Guided by: Submitted by:**

Bhairav Mehta Ankita Parekh (82995)

Kriti Mathur (89812)

Simmy Payyappilly Varghese (86448)

Table of Contents

[GitHub Link: 3](#_Toc450475645)

[Heroku Link: 3](#_Toc450475646)

[The Home Page 3](#_Toc450475647)

[Feature#1 Addition of Company Logo: 3](#_Toc450475648)

[Feature#2 Addition of Background image 4](#_Toc450475649)

[Feature#3 Addition of Reset buttons 5](#_Toc450475650)

[Feature#4 Social Share buttons 7](#_Toc450475651)

[Feature #5 The Carousel Plugin 8](#_Toc450475652)

[Feature#6 Fonts and other little things 9](#_Toc450475653)

[My Accounts Page 9](#_Toc450475654)

[Feature# 7 Addition of Accounts Table: 11](#_Toc450475655)

[Feature# 8 Association of User and Account Models: 13](#_Toc450475656)

[Feature# 9 Form Creation: 15](#_Toc450475657)

[Feature#10 Error Page Creation: 16](#_Toc450475658)

[My Expenses Page 18](#_Toc450475659)

[Feature#11 Addition of monthly expense tracker table 18](#_Toc450475660)

[Feature# 12 Addition of form to enter income and expenses 18](#_Toc450475661)

[Feature# 13 Created show details, edit and delete button 19](#_Toc450475662)

[Feature#14 Charts generated to view income, expenses and savings 20](#_Toc450475663)

[My Portfolio Page 23](#_Toc450475664)

[Feature#15 Addition of columns to existing model “stocks” of my\_portfolio page. 23](#_Toc450475665)

[Conact Us Page 24](#_Toc450475666)

[Feature#16 Created the contact us page. 24](#_Toc450475667)

[Feature#17 Addition of Forgot Password 25](#_Toc450475668)

[REFERENCES: 26](#_Toc450475669)

NEW FEATURES OF THE FINANCIAL TRACKER

# GitHub Link:

<https://github.com/ror-project/finance-tracker>

# Heroku Link:

<https://salty-eyrie-77537.herokuapp.com/users/sign_in>

# The Home Page

The original page had a simple home page with no text or images. We had a number of features on home page to make it more user friendly and attractive. We gave an introduction to our application describing the main utility of the app for the user.



# Feature#1 Addition of Company Logo:

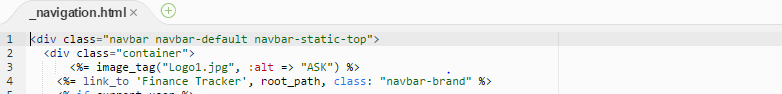
We named the application ASK Finance Tracker where ASK contains first letter from the names of all team members. The logo was designed using online tool and added using image tag in the application.



The title and thus logo needs to be placed in application.html.erb. It uses a helper file to display the title navigation is \_navigation.html.erb.

To render image in the navigation bar, we added the code:

*<%= image\_tag("Logo1.jpg", :alt => "ASK") %>*

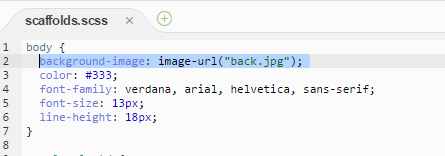


# Feature#2 Addition of Background image

The background image is similar for all the website pages. The image is kept in app/assests/images/back.jpg.

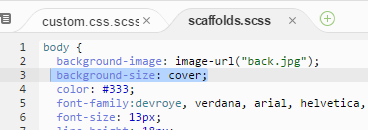
Then the below code is added to styling of body tag in /app/assests/stylesheets/scaffolds.scss :

*background-image: image-url("back.jpg");*



To make the background image fit to page, we added below code:

*background-size: cover;*



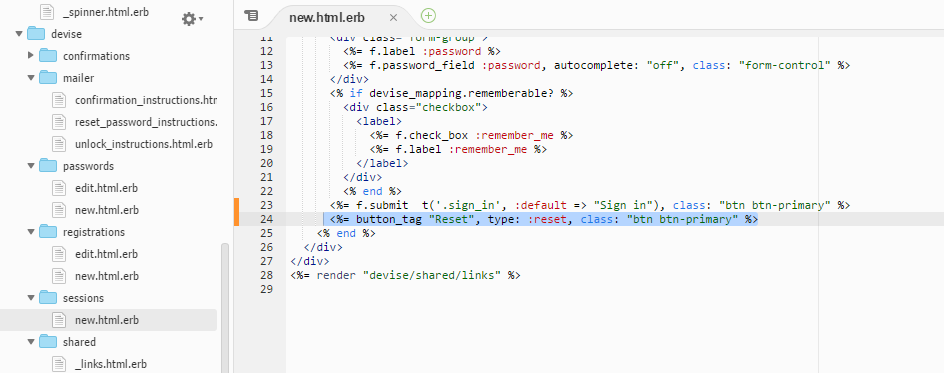
Feature#3 Addition of Reset buttons

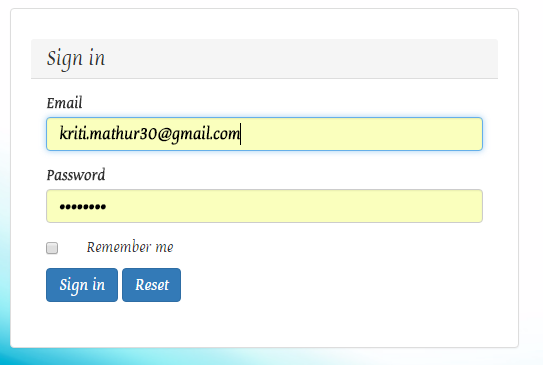
We saw that the reset button was missing from all the form. Reset button will enhance the functionality of the form and increase usability.

1. To add Reset button on Sign In Page

Add below line on views/devise/sessions/new.html.erb file:

*<%= button\_tag "Reset", type: :reset, class: "btn btn-primary" %>*

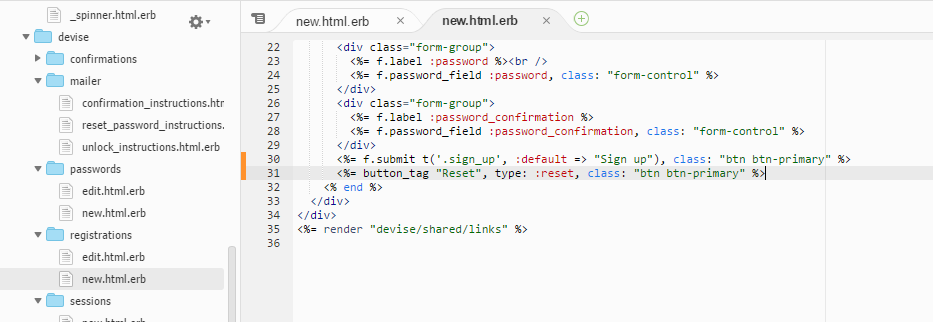


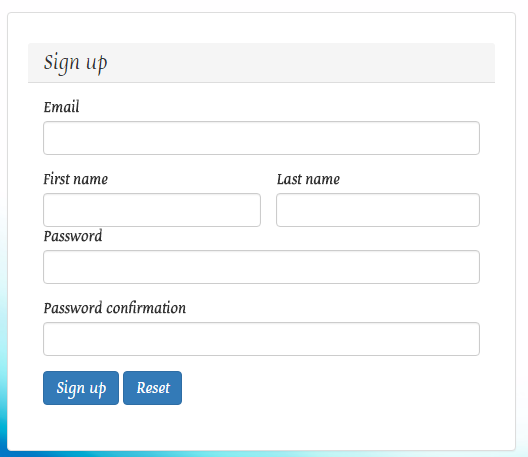


1. Add reset button on Sign Up page

Add below line on views/devise/registrations/new.html.erb file

*<%= button\_tag "Reset", type: :reset, class: "btn btn-primary" %>*





1. Add reset button on Edit User page

Add below line on views/devise/registrations/edit.html.erb file

*<%= button\_tag "Reset", type: :reset, class: "btn btn-primary" %>*

# Feature#4 Social Share buttons

For this we used gem shareable. (<https://github.com/hermango/shareable>)

Shareable gem can be used easily for adding social buttons to your Rails app. This gem runs with minimal configuration and offers many options for personalizing your social buttons. You can generate javascript buttons or static links. Render buttons for popular social media sites and your own custom code with a single method call.



1. Put this line in your Gemfile:

gem 'shareable'

1. To render social share buttons

To render button in the footer, add below code to footer in views/layouts/application.html.erb

*<nav>*

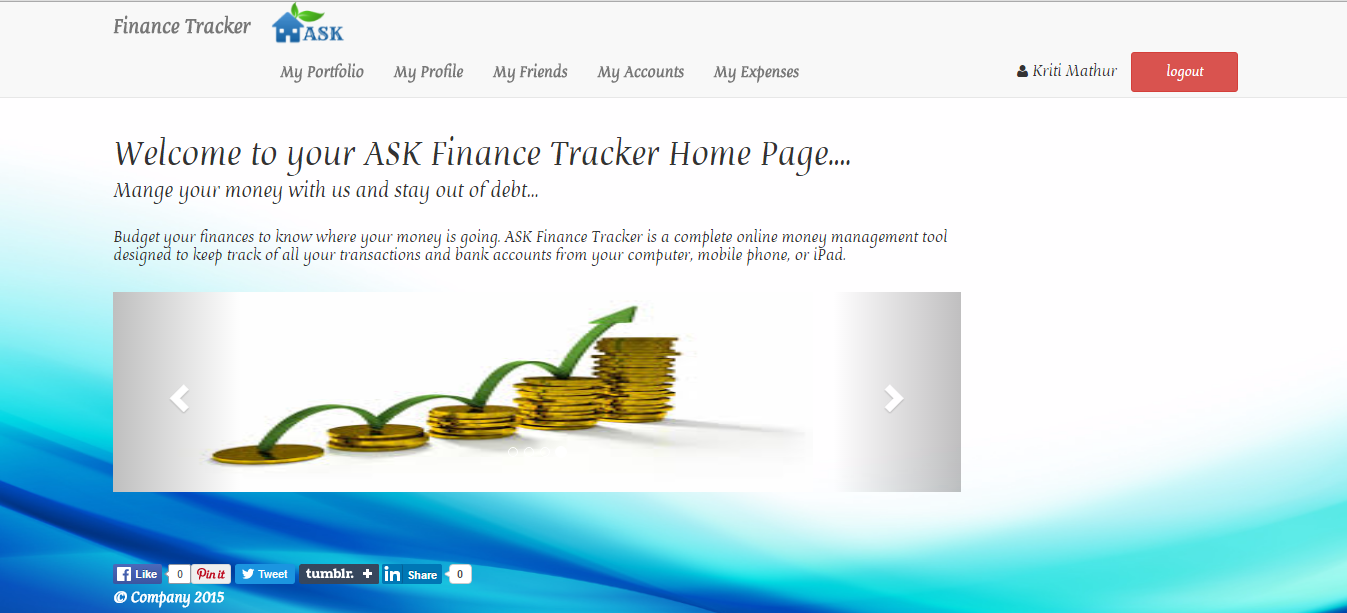
*<%= render\_shareable :buttons=> ['pinterest', 'twitter', 'facebook', 'tumblr', 'linkedin'] %>*

*</nav>*

1. Add the <nav > tag styling in scaffold.scss.css to format the display the buttons in one line.

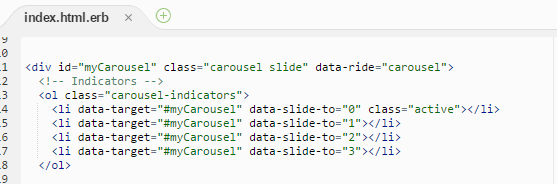
# Feature #5 The Carousel Plugin

We added an image Carousel on the home page. The Carousel images to assets/images/ folder.



To implement this, add a bootstrap code in the app/views/devise/welcome/index.html.erb.

We have used class= ‘carousel-slide’.



The class ‘carousel-inner’ is used to define the item within carousel. The class ‘item active’ is used to describe item that will appear first.

# Feature#6 Fonts and other little things

# My Accounts Page

My Accounts Tab will allow user to create and maintain accounts. This page will feature a form for adding a new account. Form will have the following fields:

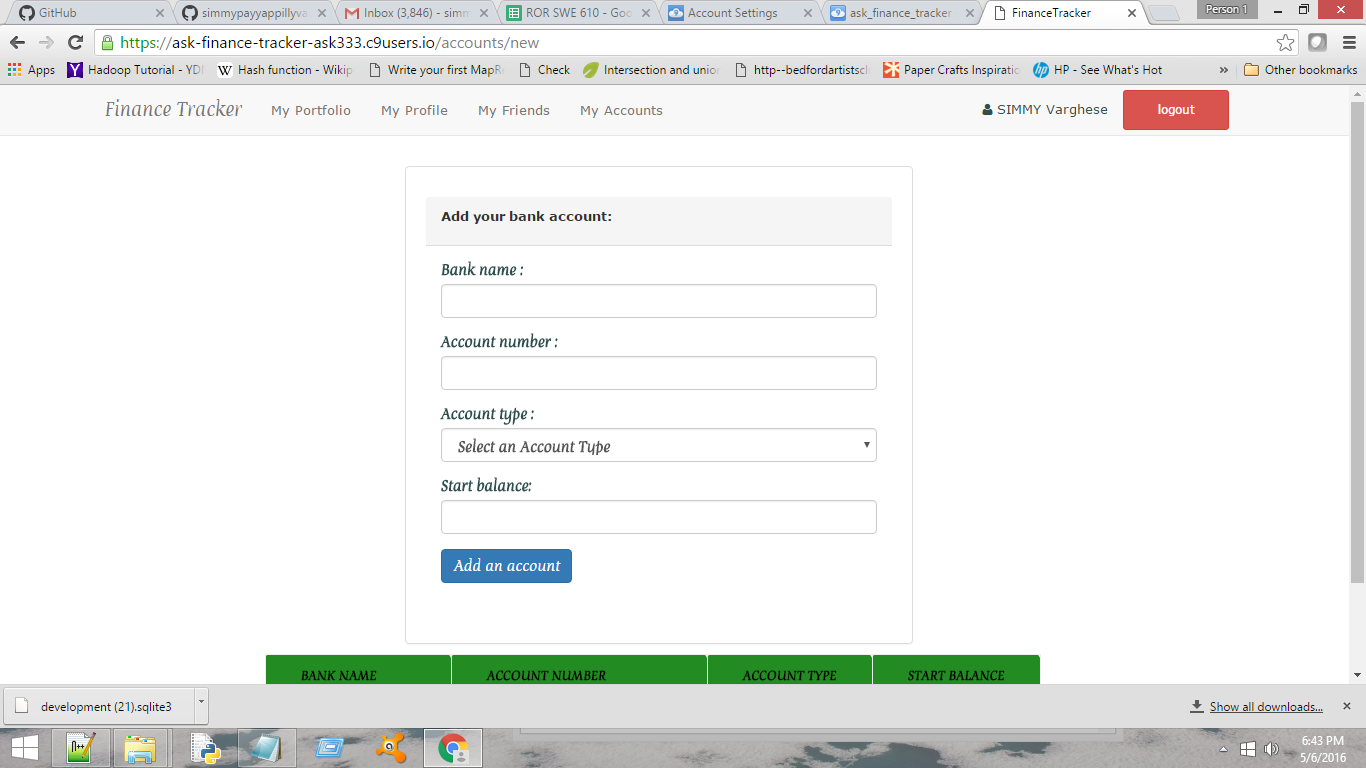
**Bank Name**: This will be the name of the Bank which hold the user’s account. It is a mandatory field

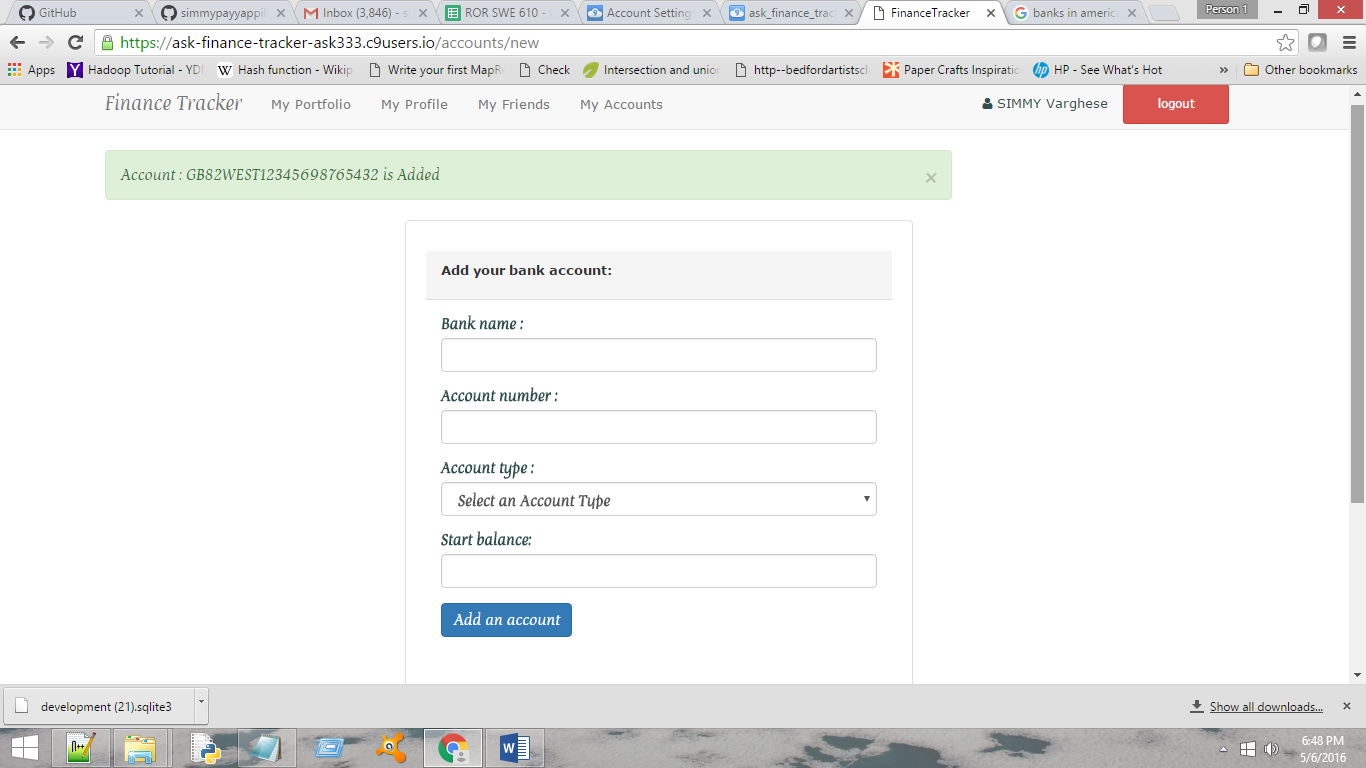
**Account Number**: This will be the bank account number. It is a mandatory field. It shouldn’t be longer than 34 characters. It should be a valid account number

**Account Type:** This will be the type of Bank Account. It is a mandatory field and should select be either Checking, Saving, Credit Card or Cash.

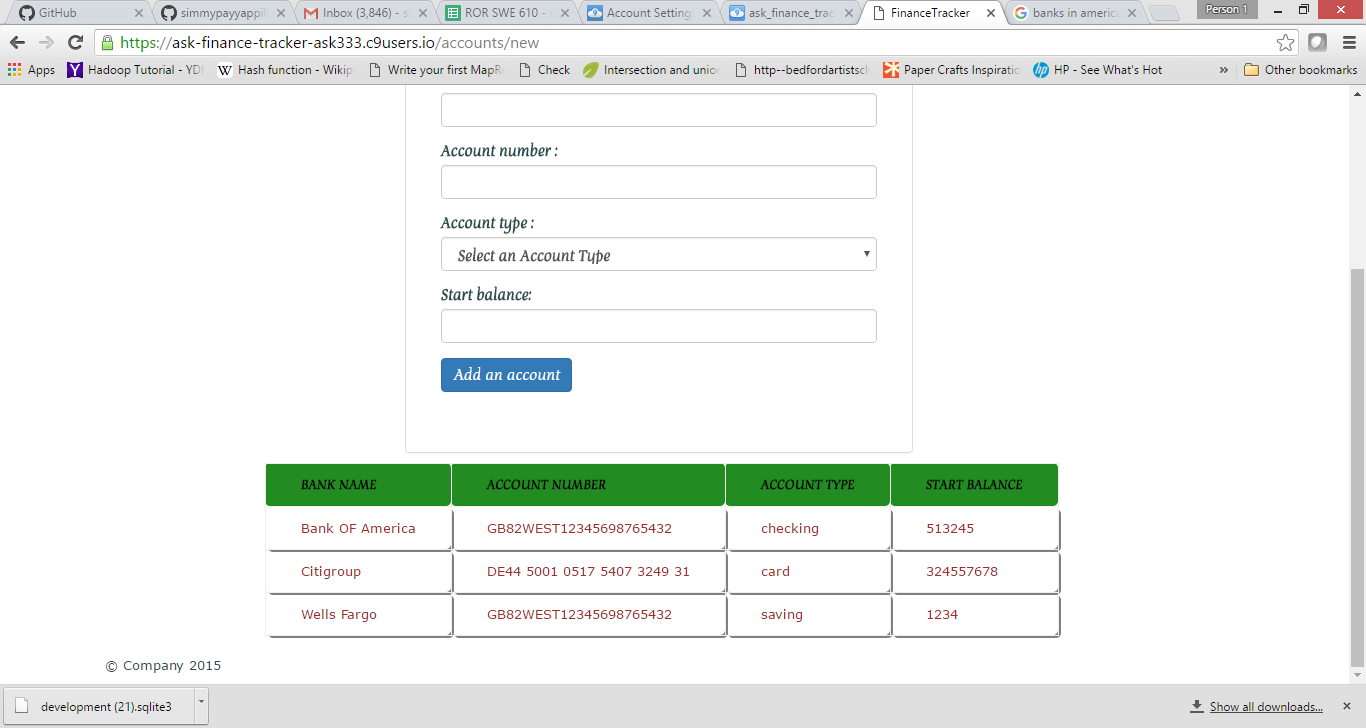
**Start Balance:** This is the starting balance of User’s account.IT is an optional field. It should be numeric.

**Add an Account:** This button will allow the user to create an account if all the entered fields are valid. User can view all the details of the account in a tabular format in the same page.





Flash Message Displayed when an account is created



My Accounts Page-Before and after adding the accounts

# Feature# 7 Addition of Accounts Table:

In order to store the Accounts data for each user, Account Model and database tables created with columns id (automatically generated), bank\_name to store the bank name of the user’s account, account\_tye to store the account type (saving, checking, credit card, cash), start\_balance to store the starting balance of the account, user\_id to store the user’s id and created\_at and updated\_at is automatically created which will store the creation and update date.

**STEPS FOLLOWED:**

Generated a controller named Accounts with new and create actions. Run the following command

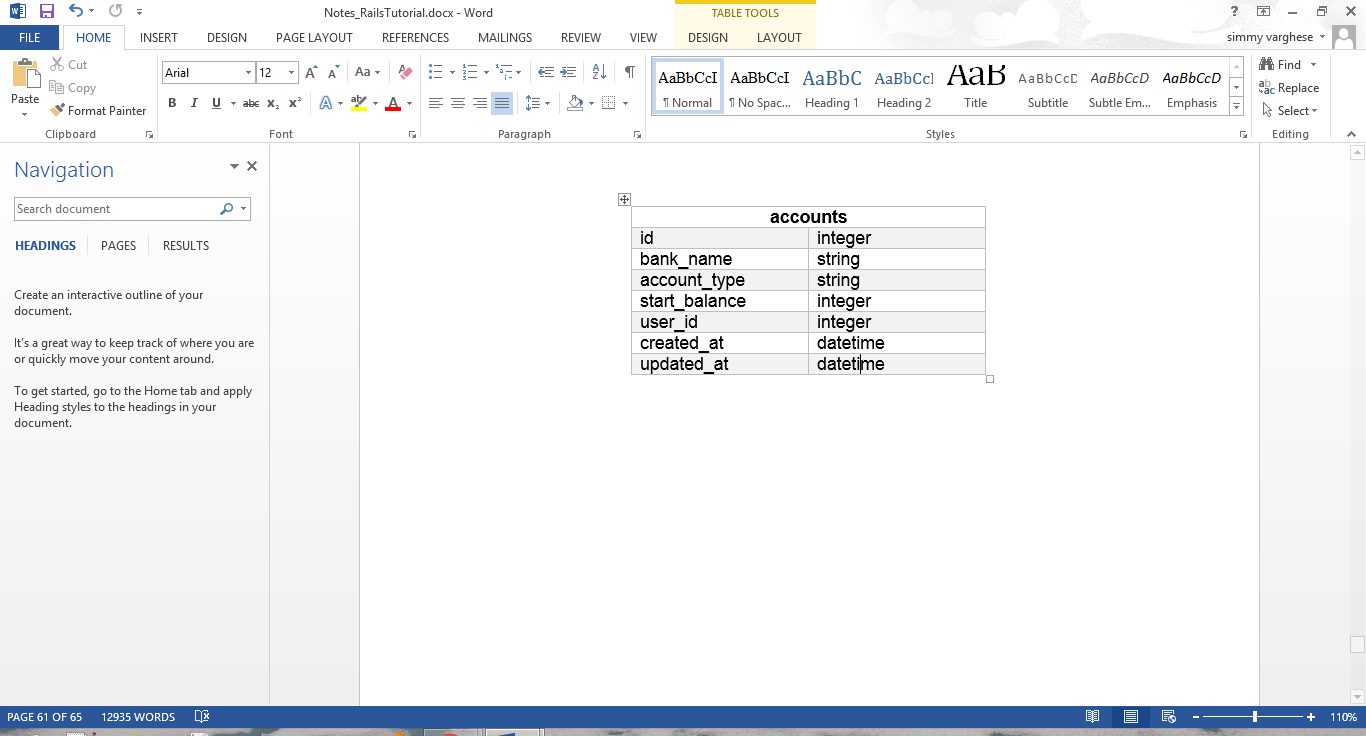
**>>rails generate controller Accounts new create**

Create a data structure to store the account Information. Generate data model named as Account with columns bank\_name (string), account\_number(string), account\_type(string), start\_balance(integer) and user\_i(integer). Use the following command

Add resources: accounts to config/routes.rb file. This will automatically handle the routing of the requests. It will ensure the, application responds to the RESTful URLs. In particular, it ensures that a POST request to /users is handled by the create action.

**>>rails generate model Account bank\_name:string account\_type:string start\_balance:integer user\_id:integer account\_number:string**

Above command creates a table accounts for storing the user’s accounts with specified columns along with columns created\_at and updated\_at which are automatically created.user\_id column will act as the foreign key to associate with the user table (already created).And whenever a user creates an account, column is updated with the respective user\_id

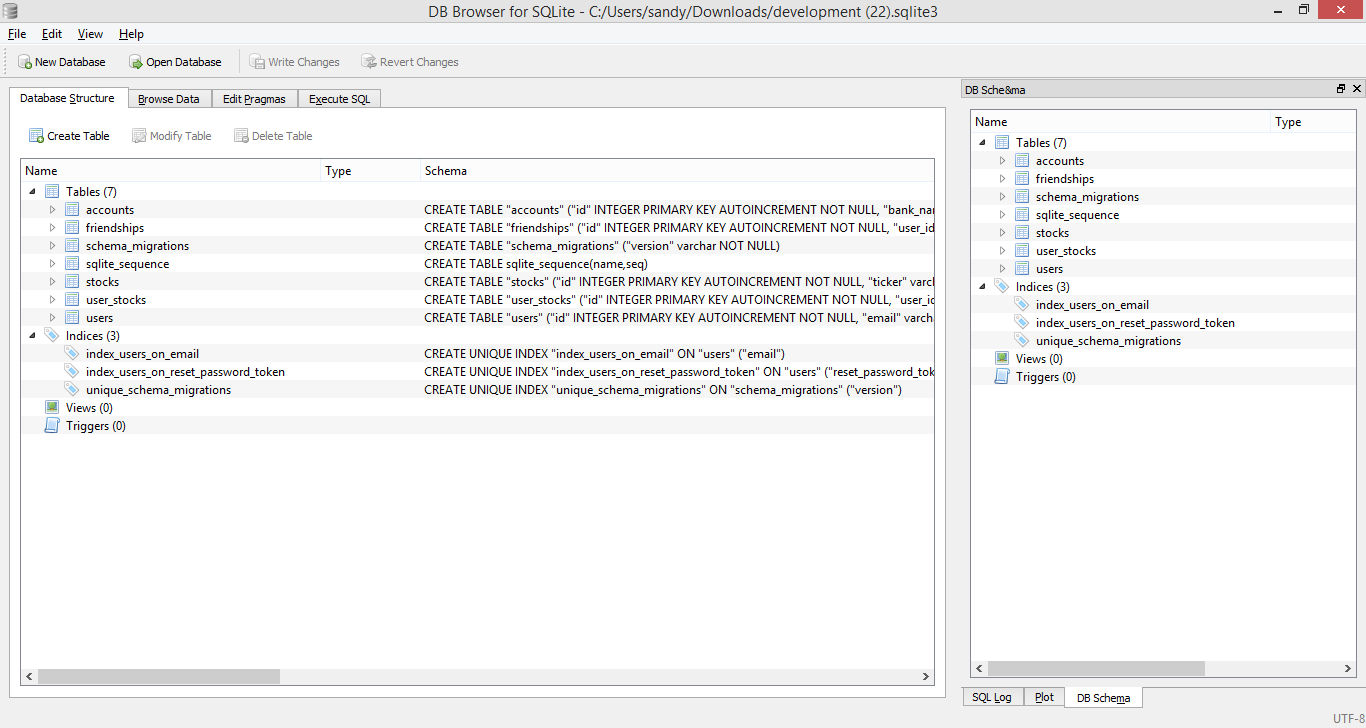


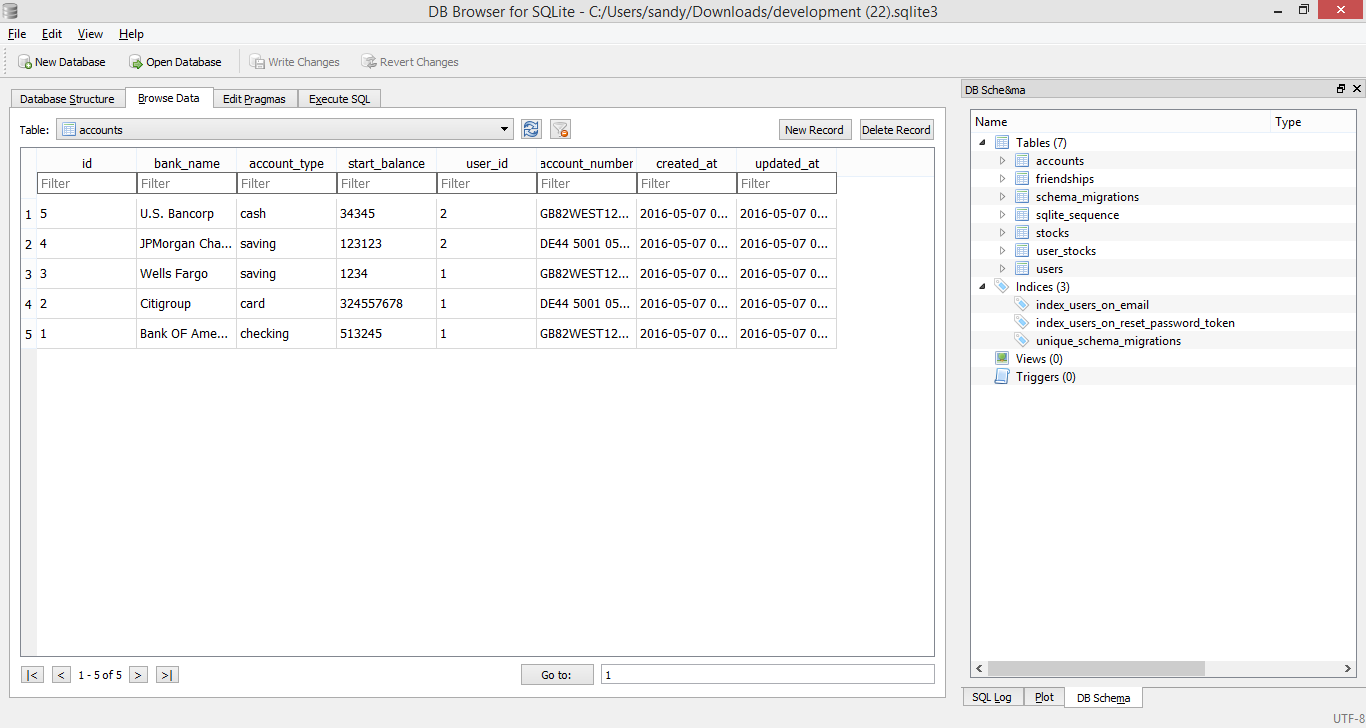
The Account Data Model

Once the model is generated, run the following command to update the database.

**>>bundle exec rake db:migrate**

First time this command is executed, it will create the development.sqlite3 file, if SQLITE3 Database is used. To view the table download the file and open in DB Browser for SQLITE

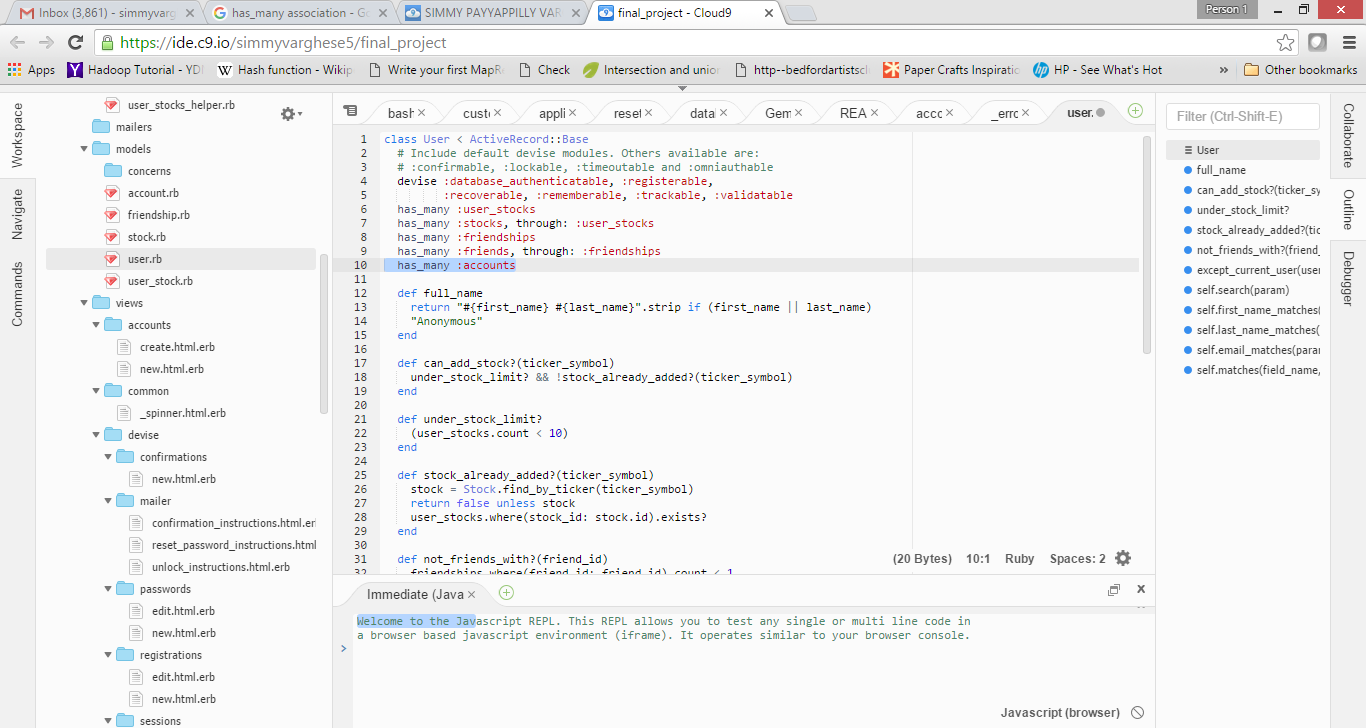




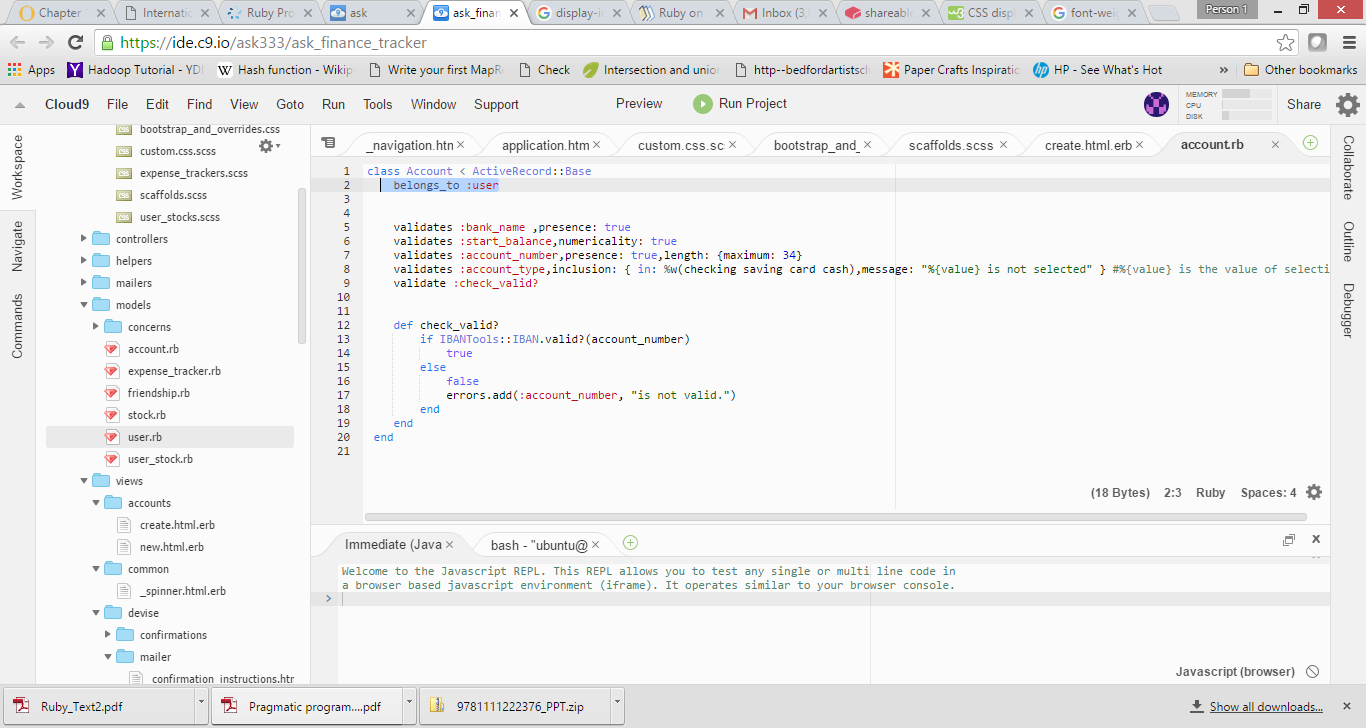
Above Snaps displays the accounts and other added for each users (specified by user\_id column)

# Feature# 8 Association of User and Account Models:

Created association between the User and Account Models. User can have many accounts and each account will have a User, this creates a has-many association. This association says that the other class will have a foreign key that refers to instance of this class. This is done by adding has\_many:accounts to the user.rb file. Similarly each account can have only one use which leads to a one to one association. A belongs\_to association sets up a one-to-one connection with another model, such that each instance of the declaring model "belongs to" one instance of the other model. This is done by adding belongs\_to:user in accounts.rb file. When an association is created , Rails makes two major assumptions -- first, that the class of the model your association points to is based directly off of the name of the association, and, second, that the foreign key in any belongs\_to relationship will be called yourassociationname\_id. This could be overridden using :foreign key

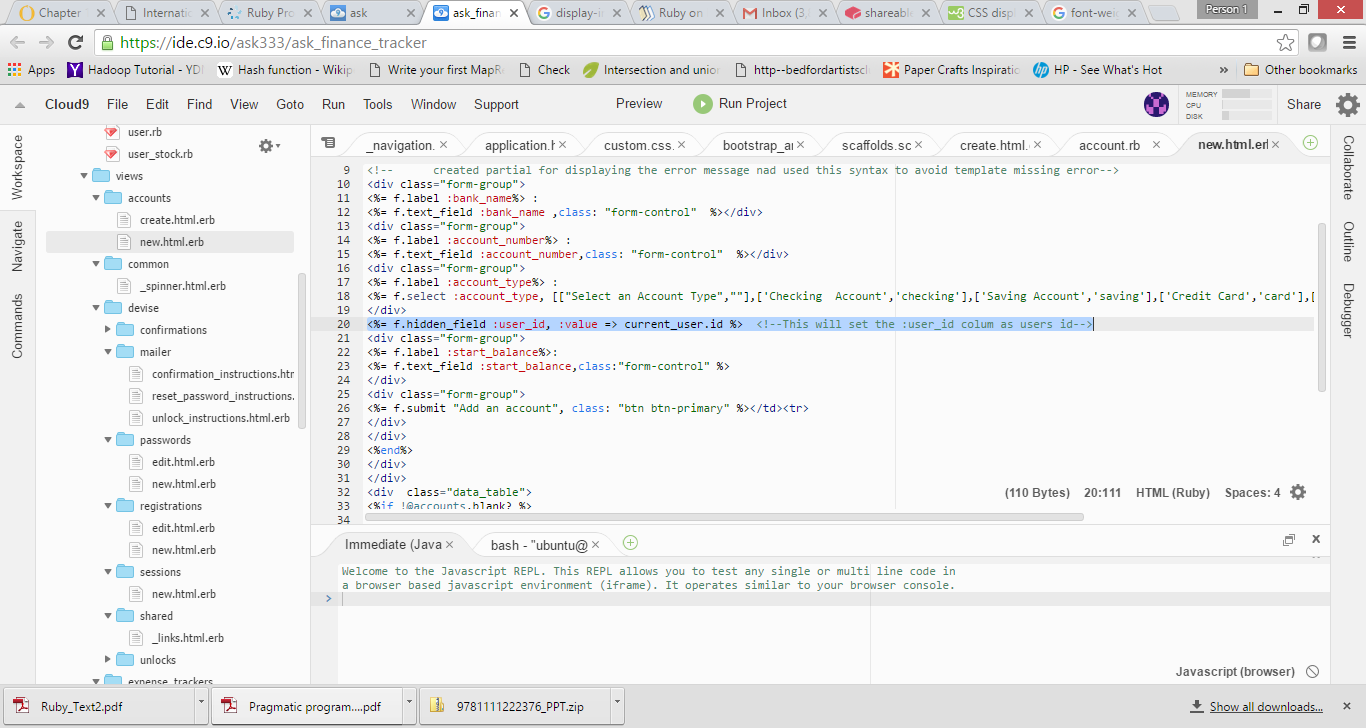


Above image shows has\_many association creation



Above image shows belongs\_to association creation

By creating the has\_many association we have set the foreign key and it is user\_id. In order to get the user\_id of Accounts table updated with id from User table, hidden field is created in the form and on creation of account the hidden field set the user\_id with current user id from the Users table (already exists) as shown below:



# Feature# 9 Form Creation:

**Form Creation and Validations:**

Using HTML 5 form is created for the accounts in new.html.erb .This is done through form\_for method, which input the Active Record Object and creates a form using the objects attributes .On submit of the form the form data gets posted to the same page in tabular format. And the HTML table will have four columns: Bank Name, Account Number, Account Type, Start Balance. Validations for the form field are added to the model/accounts.rb file. Validations are used to ensure that only valid data is saved into your database.

**Bank Name:** Bank Name shouldn’t be blank. Validates method ensure the field is not blank using presence: true property, else throws error “Bank Name can’t be blank”

**Account Number:** Account number shouldn’t be blank and it shouldn’t be longer than 34 characters. Both these properties are verified by validates method using presence: true and length: {maximum: 34}. If the account number is blank, it throws the error “Account number can't be blank” or if it is longer than 34, “Account number is too long (maximum is 34 characters)” error is displayed.

Also Account number should be a valid number. An IBAN (International Bank Account Number) is validated by converting it into an [integer](https://en.wikipedia.org/wiki/Integer) and performing a basic mod-97 operation on it. If the IBAN is valid, the remainder equals 1. The IBAN consists of up to 34 alphanumeric characters, comprising a country code (2 letters), two [check digits](https://en.wikipedia.org/wiki/Check_digit) and a long and detailed bank account-number. Permitted IBAN characters are the digits 0 to 9 and the 26 upper-case [Latin alphabetic characters](https://en.wikipedia.org/wiki/Latin_alphabet) A to Z.

This validation is defined in IBANTools::IBAN.valid? which requires gem 'iban-tools’. Using this we create a custom method that will return true if account number is valid else will add a custom error message “Account number is not valid.” using errors.add(field,erro\_message).

**Account Type:** Account type will have any of the following Checking, Saving, Credit Card or Cash. It is implemented using a drop down. It is validated for inclusion, if user does not select the valid value will throw a custom error message “Account Type is not selected”.

**Start Balance:** It is an optional field. It could have any value but numeric. This is validated using numericality: true

Once the validations are created test it by giving different valid and invalid values for the field. To display the error messages, we created a partial (shared/\_error\_messages.html).This file contains the code to display the error if any occurs with full message. It also use pluralize method which will pluralize the message based on the number of error occurred. This page will have a Back hyperlink to navigate back to the page.

Controller ensure that through the create method, whenever the save operation fails on account object, to render the partial to display the error message .And if the save operation is successful will redirect back to the new file containing form. Also the field values gets displayed in the HTML table. User id of current user is fetched and is updated in user\_id column of Accounts table.

Once all the fields are entered and form submission is successful, we can view and verify the entries are saved to database with the help of SQLITE BROWSER as shown below.

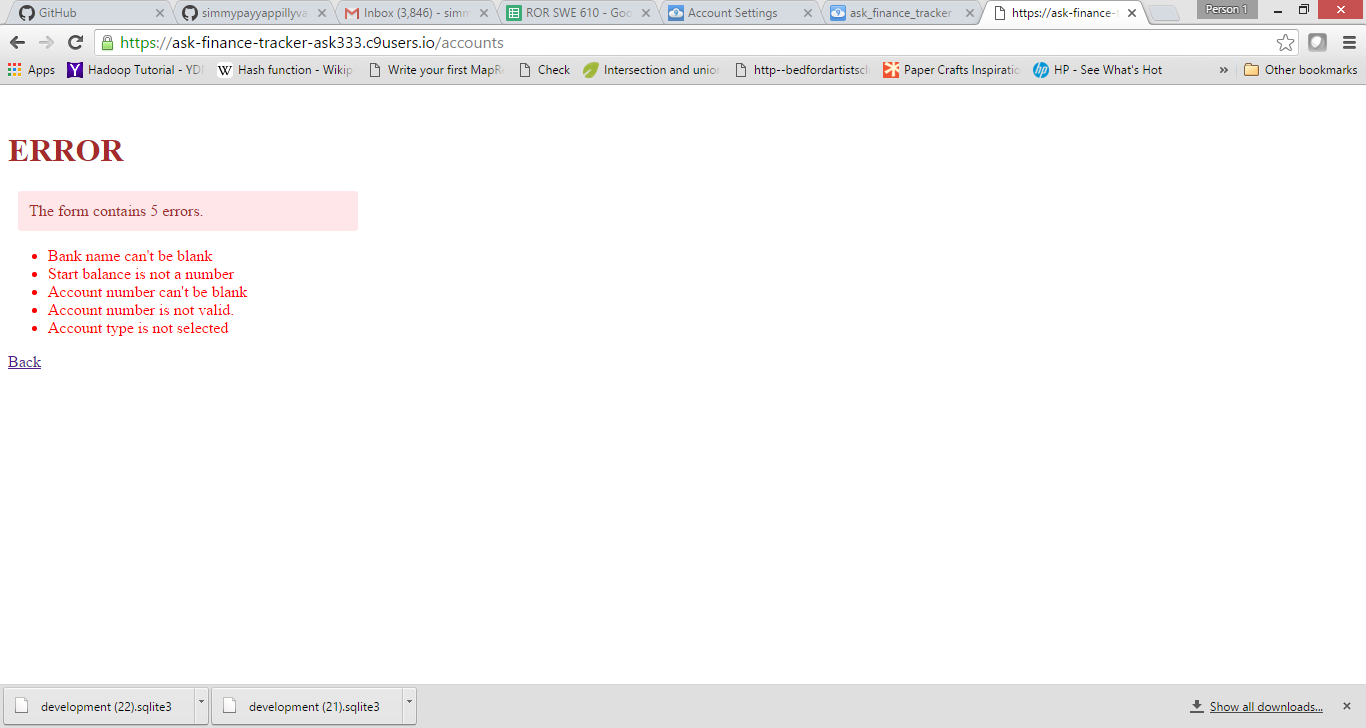
**FORM AND PAGE STYLING**

In order to style the page CSS3 and boot strap classes are used. To support bootstrap class include the gem ‘bootstrap-sass’ in the Gemfile and run the command bundle install. Also include the commands @import "bootstrap-sprockets";@import "bootstrap"; in the CSS file to modify the page. Custom fonts ‘DEVROYE ’ is added as the default font .For this add the font file to fonts folder created in assets and update the config/application.rb file with the following:

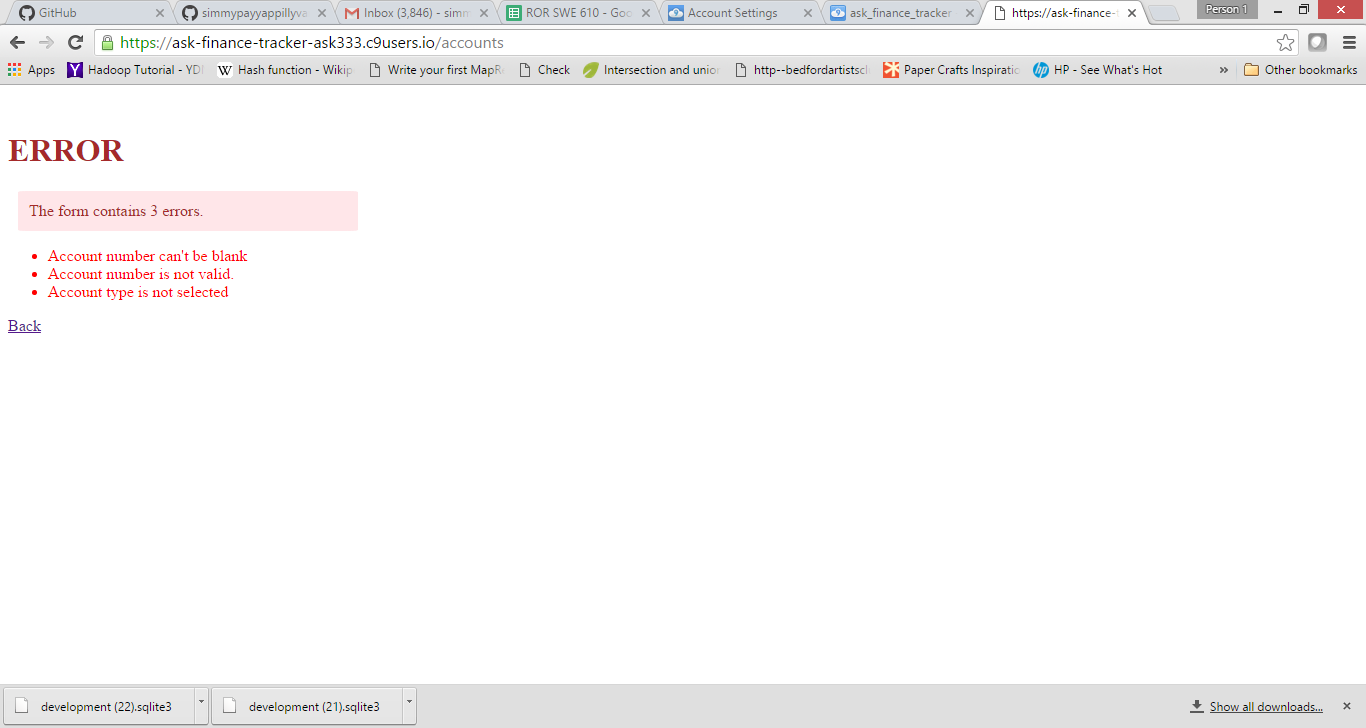
code:config.assets.paths << Rails.root.join(‘app’,’assets’,’fonts’) and config.assets.precompile << /\(?:svg|woff|ttf)$/.

# Feature#10 Error Page Creation:

In order to display the error message, new page \_error\_messages.html.erb is created in shared folder (newly created).Used HTML and CSS to style and create the page. If any validation of form field fails,the corresponding error defined in the accounts.rb file will be displayed



Error page with all field validation error



Account Number and Type Validations

**COMMANDS USED**

To generate the Controller: rails generate controller Accounts new create

To delete the Database Entries: Account.delete\_all

To update the database: bundle exec rake db:migrate

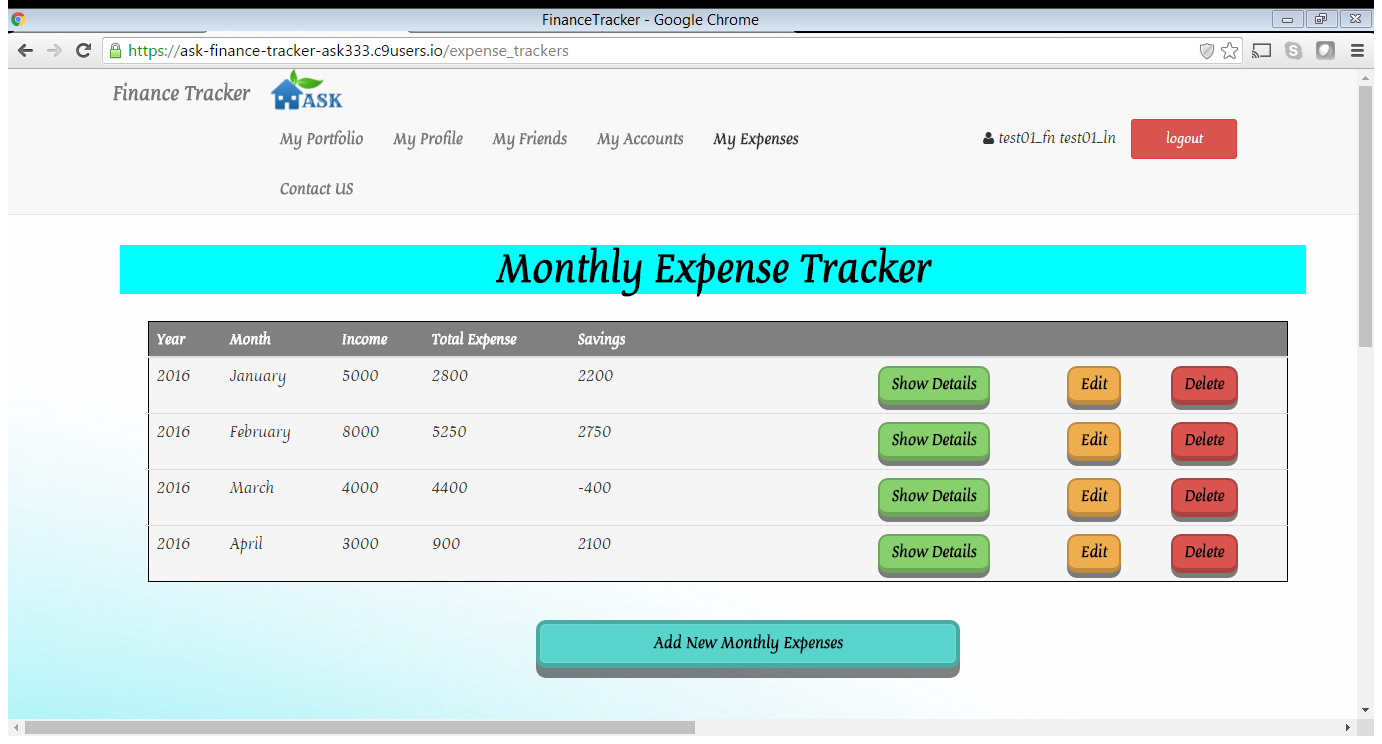
To generate model :rails generate model Account bank\_name:string account\_type:string start\_balance:integer user\_id:integer account\_number:string

To know the routes: rake routes

# My Expenses Page

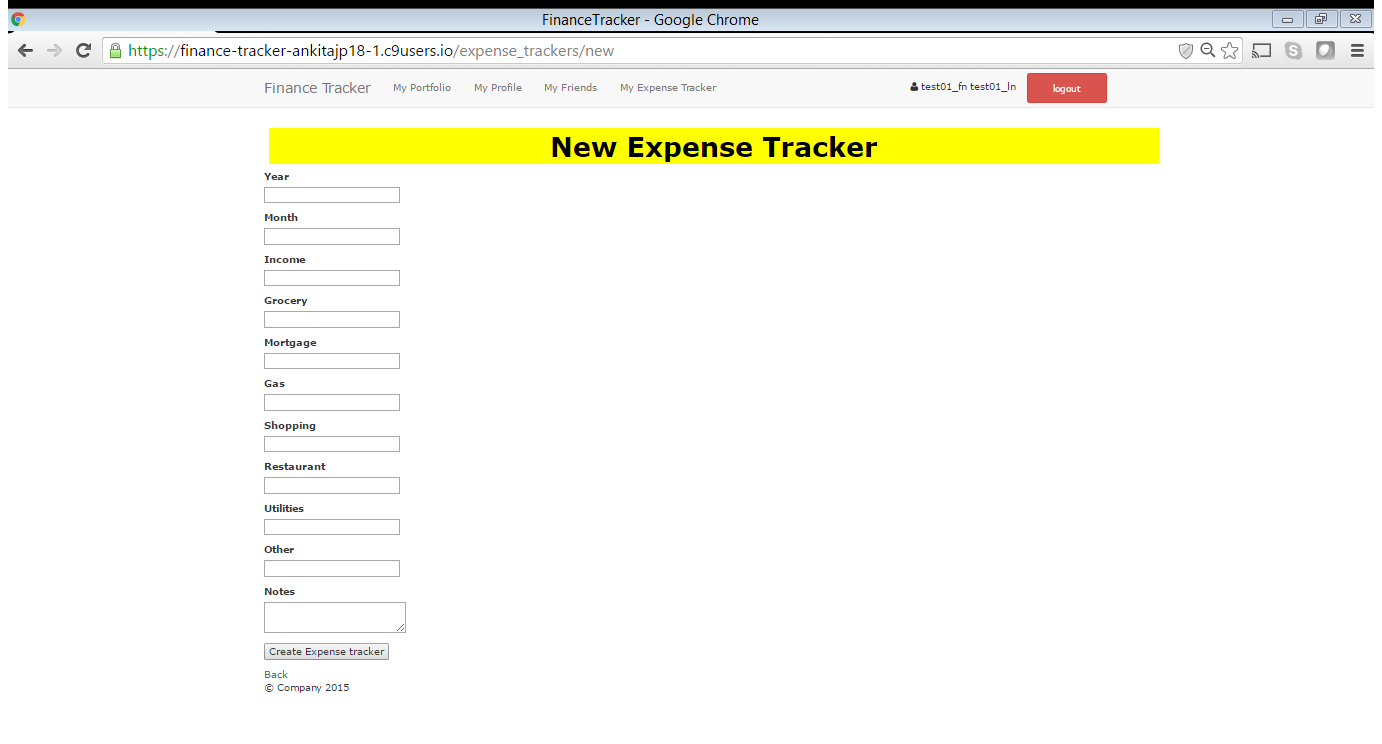
My monthly expense tab will allow users to keep track of their income, expenses and savings.

# Feature#11 Addition of monthly expense tracker table

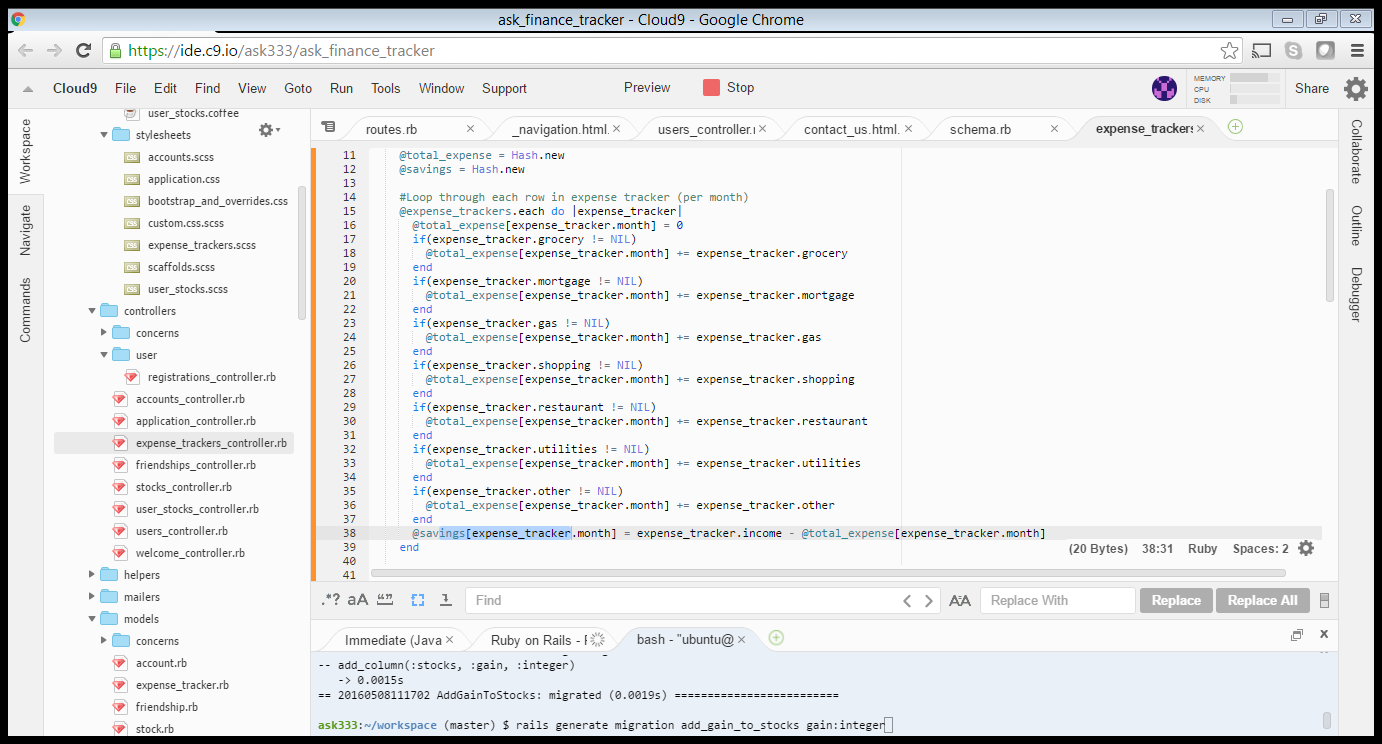


# Feature# 12 Addition of form to enter income and expenses

It allows user to enter each month’s income and expense details for various categories like Grocery, restaurant, shopping, gas, travel etc.

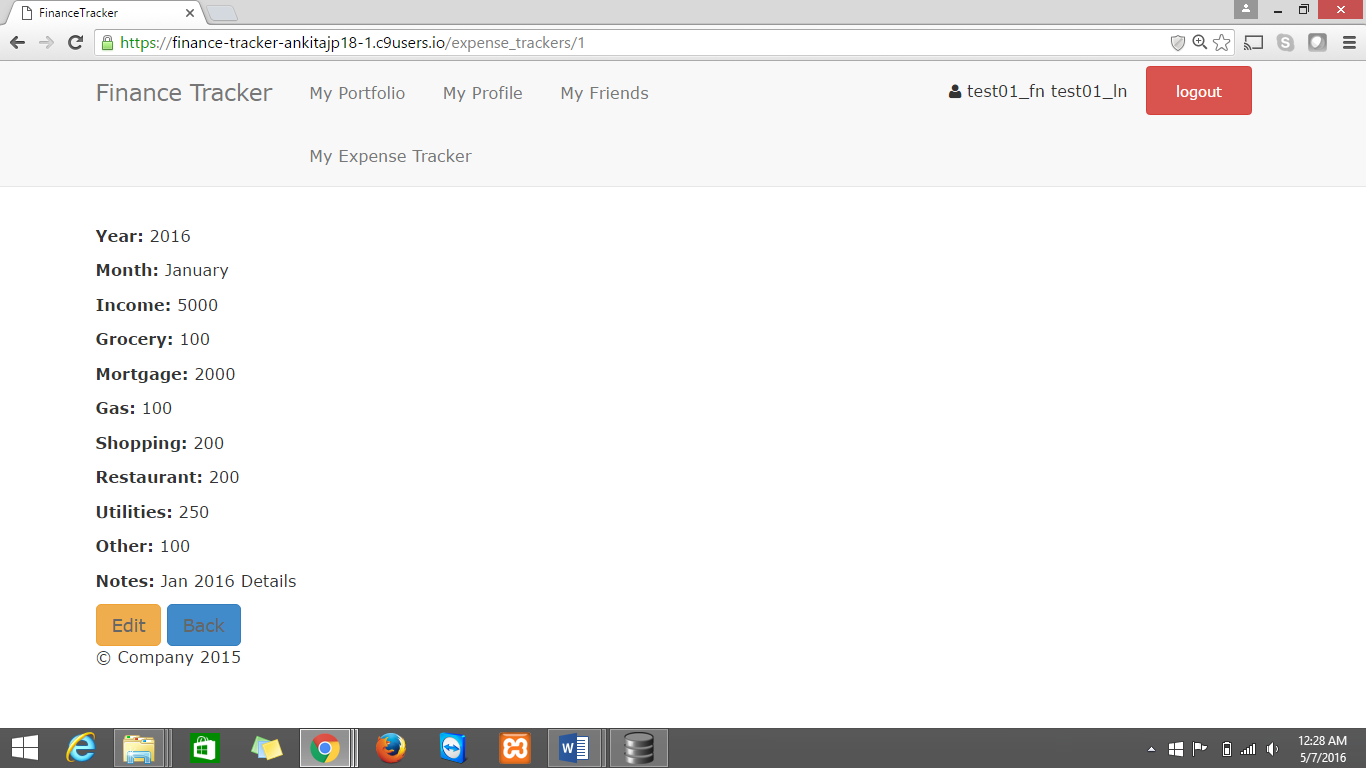


Once the user clicks on “create expense tracker” it will populate the database and also calculate the total expenses and report savings and will display it on the main page.



# Feature# 13 Created show details, edit and delete button

**Show details Button** - Displays all detailed expenses based on categories for each month.



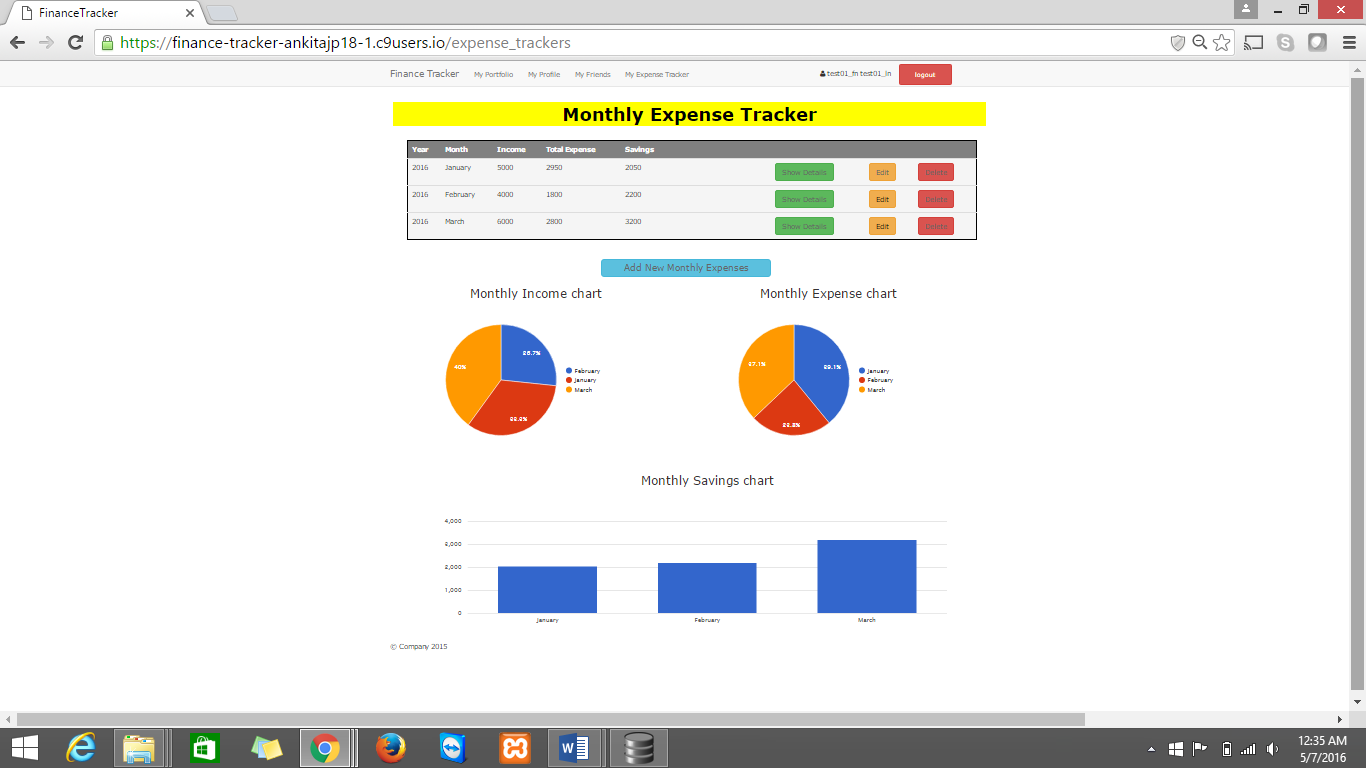
**Edit Button:** It allows user to modify any of the entries

**Delete Button:** It allow user to delete any existing entries

# Feature#14 Charts generated to view income, expenses and savings

This app will also generate following charts:

* Pie Chart for Income
* Pie Chart for Expenses
* Column Chart for Savings



**Steps**

1. Generate scaffold expense\_tracker. Run the following command.

*>>rails generate scaffold expense\_tracker year:integer month:string income:integer grocery:integer mortgage:integer gas:integer shopping:integer restaurant:integer utilities:integer other:integer notes:text*

Each scaffold created is an object of that application. It will create a directory structure of model, controller and view with the specified fields. It generates the basic framework with which user can interact with.

1. Migration of data

>>*rake db:migrate RAILS\_ENV=development*

1. Calculate and display total expenses and Savings.

*Expense\_tracker\_controller.rb*

****

**Index.html.erb**

*<thead>*

*<th>Total Expense</th>*

*<th>Savings</th>*

*</thead>*

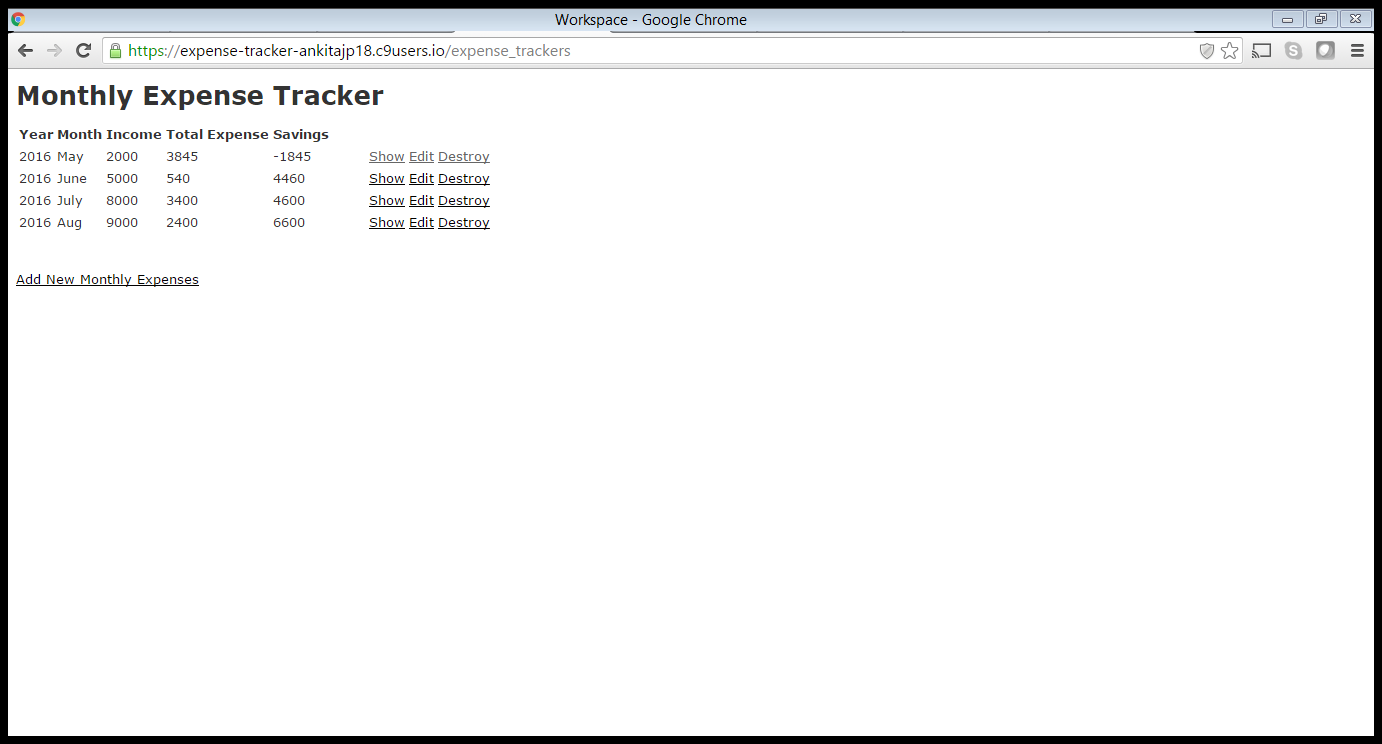
*<tbody>*

*<td><%= @total\_expense[expense\_tracker.month] %></td> <!-- Display hash -->*

*<td><%= @savings[expense\_tracker.month] %></td></tbody>*

*</tbody>*

Screenshot of index page after scaffold and editing of code



1. Generate Chart Using Chartkick library

Chartkick can use either Google Charts or Highcharts for charting. In this example we use Google Charts. We also utilize Bootstrap for a cleaner look and feel.

* Add chartkick.js under javascript
* Add following gem to gemfile

gem 'chartkick', '~> 1.2.4'  
gem 'groupdate', '~> 2.1.1'  
gem 'active\_median', '~> 0.1.0'

* run a bundle install to install all three gems.
* **Application.html.erb**

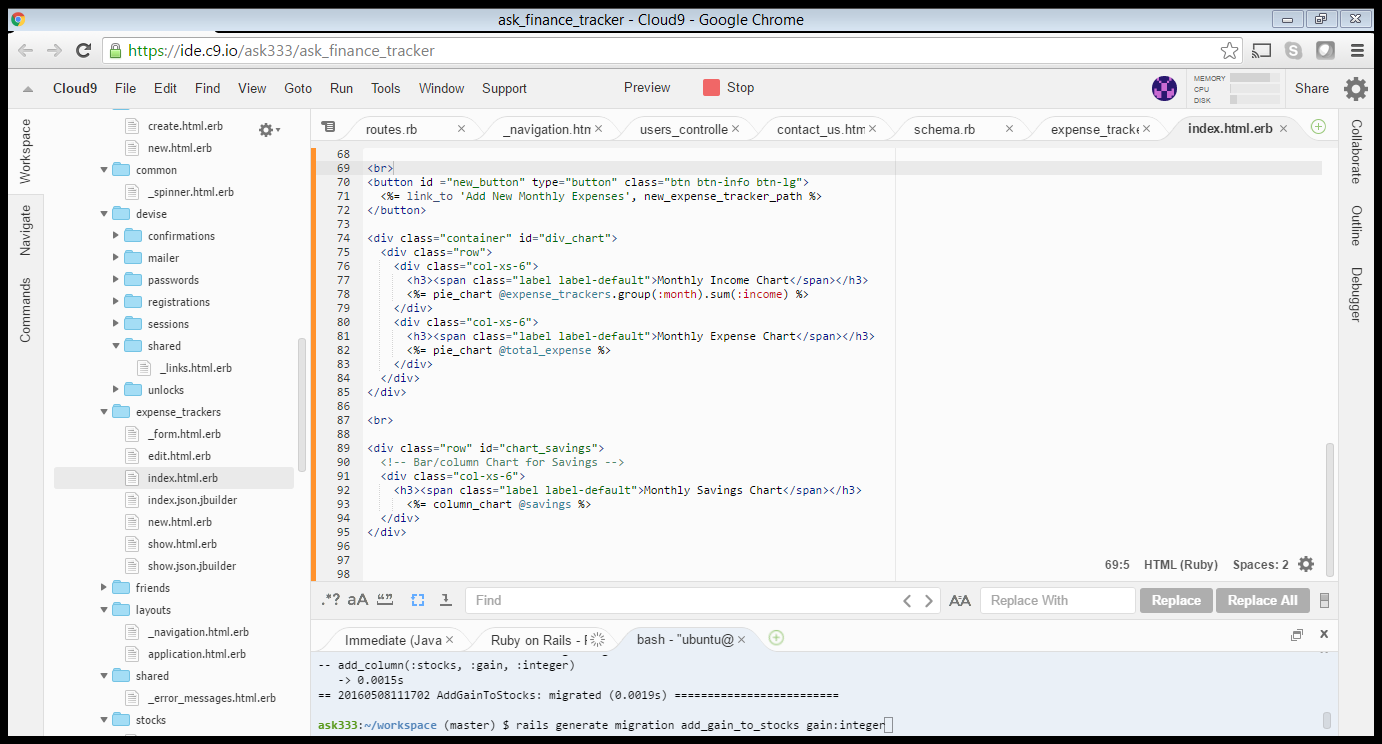
*<%= javascript\_include\_tag "//www.google.com/jsapi" %>*

*<%= javascript\_include\_tag "application", "chartkick" %>*

*<%= stylesheet\_link\_tag 'http://yandex.st/bootstrap/3.1.1/css/bootstrap.min.css' %>*

*<%= javascript\_include\_tag 'http://yandex.st/bootstrap/3.1.1/js/bootstrap.min.js' %>*

* **index.html.erb**



1. Styling with CSS and bootstrap

* Used .table class which will give basic styling to the table
* Added buttons with bootstrap and enhanced with shadows for show details, edit and delete.
* Create new button has an addition feature of transform when hover onto it.
* Highlighting text
* Display pie chart side by side
* Used the .label class for labeling charts generated

Final Layout of My Expense Page



# My Portfolio Page

# Feature#15 Addition of columns to existing model “stocks” of my\_portfolio page.

Used the following commands to add the columns buying price, selling price, total shares and Gain (Profit/Loss) to the existing model “stocks”

*rails generate migration add\_buying\_price\_to\_stocks buying\_price:integer*

*rails generate migration add\_selling\_price\_to\_stocks selling\_price:integer*

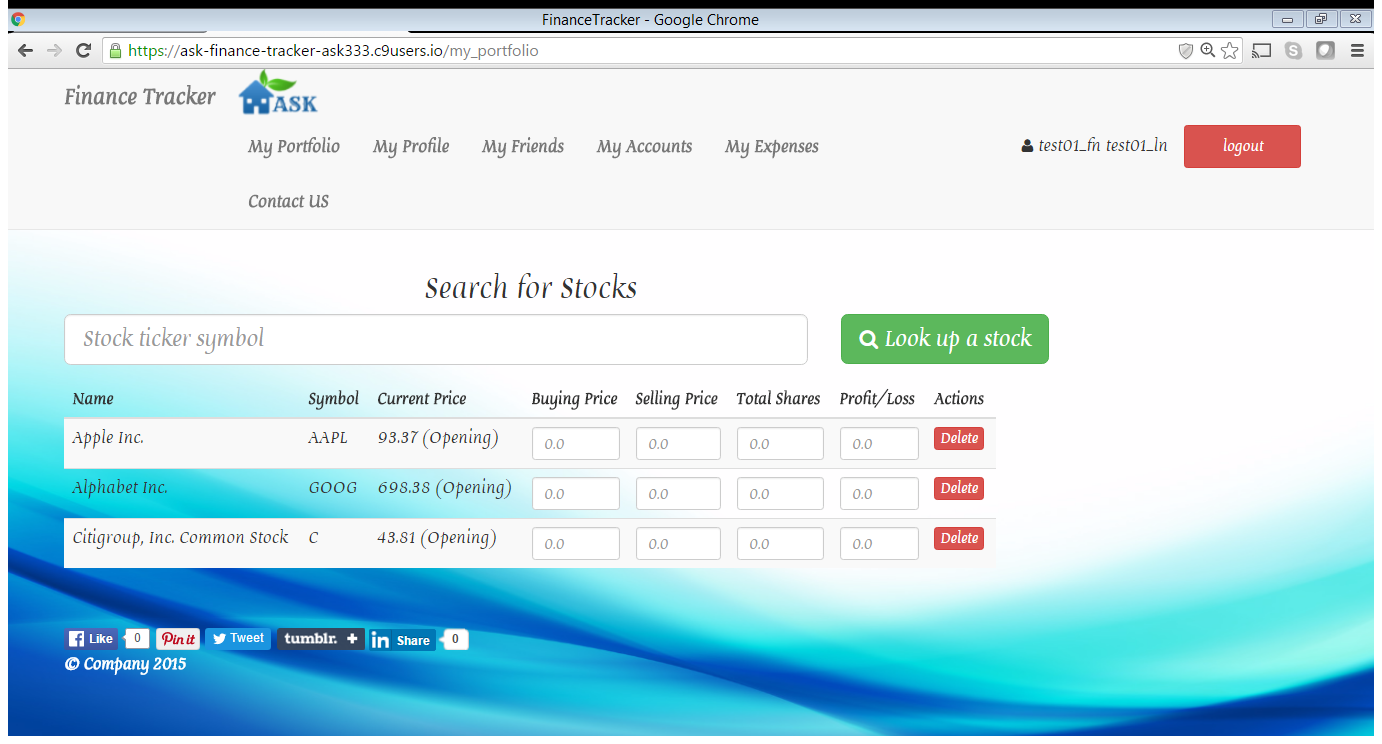
*rails generate migration add\_total\_shares\_to\_stocks total\_shares:integer*

*rails generate migration add\_gain\_to\_stocks gain:integer*

**list.html**

Added more tags to display all the columns buying price, selling price, total shares and gain on the my\_portfolio page

Final Layout of My Portfolio Page



# Conact Us Page

# Feature#16 Created the contact us page.

**routes.rb***get 'contact\_us', to: "users#contact\_us"*

**users\_controller.rb**Update the user controller with contact\_us action   
 **views/users/contact\_us.html.erb**

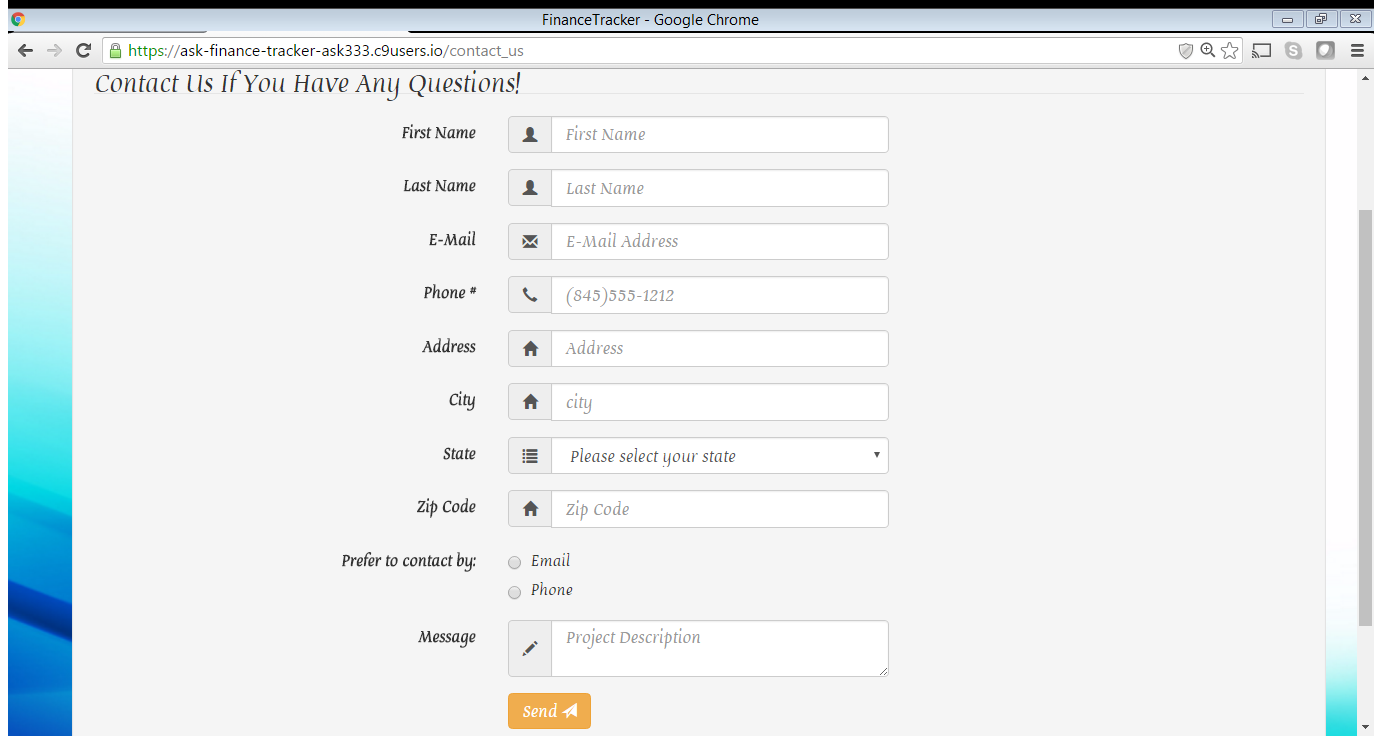
* Created a form where user can ask questions.
* Added following features to the form:

1. Icons for all the fields like users icon.
2. Text fields
3. Radio buttons
4. Drop down menu to select from the list of states for field “states”

**Assets/contact\_us.js**

Added validations for all the fields in javascript

Final Layout of Contact Us Page



# Feature#17 Addition of Forgot Password

Since the application is using devise, we can use the authentication model already present to add this functionality.

1. Add the code in views/shared/\_link.html.erb

*<%- if devise\_mapping.recoverable? && controller\_name != 'passwords' %>*

*<%= link\_to t(".forgot\_your\_password", :default => "Forgot your password?"), new\_password\_path(resource\_name) %><br />*

*<% end -%>*

1. Add below to format the mailer in views/mailer/reset\_password\_instruction.html.erb

*<p><%= t('.greeting', :recipient => @resource.email, :default => "Hello #{@resource.email}!") %></p>*

*<p><%= t('.instruction', :default => "Someone has requested a link to change your password, and you can do this through the link below.") %></p>*

*<p><%#= link\_to t('.action', :default => "Change my password"), edit\_password\_url(@resource, :reset\_password\_token => @token, locale: I18n.locale) %></p>*

*<p><%= t('.instruction\_2', :default => "If you didn't request this, please ignore this email.") %></p>*

<p><%= t('.instruction\_3', :default => "Your password won't change until you access the link above and create a new one.") %></p>

# REFERENCES:

* https://en.wikipedia.org/wiki/International\_Bank\_Account\_Number
* http://stackoverflow.com/questions/2774841/best-way-to-add-comments-in-erb
* http://www.helpitx.com/2015/07/5-stylish-css-table-designs-for-blog.html
* http://stackoverflow.com/questions/30315498/ruby-on-rails-how-to-link-route-from-one-view-page-to-another-view-page-with-a
* http://v4-alpha.getbootstrap.com/components/forms/#form-controls
* http://stackoverflow.com/questions/22111088/how-do-i-use-the-bootstrap-form-select-css-with-a-rails-model
* https://rubygems.org/gems/iban-tools/versions/1.1.0
* http://guides.rubyonrails.org/active\_record\_validations.html
* https://metova.com/blog/dev/font-face/
* http://stackoverflow.com/questions/17721540/
* http://edgeguides.rubyonrails.org/active\_record\_querying