Table of Contents

[1. Initialize Git hub - Version Control: 1](#_Toc40289059)

[2. Roots and Login Routes: 3](#_Toc40289060)

[3. Install Semantic UI 4](#_Toc40289061)

[4. Add Navigation menu 5](#_Toc40289062)

[5. Create navigation partial and enable dropdown functionality 8](#_Toc40289063)

[6. Add favicon 8](#_Toc40289064)

[7. Build Homepage 9](#_Toc40289065)

**Cmd:**

To check versions,

rails -v

ruby -v

rails new message\_me // create app

ls // list of all files in that folder

cd message\_me

rails s and go to http://localhost:3000/ => to check if all the files are created properly in the app

# Initialize Git hub - Version Control

Create a repo for the msg\_me app. While creating, do not check the readme at the bottom.

**First time GIT initialization:**

git init

git add -A

git commit -m "Initial Commit"

git remote add origin https://github.com/LogaSangeerani/message\_me.git

git remote -v

git push -u origin master

git status



**Usual push to repo:**

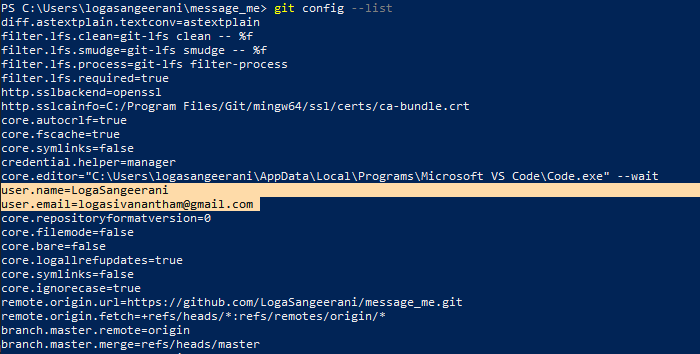
git add -A

To commit, git commit -m "Message"

git push origin master

**Git commands:**

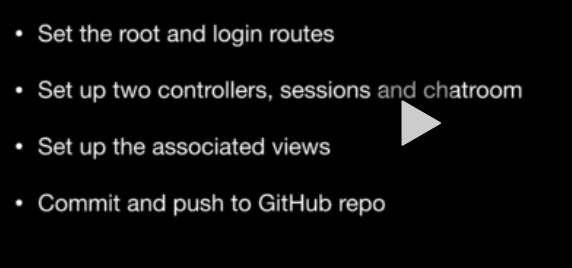
git config --list



git remote -v // Tells where the remote repository for the app is



# Roots and Login Routes



**Create Root or index page**

Step 1: Routes.rb

Rails.application.routes.draw do

  root 'chatroom#index'

end

Step 2: Create Controller with index action – chatroom\_contoller.rb

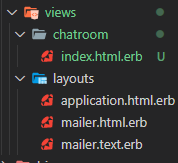
class ChatroomController < ApplicationController

    def index

    end

end

Step 3: Create folder for chatroom controller view and create index file.



**Login route** -> Same procedure

Step 1: Routes.rb

get 'login', to: 'sessions#new'

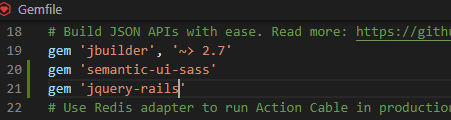
Step 2 and 3 -> same as root route. Create controller, action and view

# Install Semantic UI

1. Install Sematic ui gem. Go to “<https://github.com/doabit/semantic-ui-sass>”
2. Add this to the Gem file:

gem 'semantic-ui-sass' and

gem ‘jquery-rails’ // May be not necessary as it is inbuilt in rails 6. Not sure anyways. Installed again, no problem.



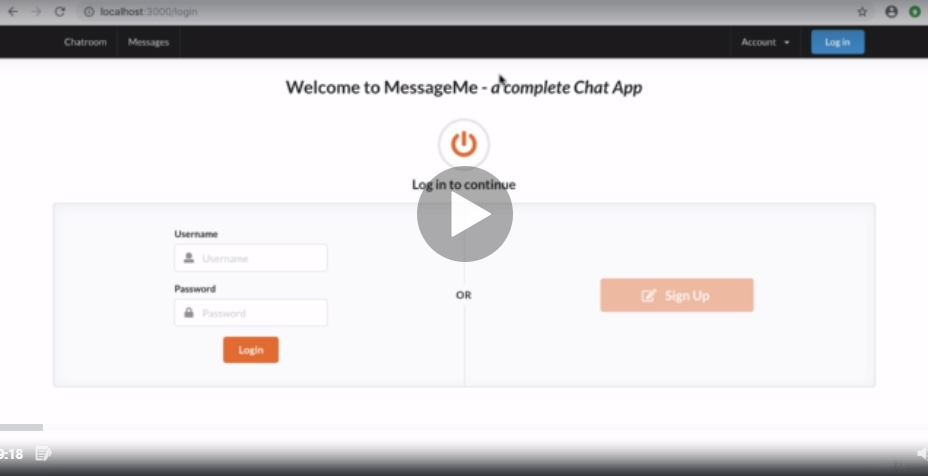
1. Cmd: bundle install
2. For CSS, create a custom.css.scss file in app -> assests -> stylesheets. And paste this code: @import “semantic-ui";
3. Follow the below steps.



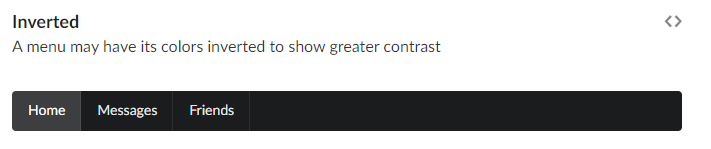
Working successfully. Run the server and check in the browser. The font style should be changed.

# Add Navigation menu

Look of our home page. It is an Inverted navigation menu. Dark in color.

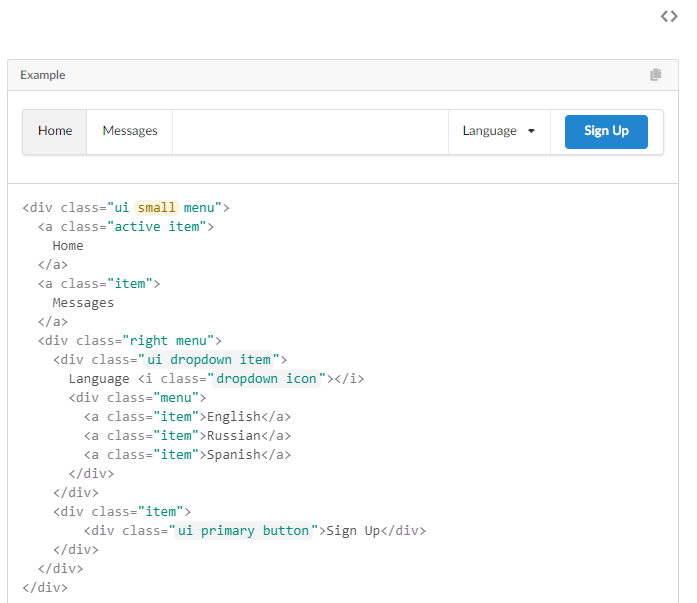


1. Go to Source: <https://semantic-ui.com/collections/menu.html>
2. Variations: It is Inverted with class name as Inverted. Inverted can be used to et the dark color to “Size”





1. Below is variation “Size”. This is the code we need. Use the below code and use class “inverted” to get the darker color.



1. Paste the menu code in the layout -> application.html.erb => so that it is available in all the web pages of the application.

Result in browser:



1. After adding inverted class.

<div class="ui small inverted menu">

Result:



1. Remove active, so that home is not highlighted.

a class="active item">

        Home

</a>

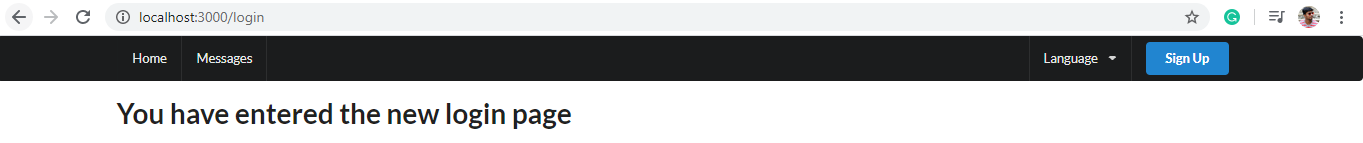
1. To add container in the menu

<div class="ui container">

1. Put the yield inside an container. This will put all the content in the page below the menu inside the container.



The same works in login page also. As we made changes in the application.html.erb



# Create navigation partial and enable dropdown functionality

1. To enable dropdown functionality in navigation bar. We should enable JavaScript functionality. Go to Semantic UI -> dropdown -> Usage and copy the javascript code. Paste the code between the specified function.

$(document).on('turbolinks:load',function() {

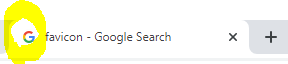
    $('.ui.dropdown').dropdown();

})

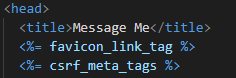
1. Make changes in the namings of the menu bar with respect to our application.
2. Create navigation partial in the layout view. And cut code from the application.html.erb. Pretty much leaving the container and render it.

# Add favicon

(**Favicons** are small square images usually 16×16 pixels which are used by **web browsers** to show a **graphical representation** of the site being visited at the left side of the browser's address bar.)



1. Go to Semantic UI -> Elements -> Icon <https://semantic-ui.com/elements/icon.html>.
2. Search in chrome “Font awesome favicon generator”. Use this <https://gauger.io/fonticon/>. Make necessary changes to the favicon and download it in .ico format
3. Open (Drag and Drop) it under app -> assets -> images
4. Use this emb ruby code <%= favicon\_link\_tag %> under the title tag of application.html.erb

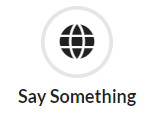


# Build Homepage

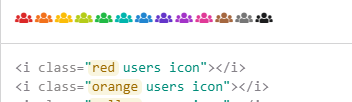
1. Page header with icon. <https://semantic-ui.com/elements/header.html#icon-headers>
2. Copy and paste code in index.html.erb and make necessary changes.

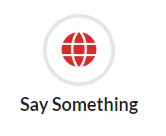
<i class="circular globe icon"></i>

Remember: (globe name here is from the icon name in semantic ui website)



1. To change the color, Icon definition -> (Right side) Variations -> Coloured. Just add red to the existing class name in the index view.

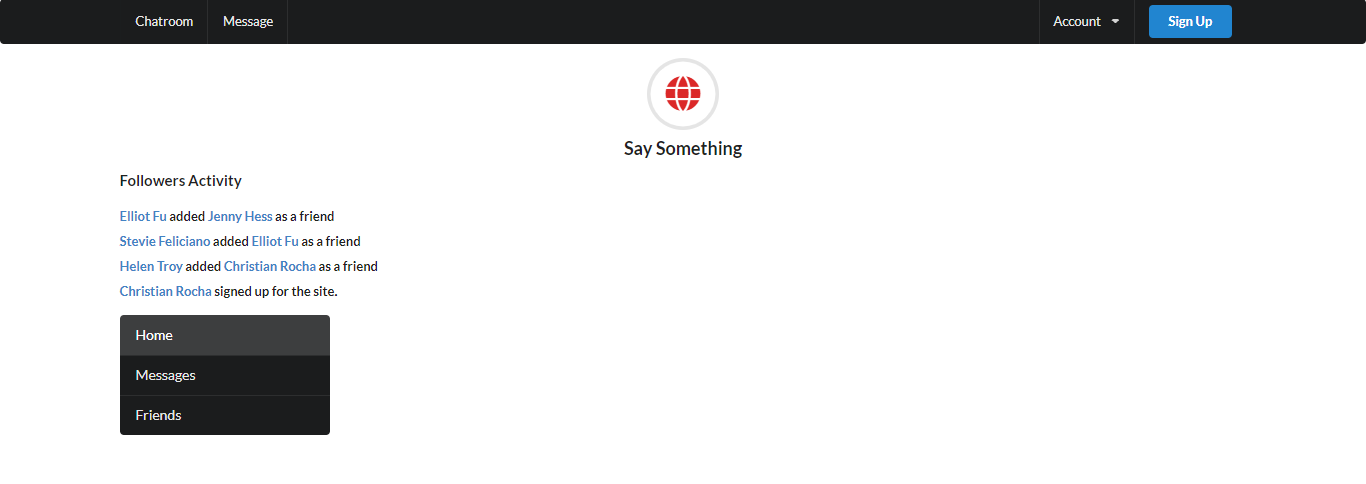




**Grid system** is used to allocate space for the **input textbox and chat area**. By default, it has 16 columns. And the **inverted vertical menu** can be used to display the **member of chat** in the right side.

1. Inverted menu. Go to <https://semantic-ui.com/collections/menu.html#inverted>. Copy the code n paste in index view.
2. Chat. Views -> Feed. Copy code from Variations -> Size.

Check in browser with both the set of codes.



1. Create grid to separate the page. (.ui.two.column.grid)

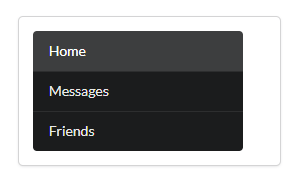
<div class="ui two column grid">

Cover entire code with this. And specify column in next command. (.twelve.wide.column – chatbox)

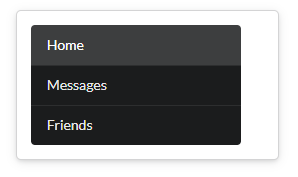
1. After separating the grid, place the chatbox and inverted menu in a card. <https://semantic-ui.com/views/card.html>
2. Use the below code to get a card – Card block (Style)

<div class="ui card">

<div class="content">

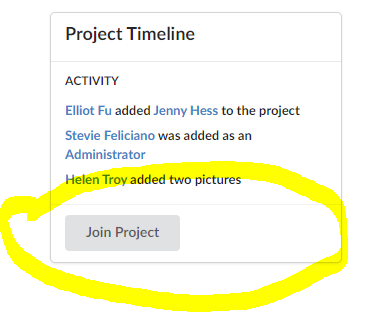
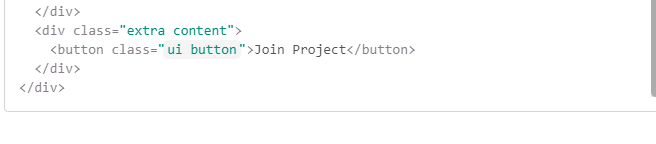


To make it better with shadows, add two more class as, <div class="ui card"> => <div class="ui fluid raised card">



1. Same for the chatbox also.
2. To specify fixed size for chatbox and menu, we specify “chatbox” in both the class and style it in custom stylesheet. <div class="ui fluid raised card chatbox">

**Add Input box and Styling:**

To get this style, we need to add

extra content class.

1. Put the extra content class and the from “Comment” <https://semantic-ui.com/views/comment.html#comments>, copy the form code for text area. And make changes.

<div class="extra content">

              <form class="ui reply form">

                    <div class="field">

                        <div class="ui blue labeled submit icon button">

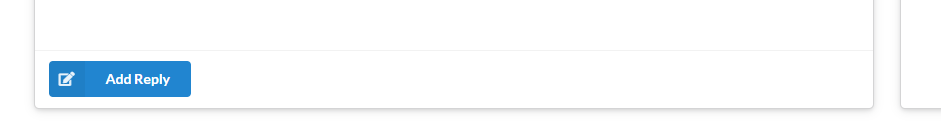
                        <i class="icon edit"></i> Add Reply

                        </div>

                    </div>

           </form>

</div>



1. After making changes:

<div class="extra content">

<form class="ui reply form">

<div class="field">

<div class="ui fluid icon">

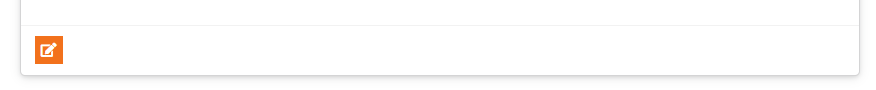
<i class="bordered inverted orange edit icon"></i>

</div>

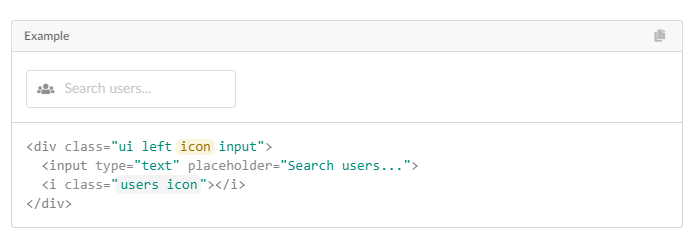
</div>

</form>

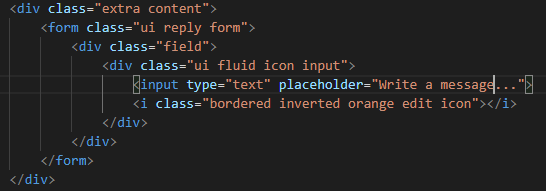
</div>



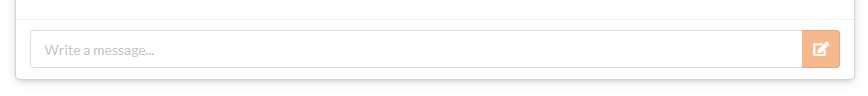
1. To get input box and move the edit icon to right side. Just use classes in our code.



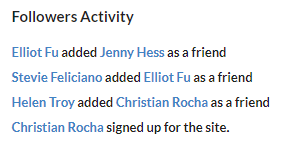
Code:



Result:



1. Clean up the content:

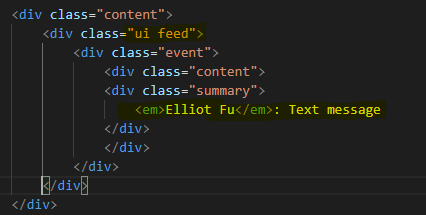


with user name and user message.

<div class="ui small feed">

Remove small, to make the content of the chatbox bigger.

Code:

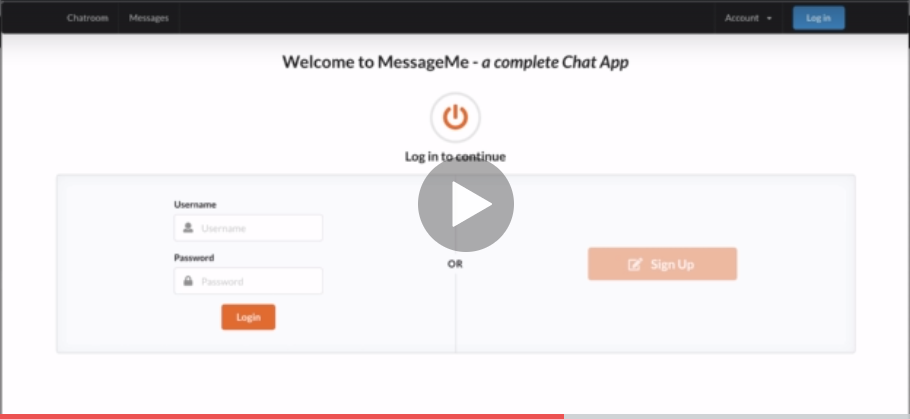


Result:



# Build login page

**Expected View:**



**Task 1:** Link Log in drop down item, to navigate to the Login page. Click on Chatroom, it should navigate to homepage.

In navigation partial, use this code.

<%= link\_to 'Chatroom', root\_path, class: "item" %>

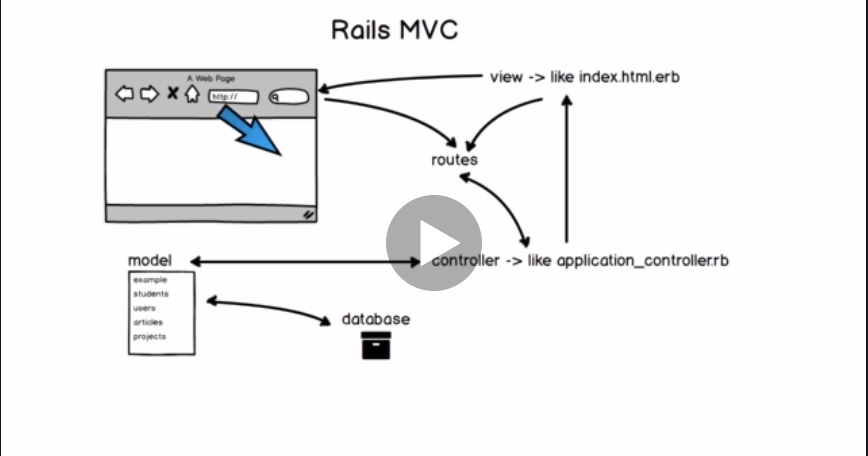
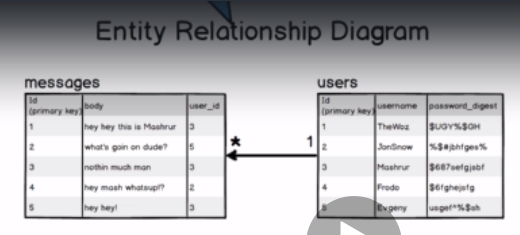
<%= link\_to 'Log in', login\_path, class: "item" %>

**Task 2:** Create the new view as above. Use divider from Semantic UI.

Copy and paste code. Point to be noted – Use form and button class. That’s how button size reduced. Disable class to disable the Sign up button.

# Back End Design

The anticipated flow in our application:

# Build User Resource

## Add multiple users at a time in the table:

**Cmd:** rails generate model user => gives model file and migration file together.

**User.create(username: “Loga”, password: “password”)** => schema of the users table

To create multiple users instead of in console. We can use seeds.rb. Copy the above structure and make changes for the username and password. Save it. And run cmd: **rails db:seed**

Now check in rails console, if all users are added to the table.

## HIRB gem:

To get table like structure in the rails console.

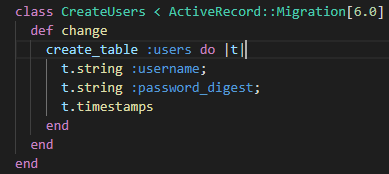
1. Add gem ‘hirb’. Then bundle install in the command prompt
2. Go to rails console, Hirb.enable => true.

Task: Create User table

**Step 1:**

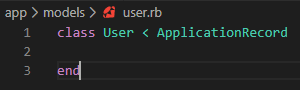
1. Create migration file for users table – To provide the structure of the table.

Cmd: **rails generate migration create\_users**. Migration file will be created.



Enter the attributes (table column names) and type. Then run **rails db:migrate** cmd.

1. Create user.rb file in the model folder.



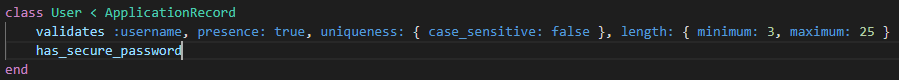
Check in the rails console. User table is created as expected.



### Step 2:

In user.rb,

1. validates check the condition while saving in db
2. has\_secure\_password – check if it has a password.



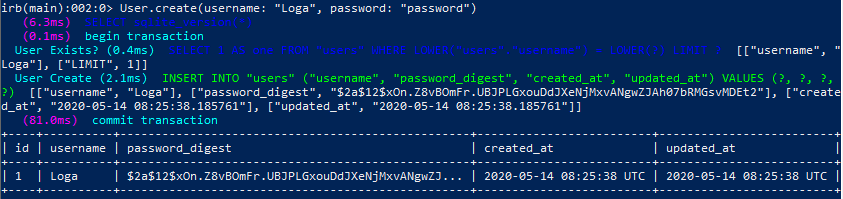
In gem file,

1. uncommand bcrypt gem.
2. Add hirb gem for having table structure in rails console.

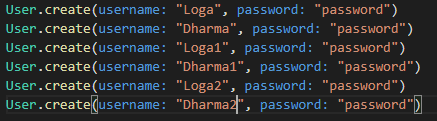


In command prompt, use cmd: “bundle install”. Go to rails console. cmd: “Hirb.enable”

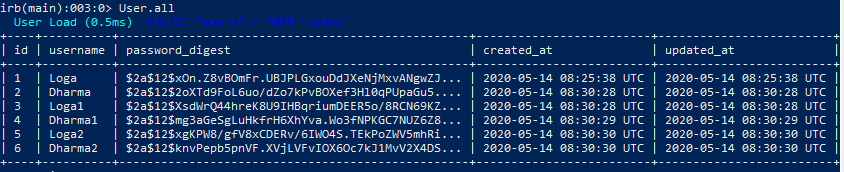
### Step 3: Create Users in the table



To create multiple users. Use seeds.rb



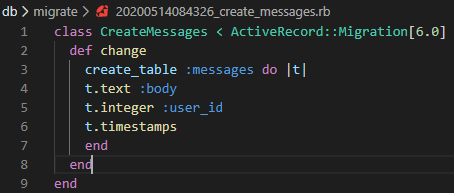
And run cmd: **rails db:seed.** Go to **rails console.**



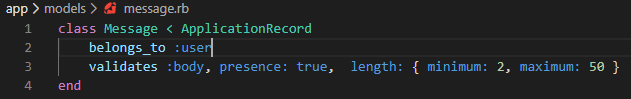
# Build Message Resource

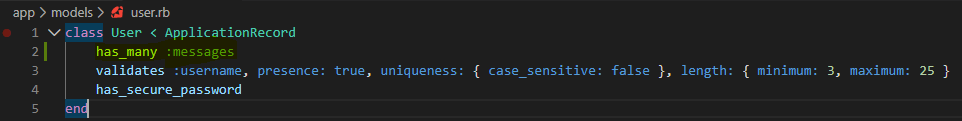
### Step 1:

1. Create migration file and model file. For creating together, use cmd: **rails generate model message**
2. In migration file, user\_id is the foreign key

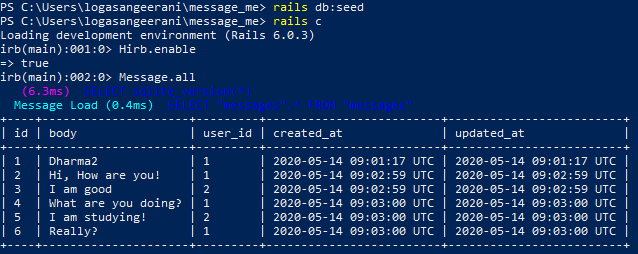


1. Run rails db:migrate
2. In Model, **specify the relationship** means association between two resources.



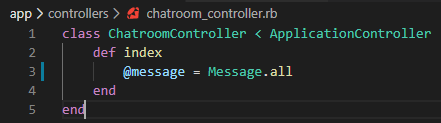


1. Add multiple messages in the seed file. Run console.

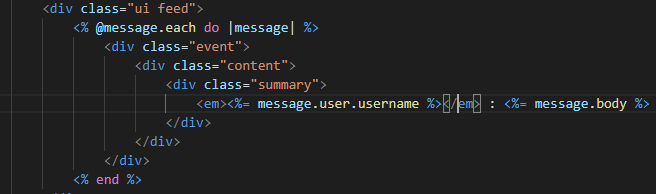


# Add/Show actual messages in the browser

### Chatroom Controller



### index view



### Output

