UNIVERSITY OF ECONOMICS AND LAW

FACULTY OF INFORMATION SYSTEMS

GRADUATION INTERNSHIP REPORT

MAJOR IN MANAGEMENT INFORMATION SYSTEM

BUILDING A WEBSITE PROVIDING RECRUITMENT SERVICES USING RUBY ON RAILS FRAMEWORK

Company/Organization:

ZIGExN VeNtura Co., Ltd

Supervisor: **MSc. Nguyen Quang Phuc**

Student:

Student ID: K194060852

Full name: Phan Quang Minh Long

Class: K19406C

Ho Chi Minh City, 12/2022

ACKNOWLEDGEMENTS

After more than two months of internship at ZIGExN VeNtura Co., Ltd, I have learned a lot of knowledge and skills to solve problems in work and life. This will be a very meaningful time for me to experience the position of a Web Development intern at the company. This job has helped me to form a sense of responsibility and initiative in solving all problems raised. In addition, it also motivates me to keep going in my career and pursue my goals.

First of all, I would like to thank the teachers in the Faculty of Systems for their dedication to imparting useful knowledge to me so that I can apply it today. I would like to thank Mr. Nguyen Quang Phuc for being my instructor so that I could complete this report

Thank you to Mr. Binh (my mentor) and Mr. Hieu Le (COO), who guided me and gave me the opportunity to participate in the company's internship projects so that I could complete the report in the best way. Thank you for your dedicated training during the internship period.

Because of my limited knowledge, in the process of implementing and perfecting this topic, I inevitably make mistakes, and I hope to receive suggestions from my instructor, teachers as well as the company.

Finally, I would like to wish company, colleagues and lecturers good health, success and happiness in career and life.

Thanks and best regards,

Phan Quang Minh Long

COMMITMENT

I hereby declare that the content of this project is self-executed. In addition to the references cited, the information used in this study is my own research and implementation. Information is collected and handled honestly.

INTERNSHIP REPORT RESULT ASSESSMENT FORM

(For instructor)

ID student: ……

Full name: ……..

Instructors: …….

No.	Criteria	Specific criteria	Point	Note
1	**Report Format (20%)**	Presentation (5%)		
		Report structure (10%)		
		Writing style (5%)		
2	**Report content (45%)**	Analysis skill (5%)		
		Objective (10%)		
		Specialized knowledge results (30%)		
3	**Student's attitude (15%)**			
4	**Evaluation of enterprise (20%)**			
TOTAL				

Ho Chi Minh City, December 12th 2022

Instructors

Sign, write full name

COMMENTARY FORM OF INSTRUCTORS

……

……

……

……

……

……

……

……

……

……

……

……

……

Instructors

Sign, write full name

WORKING SCHEDULE AND ASSIGNED TASKS

Time: 03/10/2022 – 31/12/2022

ID	Time	Task	Workplace
1	03/10 – 05/10	- Setup ENVIRONMENT on local computer (Git/Github, Ruby, Code Editor, MySQL …). - Learn basic commands on Ubuntu operating system. - Review knowledge on how to use Git. - Meeting with the mentor to discuss the learning process.	At the company office
2	06/10 – 07/10	- Review HTML/CSS and practice related exercises. - Have meeting with the mentor to discuss the learning process.	At the company office
3	10/10 – 11/11	- Review SQL knowledge and practice on MySQL. - Have meeting with the mentor to discuss learning and revision.	At the company office
4	12/10 – 14/10	- Learn the basics of Ruby language and practice advanced exercises. - Have meeting with the mentor to consolidate knowledge.	At the company office
5	17/10 – 31/10	- Learn the installation and basic concepts of Rails. - Learn Database Migrations. - Learn CRUD with ActiveRecord, Associating Models. - Practice ActiveRecord, Associating Models. - Learn routes and action index or show. - Learn routes and action new and create. - Learn routes and action edit and update. - Learn routes and action destroy and practice. - Learn apply kaminari and device gem to rails.	At the company office

		- Learn apply CSS to Rails web app. - Review and practice all about Rails. - Have meeting with the mentor to consolidate knowledge.	
6	01/11 – 31/12	- Practice Ruby on Rails with Venjob (A WEBSITE PROVIDING RECRUITMENT SERVICES). - Have meeting with the mentor every week for checking process, pull requests and reviewing codes on personal Github repository. - Presentation on the completion of the Venjob project. - Organize a "Thik Talk" seminar on technology topics related to trained knowledge (Solr) with the intern team.	At the company office

CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

RoR	Ruby on Rails
PR	Pull Request(s)
IT	Information Technology
COO	Chief Operations Officer
CV	Curriculum Vitae
MVC	Model – View – Controller
RVM	Ruby Version Manager
SQL	Structured Query Language
RDBMS	Relational Database Management System
HTML	Hypertext Markup Language
ERB	Embedded Ruby
CSS	Cascading Style Sheets
SASS	Syntactically Awesome StyleSheets
SCSS	Sassy Cascading Style Sheets
JS	Javascript

CHAPTER 1: TOPIC OVERVIEW

1.1. Reasons

Job search keyword is a very popular keyword, and surely all of us must search for it. There are many famous and reputable websites that help us find jobs that meet our needs such as: Vietnamwork, Careerlink, Timviecnhanh, TopCV, 123job... most of these websites have same functions such as job search, company search, online application, administrators can go to the website to find candidates through the CVs they have submitted to the system.

I am very interested in a system like this, and now I am also working in the position of Web Development Intern. And I challenged myself before entering the company's projects. I have decided to create a similar system with the above functions and add some functions that I find useful and necessary.

The website system I am building is like a link between employers and job seekers. The system provides functions such as job search through keywords of job name, industry and location to help users search for jobs according to their needs. Users can search for a company and from there can see all the open jobs of that company. In addition, the website also has additional functions, managing users' CVs and applying those CVs to jobs that users apply for.

With the rapid development of information technology in general and website technology in particular, website development is increasingly supported by many advanced tools, framworks and libraries, making web development less tiring and more effective.

Those are the reasons that motivated me to do “Building a website providing recruitment services using Ruby on Rails framework” topic. After more than 3 years of studying at the school, because of the desire to have more practical experience, as well as to participate in learning in a professional environment. Therefore, I decided to choose ZIGExN VeNtura, which has an ideal, modern, professional environment, as the place that will help me realize this plan.

1.2. Objectives

The topic explores the theoretical basis of web development in general, building a website system with a framework in particular, as well as how to deploy and set up a development environment for the website under the support of the framework to help develop the website less tiring, more efficient, and makes web applications perform faster, more sustainably, and more accurately than other websites. The main goal of the project is to build a website providing job search support services.

One of the important goals of this report is to improve the job search website with better quality, better performance than other websites with the same content. This will increase the overall quality of the website. Excellent quality contributes to a significant increase in customer satisfaction as well as reduced maintenance costs.

1.3. Objects and scopes

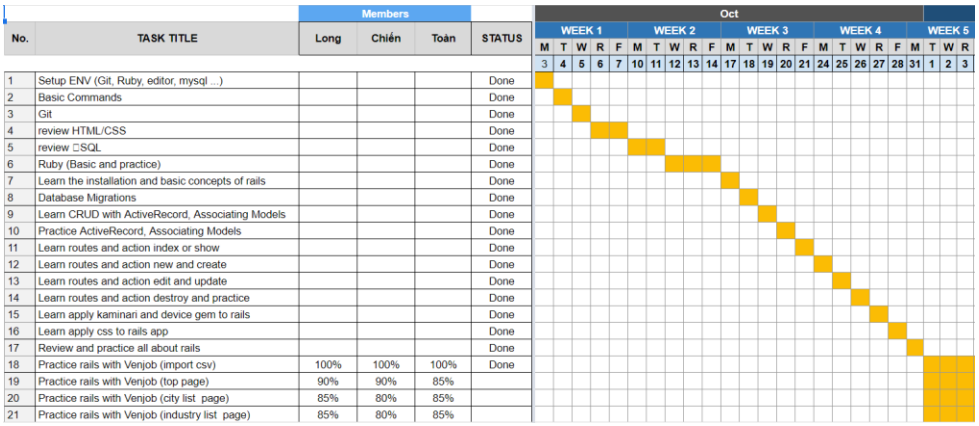
Objects: This topic focuses on analyzing the requirements to build functions on the website. Also apply the Ruby on Rails framework to build websites quickly.

Scope: Due to the limitation of the report, the report will focus on analyzing the functions to support job candidates on Venjob website using Ruby on Rails framework and a number of support libraries.

1.4. Implementation plan

Period	**Work**
Period 1	Analyze and define requirements.
Period 2	Work on data import process from FTP server.
Period 3	Use RoR to design the interface and do common functions of the website for candidates.
Period 4	Use important platforms and libraries in the project: Solr for job search optimization and Devise for user information processing.
Period 5	Write report.

Figure 1: Implementation plan



No.	TASK TITLE	Members			STATUS
		Long	Chiến	Toàn	
1	Setup ENV (Git, Ruby, editor, mysql ...)				Done
2	Basic Commands				Done
3	Git				Done
4	review HTML/CSS				Done
5	review □SQL				Done
6	Ruby (Basic and practice)				Done
7	Learn the installation and basic concepts of rails				Done
8	Database Migrations				Done
9	Learn CRUD with ActiveRecord, Associating Models				Done
10	Practice ActiveRecord, Associating Models				Done
11	Learn routes and action index or show				Done
12	Learn routes and action new and create				Done
13	Learn routes and action edit and update				Done
14	Learn routes and action destroy and practice				Done
15	Learn apply kaminari and device gem to rails				Done
16	Learn apply css to rails app				Done
17	Review and practice all about rails				Done
18	Practice rails with Venjob (import csv)	100%	100%	100%	Done
19	Practice rails with Venjob (top page)	90%	90%	85%	
20	Practice rails with Venjob (city list page)	85%	80%	85%	
21	Practice rails with Venjob (industry list page)	85%	80%	85%	

Figure 2: Image of part of the intern's work plans

1.5. Structure

The report consists of 7 chapters as follows:

CHAPTER 1: TOPIC OVERVIEW

In this chapter, the report presents the following content: Overview of the Reasons, Objectives, objects and research scope of the topic; implementation plan and project structure.

CHAPTER 2: INTRODUCTION TO THE COMPANY AND PROJECT

Introduction about ZIGExN VeNtura Co., Ltd and Information of Venjob project.

CHAPTER 3: THEORETICAL BACKGROUND

Introduction to Web Development, Bootstrap, Ubuntu, MySQL, Ruby on Rails framework.

CHAPTER 4: WEBSITE SYSTEM REQUIREMENTS

Overview of the web system, operations of the objects in the site sytem, system requirements and sitemap.

CHAPTER 5: SITE'S DATABASE DIAGRAM

Modeling data and relationships between tables and "design" tables.

CHAPTER 6: BUILDING VENJOB WEBSITE USING ROR

Implement web project using Ruby on Rails framework.

CHAPTER 7: CONCLUSION

Outlining a number of points achieved and limitations of the topic when participating in the project.

CHAPTER 2: INTRODUCTION TO THE COMPANY AND PROJECT

2.1. Introduction to the company

ZIGExN VeNtura is a 100% invested company from Japan. Known as a symbol of a young and dynamic community in the IT field today, ZIGExN VeNtura operates in the field of software development and Internet services for the Japanese market. Being present in Vietnam market since the beginning of 2013, with the right steps from the Board of Directors and enthusiastic, creative and enthusiastic staff, ZIGExN VeNtura is growing and developing constantly.

Figure 3: Logo of ZIGExN VeNtura Co., Ltd

- Mission: With sologan "Be a creator", ZIGExN VeNtura believing that everyone has infinite potential to achieve the great thing.
- Tax code: 0312124787.
- Vision: Bring the Innovation from VIETNAM to the WORLD.
- Industry: IT Services and IT Consulting.
- Company size: 51 - 200 employees (Includes members with current employer listed as ZIGExN VeNtura Co., Ltd, including part-time roles).
- Specialties: Web development, Ruby, Ruby on Rails, Market Research, Human Resources Consulting, Software Development, and Online Ticketing Services…

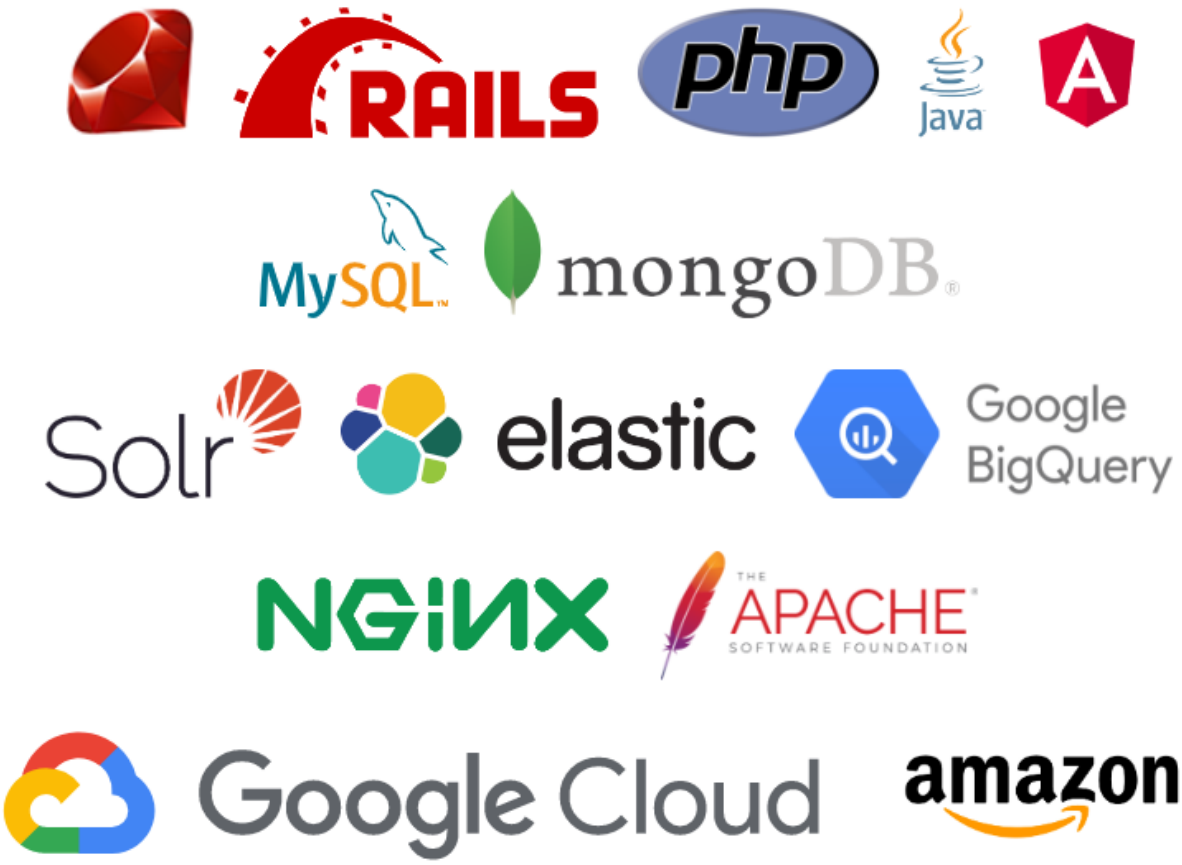


Figure 4: ZIGExN Technologies

2.2. Introduction to the project

- Job position: Web Development Intern.
- Responsibility and authority: As assigned by the mentor.
- Assigned project: Venjob is a fully functional job search service in Vietnam and international. Users / candidates can search for jobs by job name, workplace, industry, description... to apply for available positions. Site can manage the website's information and data. Jobs data will be imported from daily updated CSV file from secret FTP server.
- Tasks assigned and performed:
  - Participate in review and analyze the project requirement to understand objectives.

- Meeting with the mentor every week to discuss as well as correct the work being done in the project.
- Design website interface with HTML, SCSS, Javascript and use Boostrap library to speed up progress.
- Build database system through Ruby on Rails framework and save them on MySQL, use some libraries (gems) to improve the website more.
- Use the Solr platform to perform multi-field job search.
- Prepare the reports related to web development carried out.
- Ensure that all related work is carried out as per the defined standards and procedures.

CHAPTER 3: THEORETICAL BACKGROUND

3.1. Website Development

Website development, also known as web programming: is the work of building websites, creating web applications that run on any web browser. Web development includes front-end, back-end programming to help the website operate smoothly and stably, providing the best user experience.

Web Developer's work is often very flexible. Depending on the expertise of each person, they can develop in different directions such as:

- Front-end Developer: FrontEnd Developer's job is to use tools to build things that users, customers or visitors to the website can see. Or simply build what is present on the website.
- Back-end Developer: BackEnd Developer's job is to build core values within. BackEnd Developer builds the code and language that runs on the server. These values users as well as customers can only "feel" it through the effective operation of the website.
- Full-Stack Developer: Full Stack Developers are considered to be comprehensive in all aspects, multi-functional. They can do a mix of back-end and front-end, providing a full package of web development services.

3.2. Bootstrap

Figure 5: Bootstrap logo

Bootstrap is a free collection of open source and tools for creating a complete website template. With predefined interface properties such as size, color, height, width, etc., designers can create many new products but still save time when working with this framework in the process of web interface design.

Bootstrap allows the website design process to be faster and easier based on the basic elements available such as typography, forms, buttons, tables, grids, navigation, image carousels...

3.3. Ubuntu

Ubuntu is an operating system developed from the Linux kernel, primarily for desktops based on Debian GNU/Linux but with a stronger focus on usability, frequent releases and ease of installation as well as ease of installation. platform diversity. Ubuntu is designed for computers, smartphones and network servers. All principles used to develop software on Ubuntu are based on open source software development principles.

Figure 6: Ubuntu logo

3.4. MySQL

MySQL is an RDBMS that operates under the client-server model. MySQL manages data through databases. Each database can have many relational tables containing data. MySQL also has the same access and code as the SQL language.

Figure 7: MySQL logo

3.5. Ruby on Rails framework

3.5.1. Overview

Ruby on Rails (RoR) is a web framework written in Ruby and all applications in Rails will be written in Ruby. Ruby on Rails was created to help developers develop web-based software as quickly as possible.

The Rails framework takes advantage of features of the Ruby language. Ruby is a scripting, dynamic type definition, and object-oriented language, it's designed with a clean syntax, making it as easy to read and as concise as possible for the user. To make software development faster, RoR uses radical conventions and takes care of a lot of tasks so that programmers don't have to worry about it anymore, such as: mail manager, object- database mappers, file structures, code generation… these are the 2 most outstanding features of RoR, which not only help programmers write less code, develop applications faster, but also make applications easier understand and easier to maintain.

Figure 8: Ruby and Ruby on Rails logos

3.5.2. MVC pattern

RoR uses Model – View – Controller (MVC) architectural patterns to enhance the maintainability and development of the application. MVC allows us to clearly divide the application into logical, business and user-interface layers, which also makes it easier to test and reuse the code.

3.5.2.1. Model

The Model layer handles the application's business and directly manipulates the data. In RoR, the model layer is often used to interact with their respective components in the database and validate data.

3.5.2.2. View

The view layer displays the user interface, in RoR views are HTML files embedded with Ruby code. The Ruby code embedded in the HTML file is quite simple, usually consisting of only loops and branching conditional statements, which are used to display data on the view form.

3.5.2.3. Controller

The controller interacts with the model and the view. Requests coming from the browser will be handled by the controller, then the controller can interact with the model to get data and then return it to the view to display the information.

3.5.3. RoR architecture

The architecture of RoR has the following characteristics:

- MVC Architecture.
- Representational Sate Transfer (REST) for web services.
- Supports many large database management systems such as MySQL, Oracle, MS SQL, PostgreSQL...
- The Ruby scripting language is written on the Server side.
- Use conventions instead of configuration.
- There are script generators to automate tasks.
- Corresponding to the above characteristics RoR includes the following components:
    - Action Mailer.
    - Action Pack.
    - Action Controller.
    - Action Dispatcher.
    - Action View.

- Active Model.
- Active Record.
- Active Resource.
- Active Support.
- Railties.

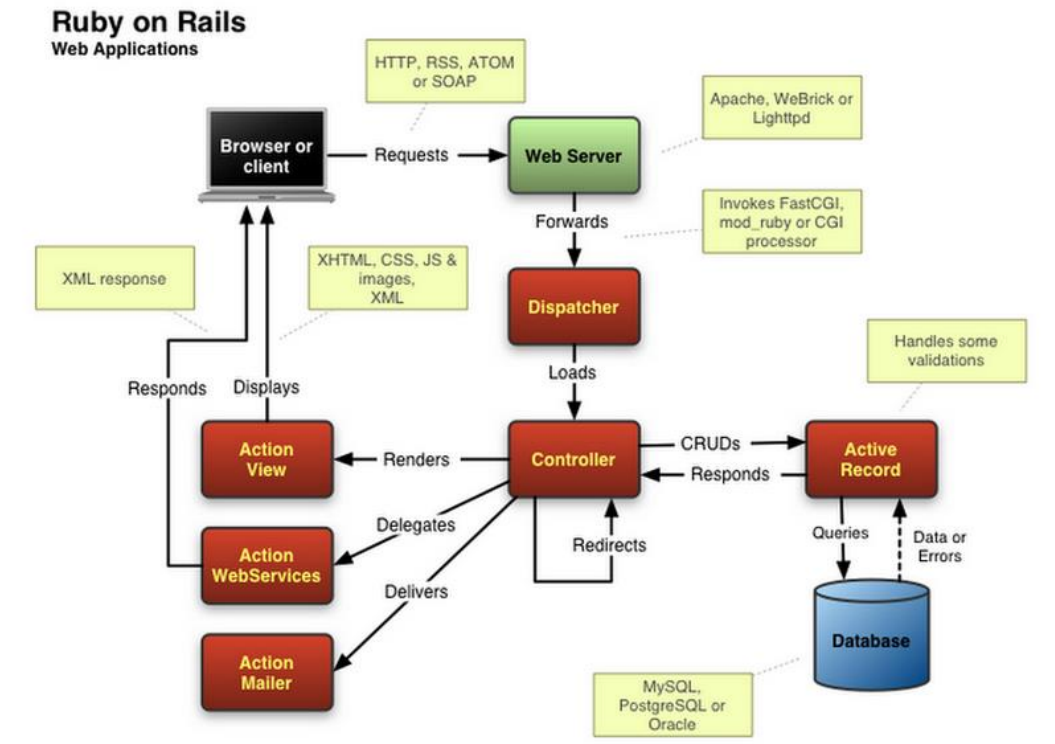


Figure 9: Overall architecture of RoR

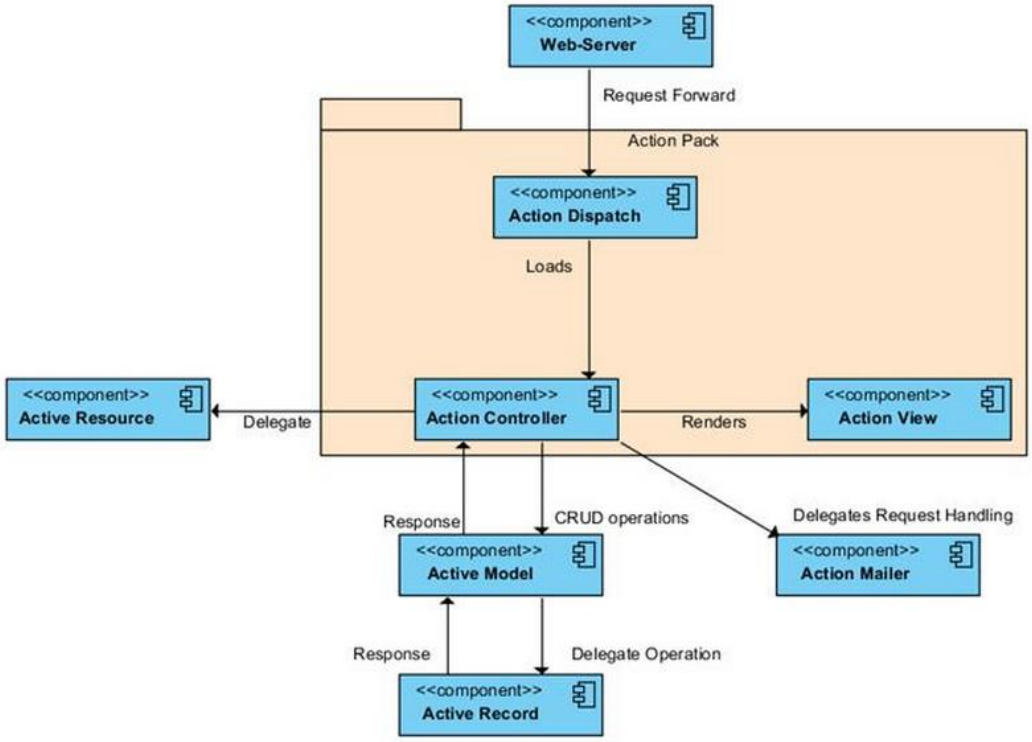


Figure 10: Dynamic View Pakage Diagram

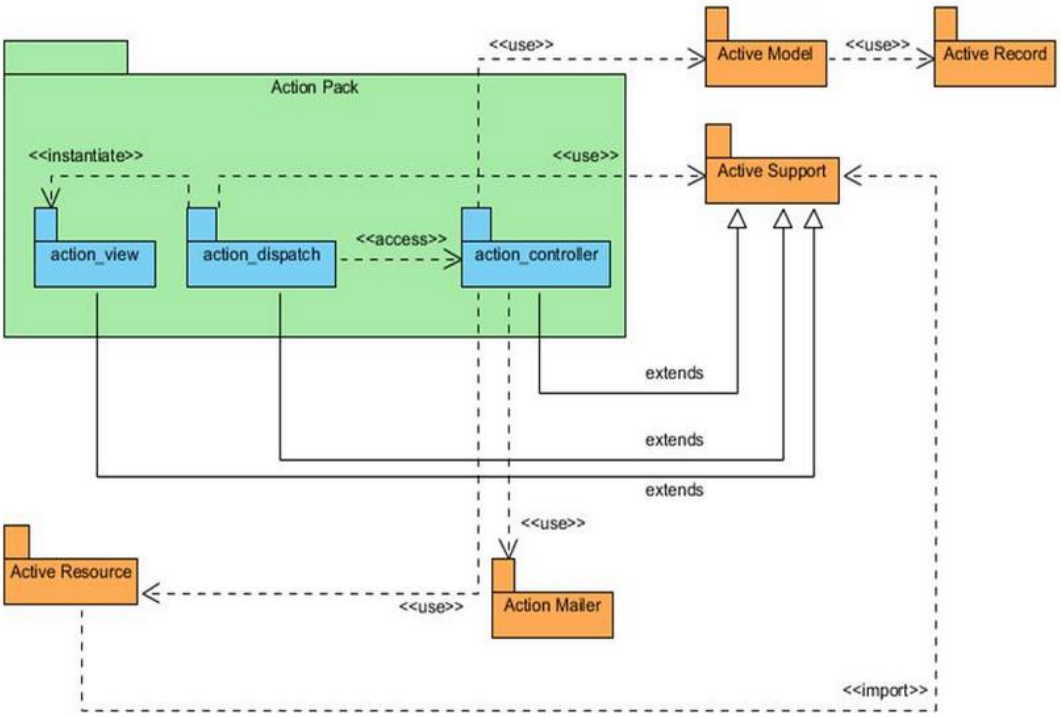


Figure 11: Static View Page Diagram

3.5.4. Conclusion

Rails is built with Ruby, so it inherits the strengths and weaknesses of this language. Since it is an interpreted language, it runs slower than other compiled languages like Java, C++ or C…

One more thing, Rails supports developers too much and forces our application to follow the MVC patterns and conventions. It will be difficult if we continue to develop an application that has been built in another language, with pre-existing databases, with a different model in RoR.

Although there are some weaknesses, but we also have to admit the strengths that RoR has while other frameworks do not, RoR architecture has accepted to sacrifice flexibility in declaring configuration files by follows conventions to take advantage of framework support, so RoR is suitable for developing from scratch small and medium-sized applications that require short development time and are easy to maintain, read and understand, not requires high performance.

3.6. Solr

Solr is a professional search platform, open source and built in Java on Apache Lucene. It is very popular and has a fast processing speed. The Sunspot library is built to work with Rails.

By default, Sunspot uses a rather conservative configuration for fulltext search. Text will be split into tags based on spaces and other characters (using a clever tokenizer called **StandardTokenizer**).

Solr is extremely flexible in terms of indexing and full-text search, and a lot of advanced functionality can be configured quite easily.

Figure 12: Solr logo

CHAPTER 4: WEBSITE SYSTEM REQUIREMENTS

4.1. Overview of the web system

- The system is a fully functional job-searching service. It's a website that provides functions such as job search through keywords of job name, position, and industry to help users find jobs according to their needs.
- In addition, the website also has a CV upload function, helping users to manage their CVs and apply those CVs to the jobs they apply for.
- Administrators can search for candidates through the jobs they have applied for by email, location, industry and time period.
- Job data is always updated every day.

4.2. Operations of the objects in the site system

4.2.1. Candidates Activity

- Candidates can go to the website and search for a job that is suitable for them or log in to the website's system to be able to apply online.
- Candidates who register for an account of the website will be able to update information such as: full name, email, password, CV... contact information so that companies can contact candidates.
- Candidates can view the detailed description, benefits and perks of the job.
- Candidates can save the jobs they want.

4.2.2. Administrators activities

- Database administration and general information of the website system.
- Log in to the site as an administrator.
- Search for jobs that candidates have applied for or CVs by email, city, industry and time period. Admins can download this information into a CSV file on their computer.
- Create, read, update, delete (CRUD) data such as cities, industries, job names, job types,

job levels as well as admin and candidate accounts.

4.3. System requirements

To overcome the disadvantages of recent old job-search web systems on the Internet, the new web system should have the following functions and requirements:

- Candidates can submit their resumes to recruiters.
- Candidates can archive jobs that they feel are suitable.
- Administrators can view candidate's applied jobs, from there can contact them.
- Jobs need to be import from a CSV file which is updated everyday on a FTP server. All the import code need to be written so that it can be execute everyday (by crontab).
- Use Solr for full-text search.
- Use latest MySQL version for database.
- Use Ruby on Rails (Ruby version: 2.7.4, Rails latest version: 7.0.4).
- Use latest Bootstrap version (v5) for HTML layout.
- Paginagion: Use Kaminari gem.
- File Upload: Active Storage or Carrierwave gem.

4.4. Sitemap

This is the sitemap of VENJOB released by the right holders.

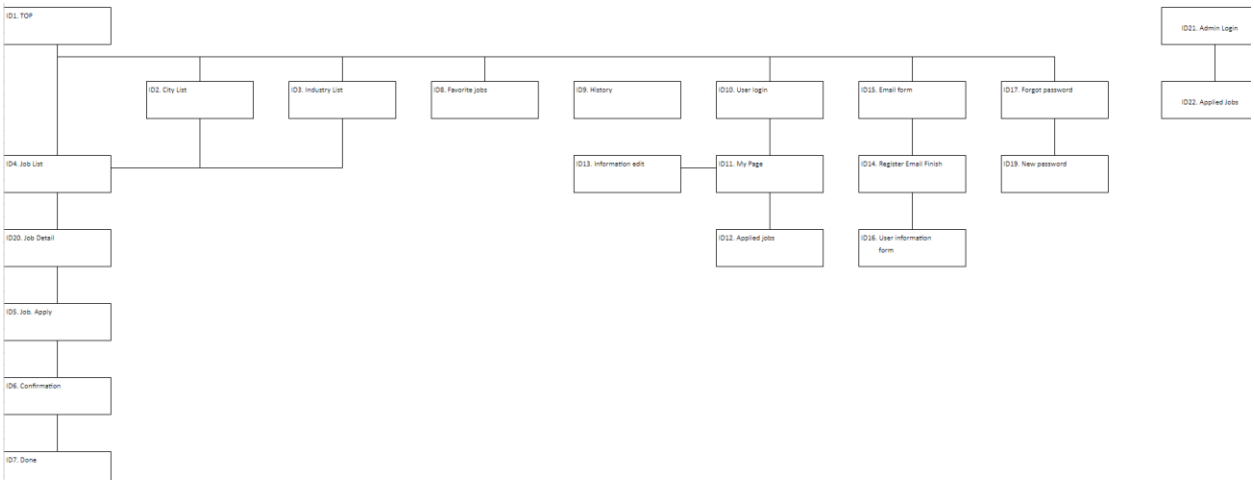
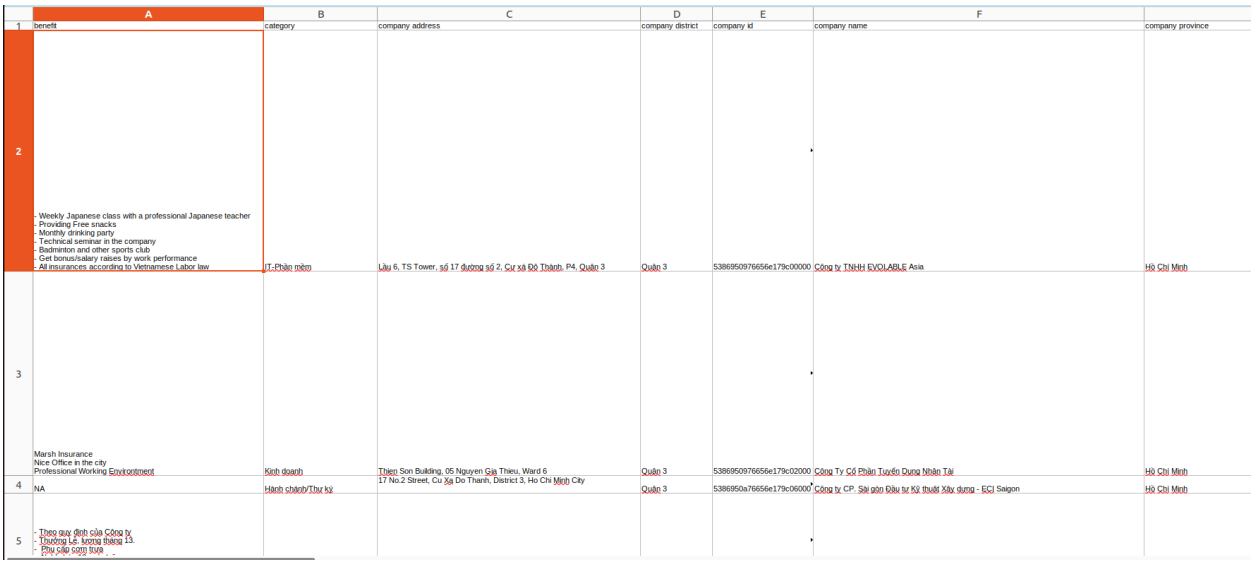


Figure 13: Sitemap of VENJOB

CHAPTER 5: SITE'S DATABASE DIAGRAM

Because jobs data will be imported from daily updated CSV file from secret FTP server, so understanding the data in the CSV file is the top priority that needs to be done first. From there we can design the right database tables and use them for the website as the steps below.



	A	B	C	D	E	F	
1	benefit	category	company address	company district	company id	company name	company province
2	- Weekly Japanese class with a professional Japanese teacher - Providing Free snacks - Monthly drinking party - Technical seminar in the company - Badminton and other sports club - Get bonus/salary raises by work performance - All insurances according to Vietnamese Labor law	IT-Phần mềm	Lầu 6, TS Tower, số 17 đường số 2, Cư xá Đô Thành, P4, Quận 3	Quận 3	5386950976656e179c00000	Công ty TNHH EVOLABLE Asia	Hồ Chí Minh
3	Marsh Insurance Nice Office in the city Professional Working Environment	Kinh doanh	Thien Son Building, 05 Nguyen Gia Thieu, Ward 6	Quận 3	5386950976656e179c02000	Công Ty Cổ Phần Tuyển Dụng Nhân Tài	Hồ Chí Minh
4	NA	Hành chánh/Thư ký	17 No.2 Street, Cu Xa Do Thanh, District 3, Ho Chi Minh City	Quận 3	5386950a76656e179c06000	Công ty CP. Sài gòn Đầu tư Kỹ thuật Xây dựng - ECI Saigon	Hồ Chí Minh
5	- Theo quy định của Công ty - Thưởng Lễ, lương tháng 13. - Phụ cấp cơm trưa						

Figure 14: Sample data from CSV file (view on Ubuntu)

1.1. Modeling data and relationships between tables

After learning about the CSV file downloaded from FTP Server, I made a sketch of a diagram from which I can program relationships and constraints for data objects.

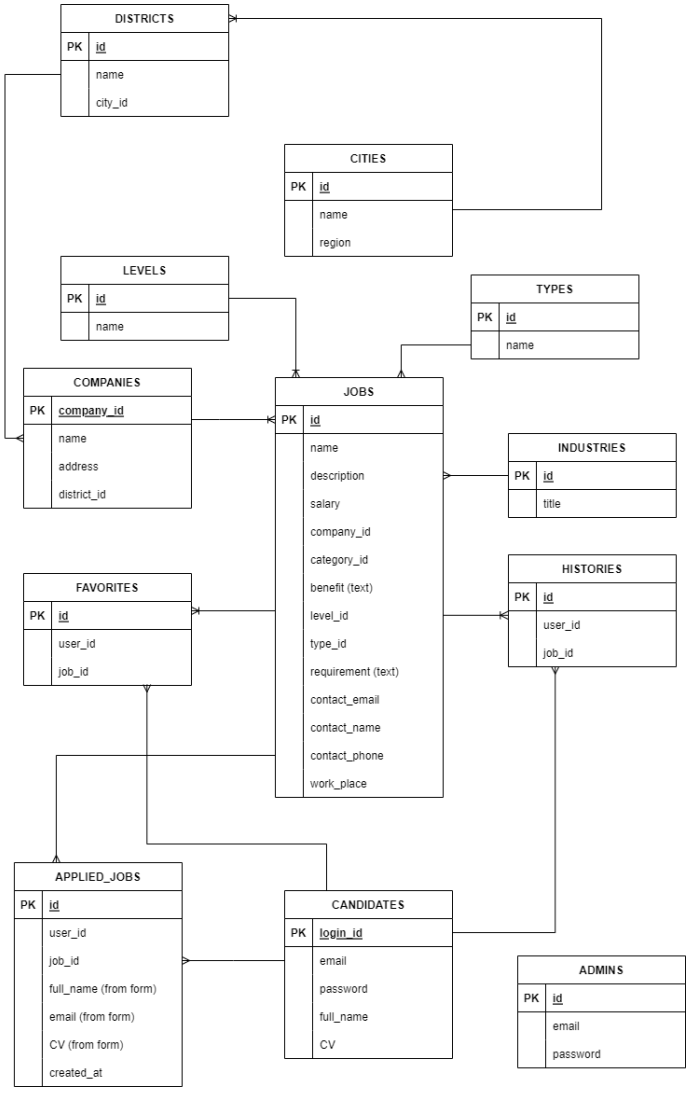


Figure 15: Data diagram of website application system: Demo (demoed by me)

1.2. Design tables

1.2.1. Jobs Table

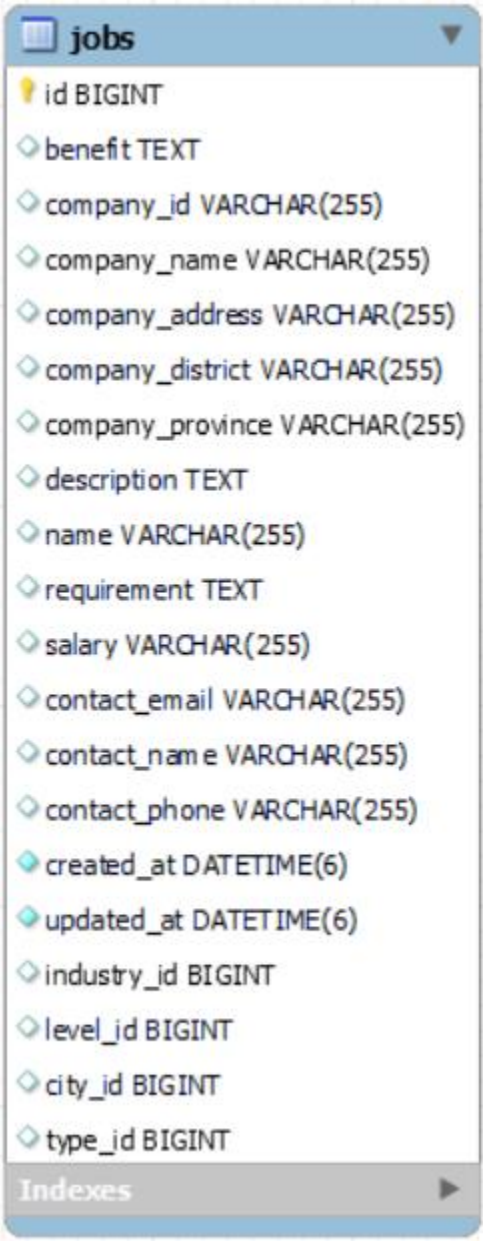


Figure 16: Jobs table

1.2.2. Cities Table

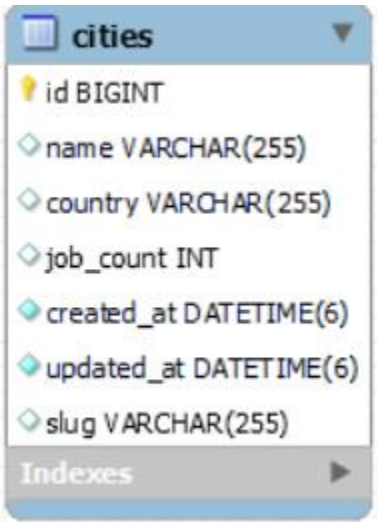


Figure 17: Cities table

1.2.3. Industries Table

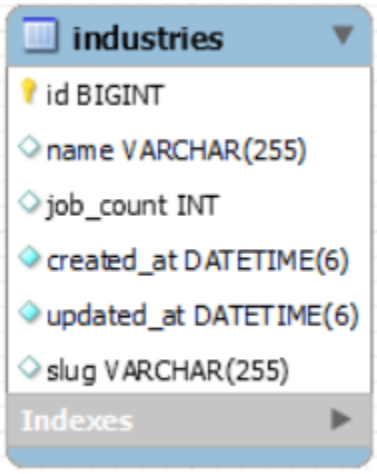


Figure 18: Industries table

1.2.4. Types Table

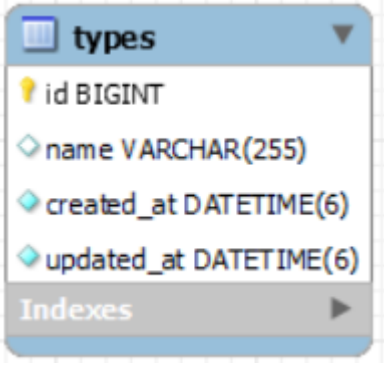


Figure 19: Types table

1.2.5. Levels Table

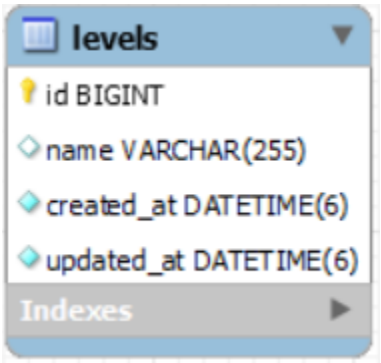


Figure 20: Levels table

1.2.6. Users Table

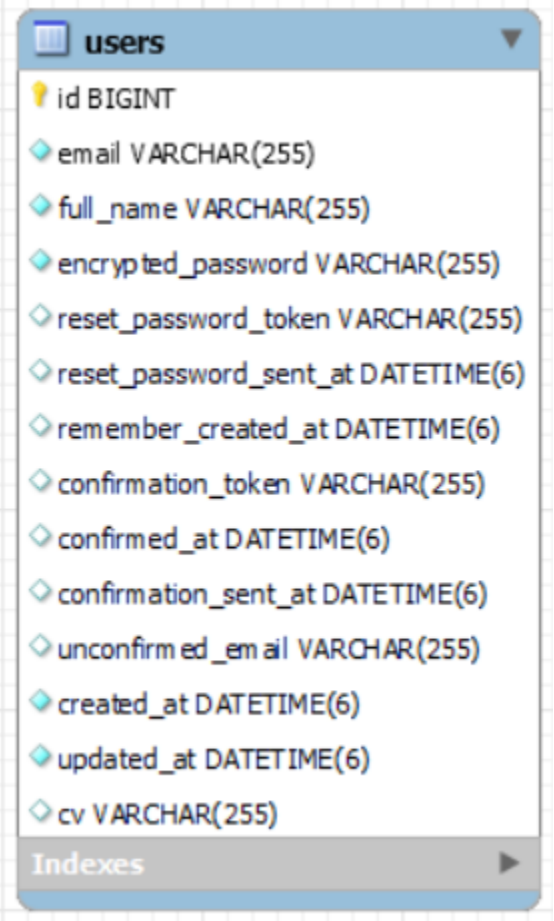


Figure 21: Users table

1.2.7. Admins Table

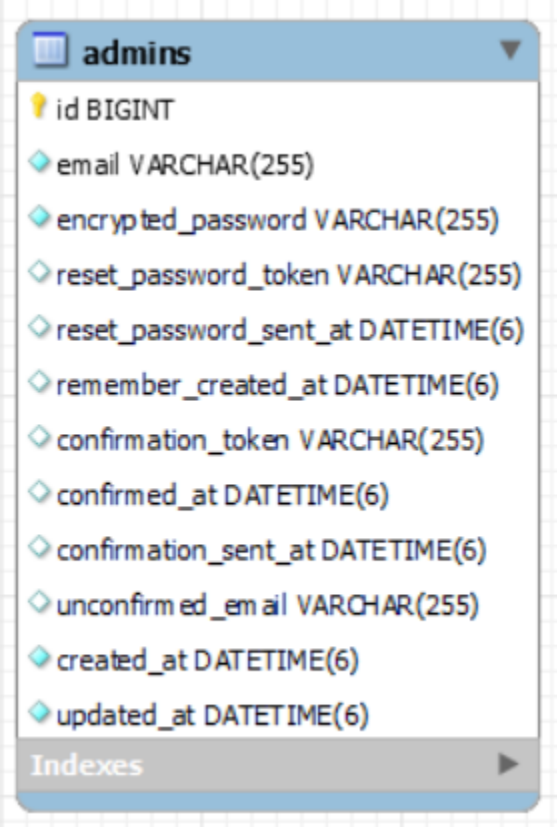


Figure 22: Admins table

1.2.8. Favorites Table

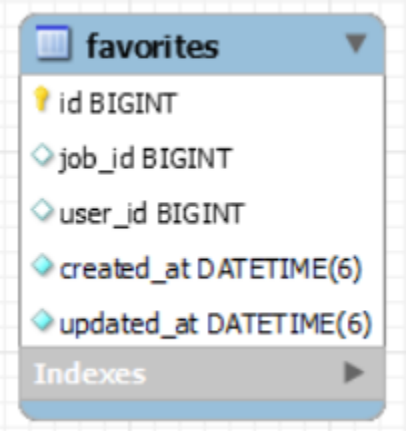


Figure 23: Favorites table

1.2.9. Histories Table

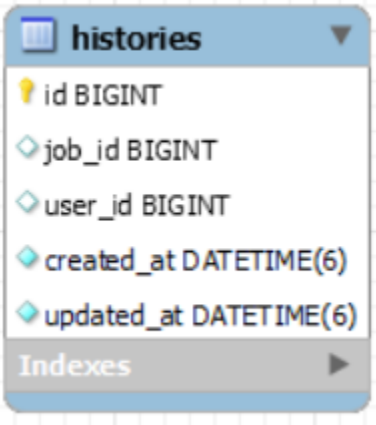


Figure 24: Histories table

1.2.10. Applies Table

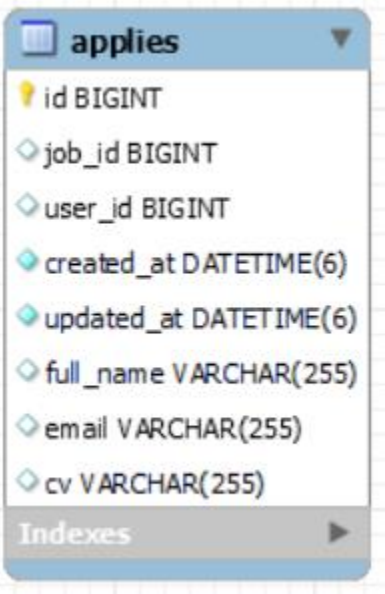


Figure 25: Applies table

CHAPTER 6: BUILDING VENJOB WEBSITE USING ROR

6.1. Initializing a Ruby on Rails project

- First, I open VS Code in the folder where I will initialize the project. Then create .ruby-version file with the following line:

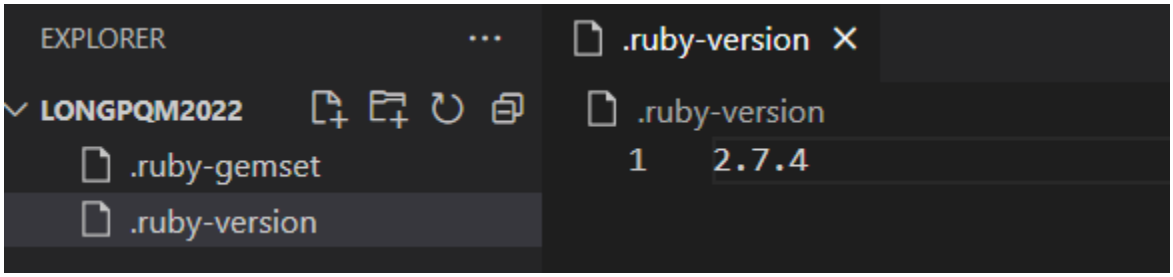


Figure 26: Add a line to .ruby-version file

- And create .ruby-gemset file with the name of the project in the following line:

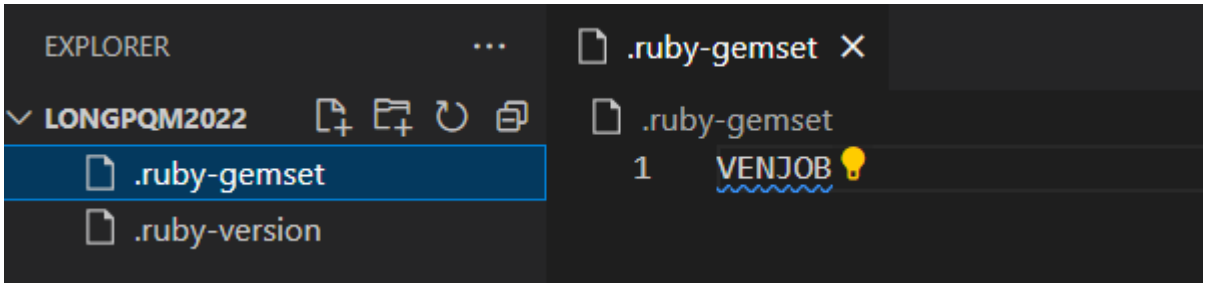


Figure 27: Add a line to .ruby-gemset file

- Next, open terminal in the project and type the following command:

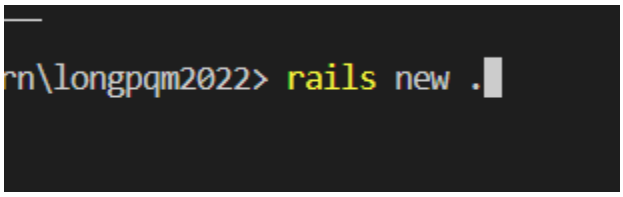


Figure 28: Creating Rails project command line

- When I finish typing commands, Rails will automatically create a Rails project structure so that developers can start using and building the web application they want. The reason I create and add lines to the 2 files .ruby-version and .ruby-gemset before running the command to create the project directory is so that Rails understands the Ruby version we will use and the name of the project.

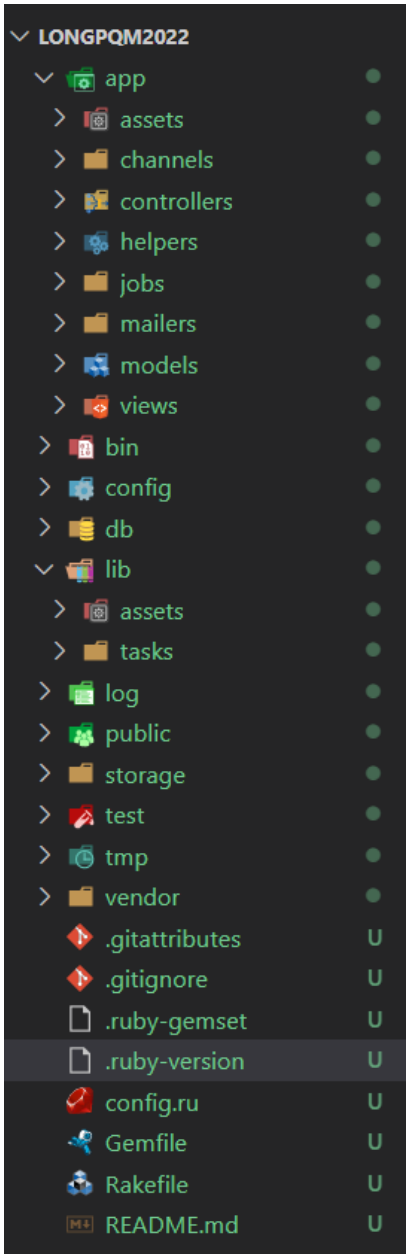


Figure 29: Project directory structure after successful initialization command

Since projects always have some files containing important information, we cannot push them all to Github because of security issues. Therefore, the ".gitignore" file is created to not bring highly sensitive files to Github. We will add paths and filenames that we don't want to put on Github.

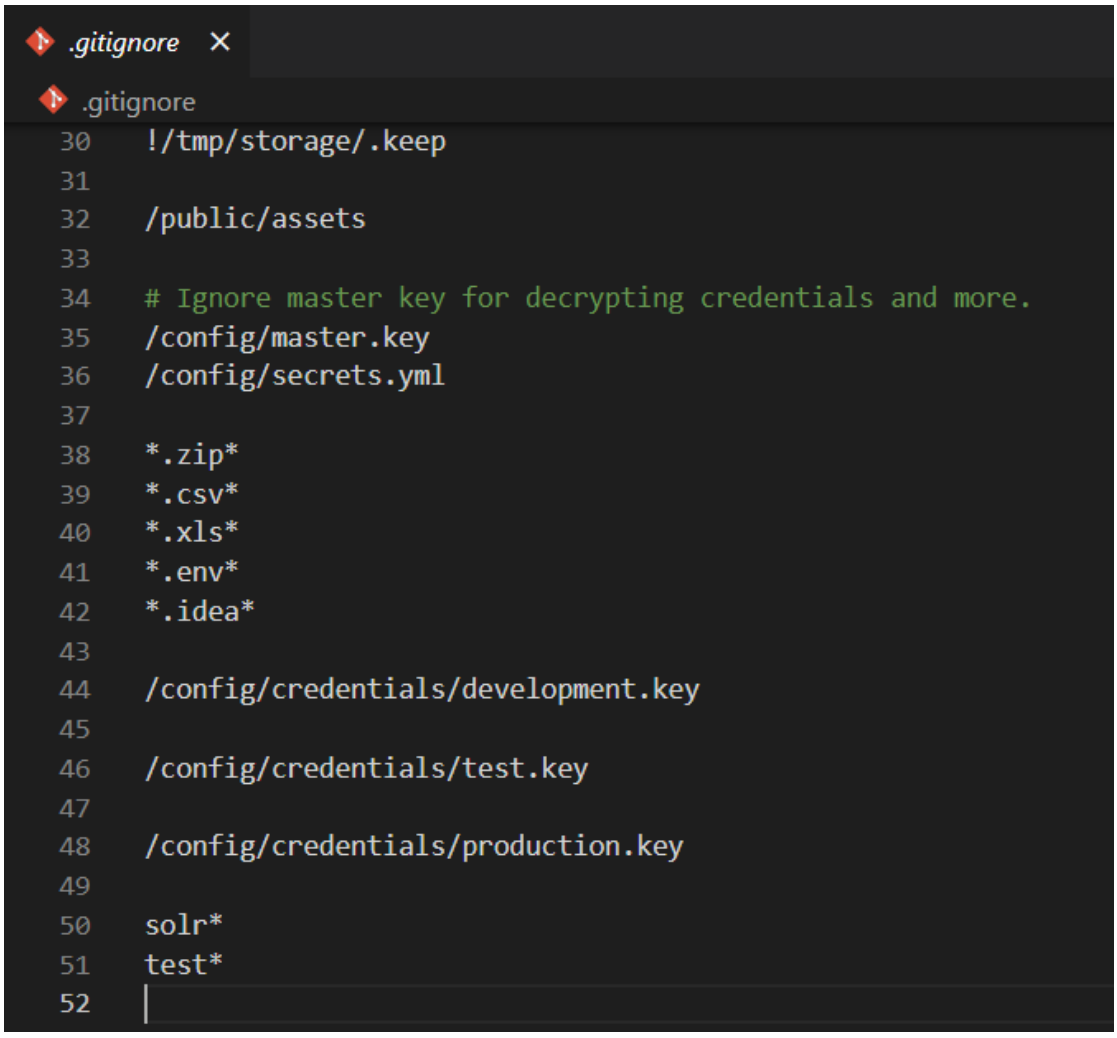


Figure 30: Sample of ".gitignore" file

Once we've initialized the project, if we want to run the website's server up to see if it's working properly, we type the command "rails server" into the console. If the server startup is successful, the following log will be displayed in the console:

```
longpqm@zigexn-minhlong:~/Desktop/report$ rails server
=> Booting Puma
=> Rails 7.0.4 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 5.6.5 (ruby 2.7.4-p191) ("Birdie's Version")
*  Min threads: 5
*  Max threads: 5
*  Environment: development
*          PID: 63848
* Listening on http://127.0.0.1:3000
* Listening on http://[::1]:3000
Use Ctrl-C to stop
```

Figure 31: Notification when running server successfully

Console shows the localhost path. I click to test, if successful, the screen will appear like this:

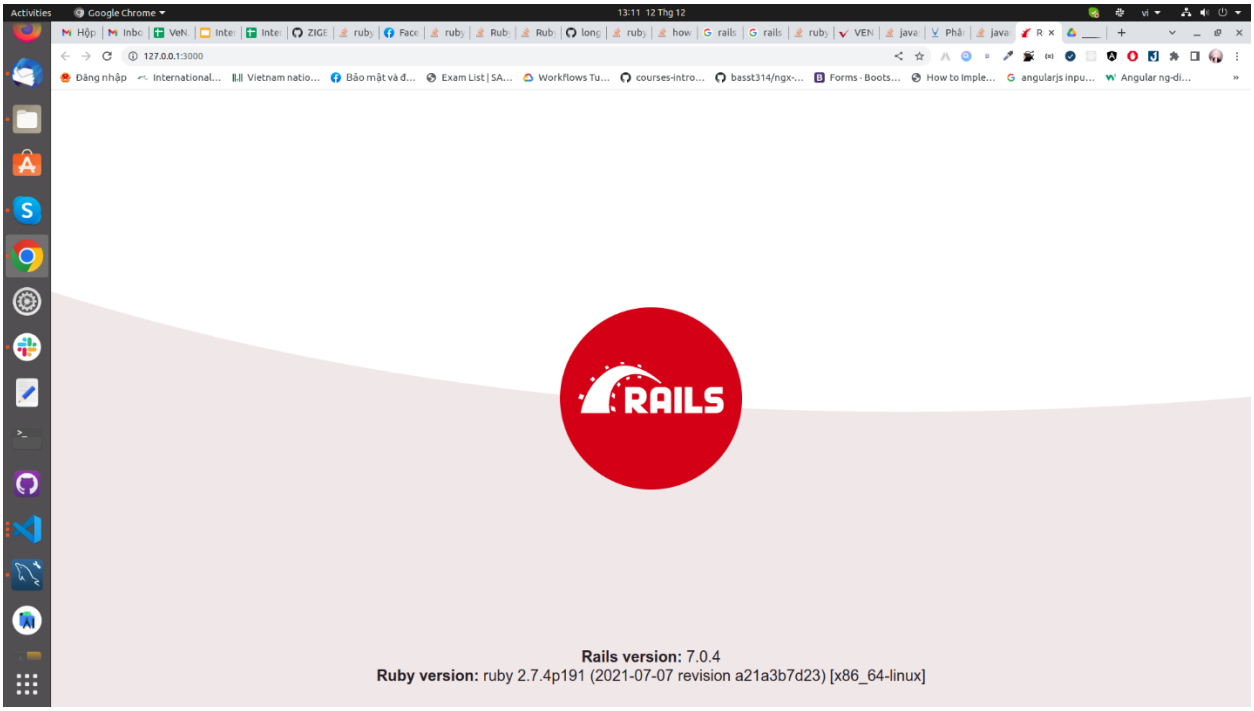


Figure 32: Original root page of Rails project

6.2. List the gems used for the project, download and install them

For the needs of the site, I found and wrote the gems to be used in the "Gemfile" file, then ran the "bundle install" (or just "bundle i") command in the console to install, then I could use their features in the project. Next time, if I need to use new gems, I just follow the same

steps; or if I want to remove gems I don't use then I just delete them in Gemfile and run "bundle i".

database.yml Gemfile M

Gemfile

You, 4 days ago | 1 author (You)

```
1   source "https://rubygems.org"
2   git_source(:github) { |repo| "https://github.com/#{repo}.git" }
3
4   ruby "2.7.4"
5
6   gem "mysql2"
7
8   gem 'countries', require: 'countries/global'
9
10  group :development do
11    gem 'better_errors'
12    gem 'binding_of_caller'
13  end
14
15  gem 'dotenv-rails'
16
17  gem 'kaminari'       # giúp phân trang
18  gem 'ransack'     # tạo form tìm kiếm
19  gem 'bootstrap-kaminari-views' # giao diện đẹp cho phân trang
20  gem 'bootstrap_form'
21  gem 'bootstrap4-kaminari-views'
22
23  gem 'jquery-rails'
```

Figure 33: List of some gems used in the project

6.3. Create a database and handle related operations

Since the project uses MySQL, we must configure the project data to be saved to MySQL (The original project will use SQLite). Go to database.yml file and configure.

database.yml

config > database.yml

You, 1 second ago | 1 author (You)

```
# SQLite. Versions 3.8.0 and up are supported.
#   gem install sqlite3
#
#   Ensure the SQLite 3 gem is defined in your Gemfile
#   gem "sqlite3"
#
default: &default
  adapter: mysql2      You, 1 second ago • Uncommitted changes
  host: <%= ENV['SECRET_HOST'] %>
  username: <%= ENV['SECRET_DBUSERNAME'] %>
  password: <%= ENV['SECRET_DBPASSWORD'] %>

  # https://stackoverflow.com/questions/59639750/how-to-get-default-collation-in-rails
  encoding: utf8mb4
  collation: utf8mb4_unicode_ci

development:
  <<: *default
  database: <%= ENV['SECRET_DB_DEVELOPMENT'] %>

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
```

Figure 34: Configuration in database.yml

Then, we go to the console and run the command "rake db:create" to create database in MySQL so that I can import data to MySQL, when successfully run it will show the following log:

```
PS C:\Users\Long\Desktop\ZIGEXN\
longpqm2022> rake db:create
Created database 'venjobs_develo
pment'
Created database 'venjobs_test'
PS C:\Users\Long\Desktop\ZIGEXN\
longpqm2022>
```

Figure 35: Console displays log after running the command to create database

6.4. Database migration

After creating the database successfully, I create database migrations for the data objects by lots of commands. For example: “rails generate migration create_jobs”.

```
C:\Users\Long\Desktop\ZIGEXN\demointern\longpqm2022> rails generate migration create_jobs
   invoke  active_record
   create    db/migrate/20221211165203_create_jobs.rb
C:\Users\Long\Desktop\ZIGEXN\demointern\longpqm2022>
```

Figure 36: Create database migration

Open the file at /db/migrate/<time_stamp>_create_jobs.rb and create the table fields:

20221108082427_create_jobs.rb

db > migrate > 20221108082427_create_jobs.rb > CreateJobs > change

You, 3 weeks ago | 1 author (You)

```ruby
class CreateJobs < ActiveRecord::Migration[7.0]
  def change
    create_table :jobs do |t|
      t.text :benefit
      t.string :company_id
      t.string :company_name
      t.string :company_address
      t.string :company_district
      t.string :company_province
      t.text :description
      # t.string :level
      t.string :name
      t.text :requirement
      t.string :salary
      t.string :contact_email
      t.string :contact_name
      t.string :contact_phone
      t.index [:company_id, :name], unique: true
      t.timestamps      You, 3 weeks ago • update import working task …
    end
  end
end
```

Figure 37: Create the table fields

Once we have created the fields, we run the command "rake db:migrate". When running successfully, the console will display the following log:

```
== 20221211165203 CreateJobs: migrating ==============================
-- create_table(:jobs)
   -> 0.0080s
== 20221211165203 CreateJobs: migrated (0.0111s) =====================

PS C:\Users\Long\Desktop\ZIGEXN\demointern\longpqm2022>
```

Figure 38: Console will show log after migration

I perform the same database migration with data objects like industries, cities, types and levels, ... I can also add, edit, delete or customize data types (with some conditions) for fields using database migration depending on the requirements of the project.

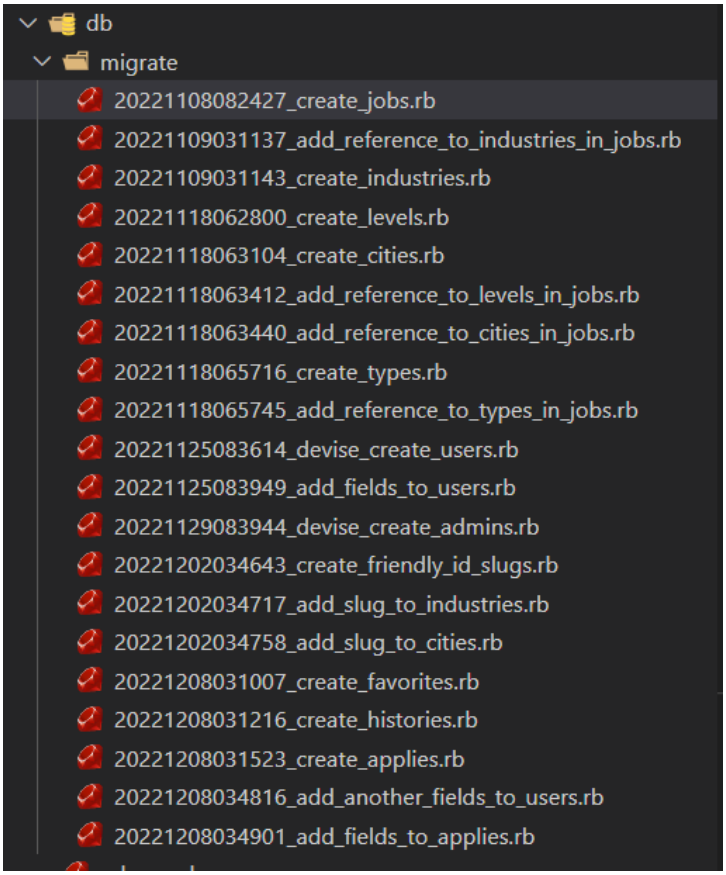


Figure 39: Migration files generated up to the time of the internship report

I can go to the "schema.rb" file to check the fields of the data tables that I have done the migration.

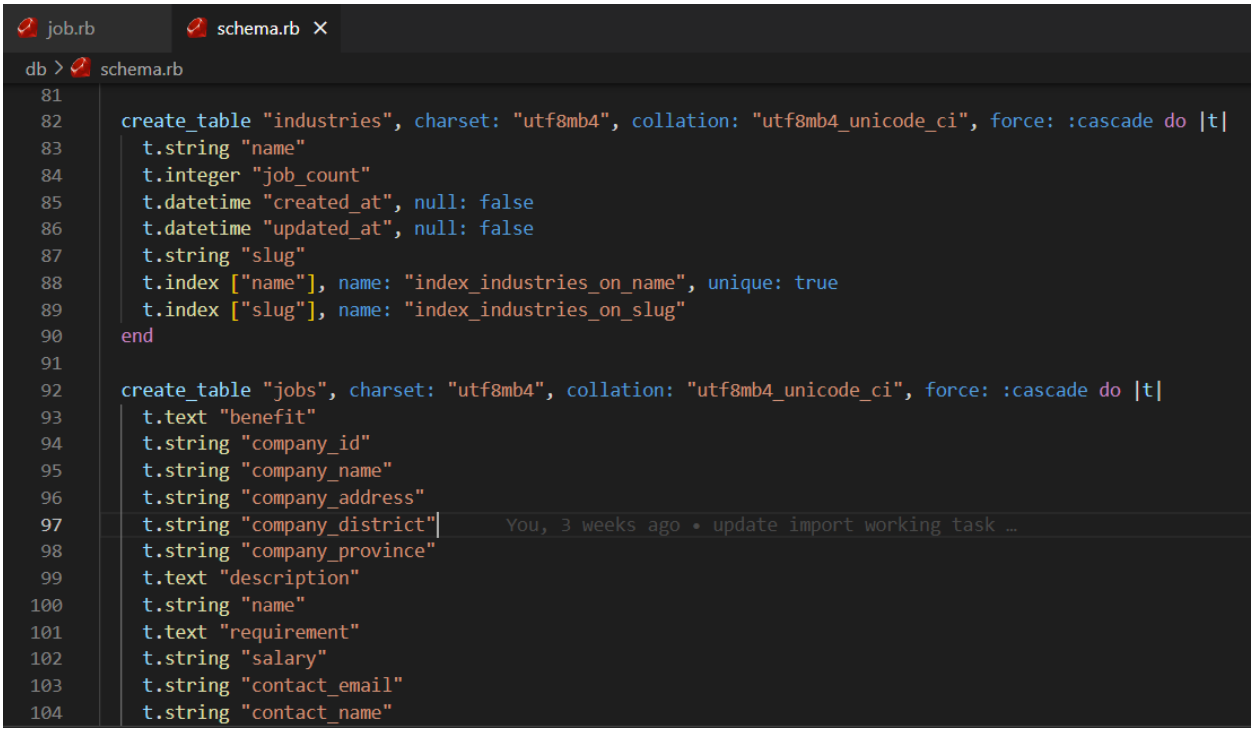


Figure 40: "schema.rb" shows the fields of the data tables

6.5. Modeling

I wrote a model for the project's data objects so that when importing data from a CSV file to MySQL, Rails and MySQL will understand the data and the relationship between them so that they will do the import quickly and reasonably.

⇨ Contribute to faster queries, less data consumption and the web will also run better.

Write model for “Job”:

```
class Job < ActiveRecord::Base
    belongs_to :industry
    belongs_to :level
    belongs_to :type
    belongs_to :city
    has_many :favorites
    has_many :applies
    has_many :histories

    before_save :strip_html_from_texts

    def strip_html_from_texts
        self.description =
ActionView::Base.full_sanitizer.sanitize(self.description)
```

```
        self.requirement =
ActionView::Base.full_sanitizer.sanitize(self.requirement)

        self.benefit = ActionView::Base.full_sanitizer.sanitize(self.benefit)
    end

    searchable do
        text :name, stored: true, boost: 55
        text :benefit, stored: true, boost: 25
        text :description, stored: true, boost: 45
        text :requirement, stored: true, boost: 15
        text :city_name, stored: true, boost: 54 do |i|
           i.city.name unless i.city.nil?
        end
        text :industry_name, stored: true, boost: 53 do |i|
           i.industry.name unless i.industry.nil?
        end

    end

end
```

The way to write models for the rest of the data objects is similar, depending on the data type and the relationship between them.

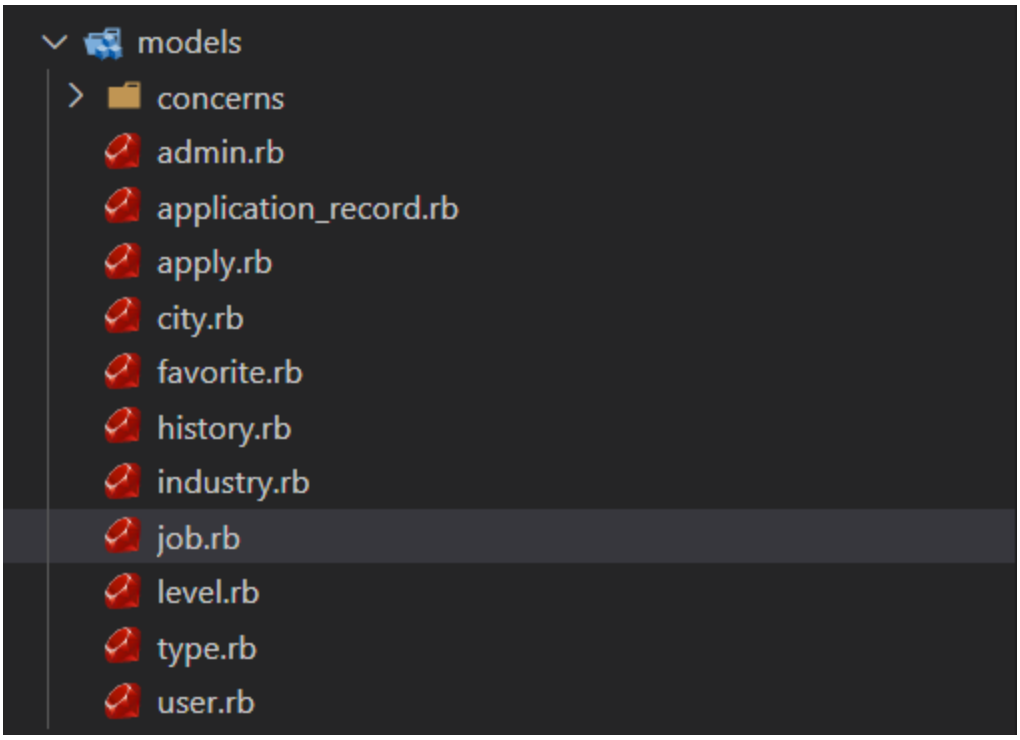


Figure 41: Project models (until December 12th, 2022)

6.6. Import CSV to database

After performing migrations and writing models for data objects, I can import data. I have programmed to execute "CSV import" task in "importwork.rake" file with 2 self-defined classes, DownloadData and ExtractData. Because FTP information must be kept confidential, information such as FTP_SERVER, FTP_USERNAME, FTP_PASSWORD has been saved to the environment variable.

The file "importwork.rake" is saved in the "lib/tasks" path of the project. In addition, there are 2 classes, DownloadData, ExtractData for import, and SlugWork, which is used to create permalink links to the source page of the website.

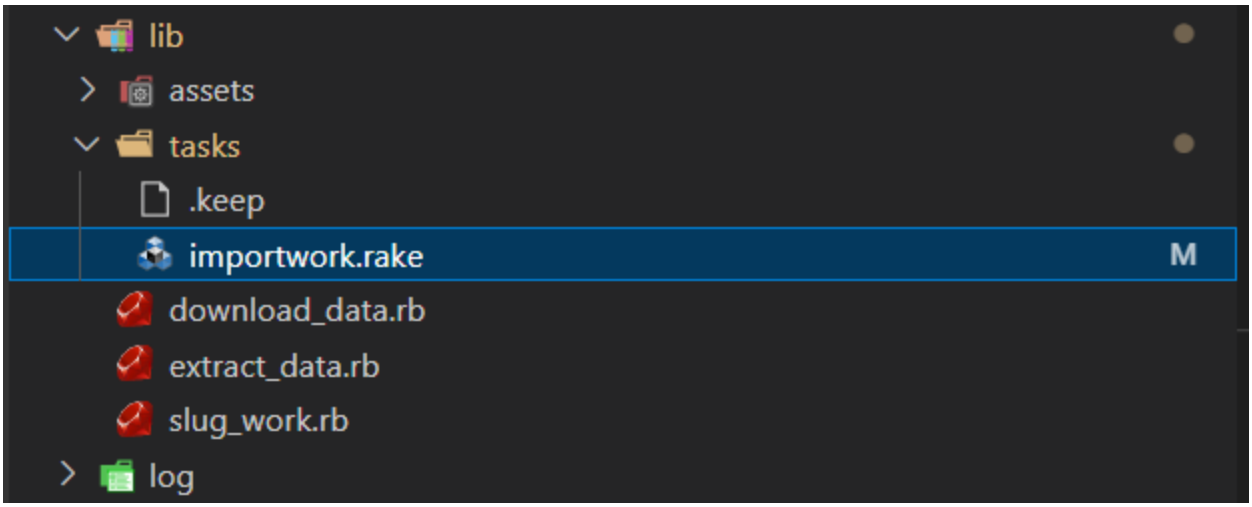


Figure 42: "importwork.rake" file is saved in the "lib/tasks" path of the project

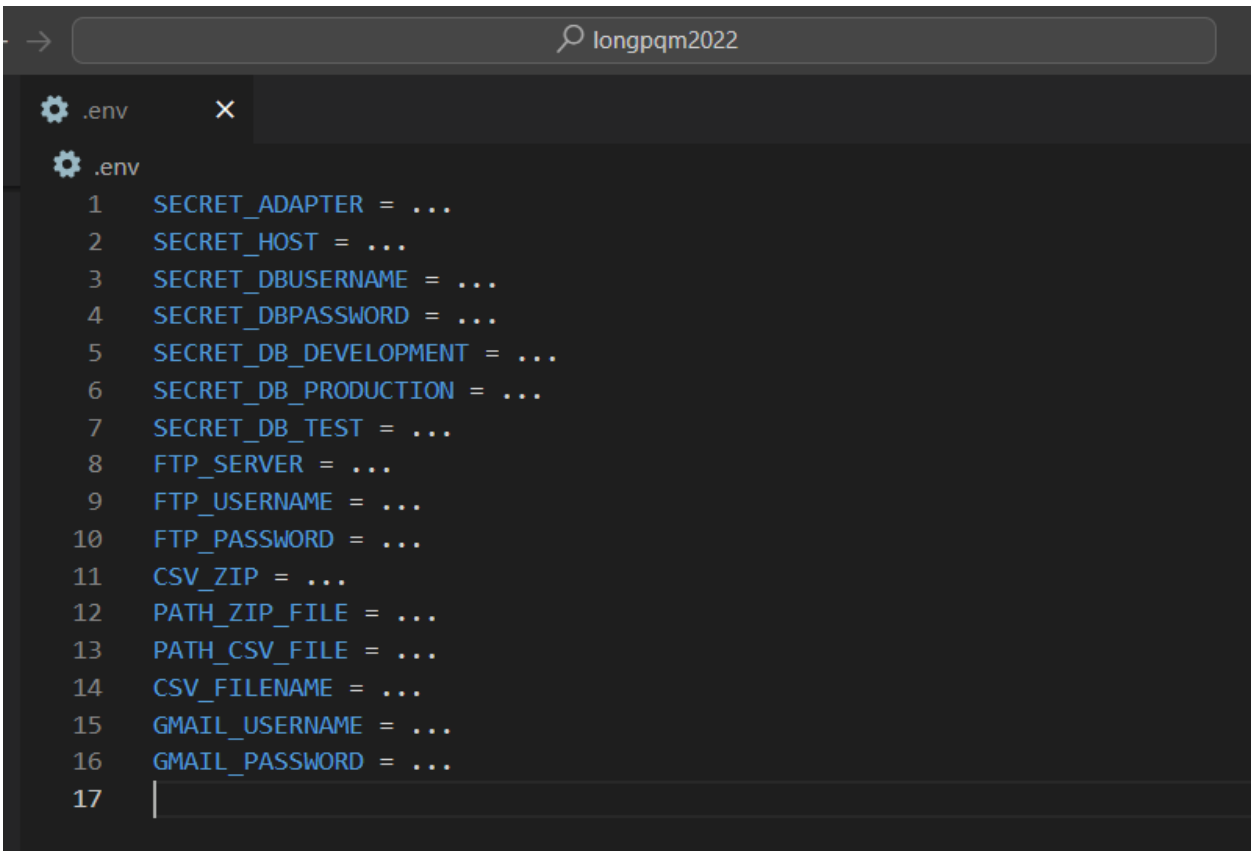


Figure 43: Confidential info has been saved to the environment variable (illustration)

Here are the lines of code that I have executed:

```
require './lib/download_data.rb'
require './lib/extract_data.rb'
require 'csv'
```

```ruby
namespace :importwork do
  desc "TODO"

  task import_db: :environment do

    DownloadData.new.downloadzipfile(ENV['FTP_SERVER'], ENV['FTP_USERNAME'],
ENV['FTP_PASSWORD'], ENV['CSV_ZIP'], ENV['PATH_ZIP_FILE'])

    ExtractData.new.extractzipfile(ENV['CSV_ZIP'], ENV['PATH_CSV_FILE'])

    industry_list = Array.new
    level_list = Array.new
    type_list = Array.new
    work_city_list = Array.new
    jobs_list = Array.new
    file = ENV['CSV_FILENAME']

    CSV.foreach(file, headers: true) do |row|

      cate = Industry.new(name: row['category'])
      industry_list << cate if cate.valid?

      level  = Level.new(name: row['level'])
      level_list << level if level.valid?

      type  = Type.new(name: row['type'])
      type_list << type if type.valid?

      cities = ISO3166::Country.find_country_by_alpha2('VN').subdivisions
      set_cities=[]

      cities.each do |name, city|
        set_cities << city.code
        set_cities << city.name
        set_cities << city.translations['en']
      end

      add_more_correct_cities = ["16", "Bắc Cạn" , "Xã Xuân Giao", "Hồ Chí Minh",
"Bà Rịa - Vũng Tàu", "Cần Thơ", "Đà Nẵng", "Hà Nội", "Hải Phòng", "Thừa Thiên
Huế"]
      add_more_correct_cities.each {|city| set_cities << city}

      work_city_name = row['work place'].gsub(/[\[\]\"]/, '')
      country = set_cities.include?(work_city_name) ? "Vietnam" : "International"
```

```
      work_city = City.new(name: work_city_name, country: country)
      work_city_list << work_city if work_city.valid?
    end
    Industry.import industry_list, on_duplicate_key_ignore: true, validate: false
    Level.import level_list, on_duplicate_key_ignore: true, validate: false
    Type.import type_list, on_duplicate_key_ignore: true, validate: false
    City.import work_city_list, on_duplicate_key_ignore: true, validate: false

    # Import for jobs

    h_industries = Industry.select(:id, :name).index_by(&:name)
    h_levels = Level.select(:id, :name).index_by(&:name)
    h_types = Type.select(:id, :name).index_by(&:name)
    h_work_cities = City.select(:id, :name).index_by(&:name)
    CSV.foreach(file, headers: true) do |row|
      work_city_name = row['work place'].gsub(/[\[\]\"]/, '')
      jobs = Job.new({
        benefit: row['benefit'],
        industry: h_industries[row['category']],
        company_address: row['company address'],
        company_district: row['company district'],
        company_id: row['company id'],
        company_name: row['company name'],
        company_province: row['company province'],
        description: row['description'],
        level: h_levels[row['level']],
        type: h_types[row['type']],
        name: row['name'],
        requirement: row['requirement'],
        salary: row['salary'],
        contact_email: row['contact email'],
        contact_name: row['contact name'],
        contact_phone: row['contact phone'],
        city_id: h_work_cities[work_city_name].try(:id)
      })

      jobs_list << jobs if jobs.valid?

    end

    Job.import jobs_list, on_duplicate_key_update: [:benefit, :description,
:requirement, :salary], validate: false

    Job.reindex
```

```
    ## Do Solr for cities
    a_cities = []
    all_city = City.all
    all_city.each do |item|
      kq = Job.search do
        fulltext item.name
      end

      item.job_count = kq.total
      a_cities << item

    end

    City.import a_cities, on_duplicate_key_update: [:job_count], validate: false

    ## Do Solr for industries
    a_industries = []
    all_industries = Industry.all
    all_industries.each do |item|
      kq = Job.search do
        fulltext "#{item.name}"
      end

      item.job_count = kq.total
      a_industries << item

    end

    Industry.import a_industries, on_duplicate_key_update: [:job_count],
validate: false

  end

end
```

To execute the file's action, I run this command in the console pointing to the project's path: rake "importwork:import_db".

After successfully importing the data, we can go to MySQL, execute a query to check if the data has been imported and is correct or not.

```
2 •  select * from jobs
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

description	name	requirement	salary	contact_email
I. Tích cách chung của ứng viên - Có thái độ làm...	Cửa hàng trưởng	- Có ít nhất 2 - 3 năm kinh nghiệm trong ngành ...	Thỏa thuận	phuongtram@vnpi
• Theo dõi và lập báo cáo hàng tồn kho (Nhập, ...	KẾ TOÁN TỔNG HỢP	• Tốt nghiệp đại học chuyên ngành kế toán-kiế...	Thỏa thuận	haypt@vng.com.v
-Phát triển nhận dạng thương hiệu cho một chu...	Họa viên thiết kế 3D (Thực tập viên)	- Sinh viên năm 4, 5 - Sinh viên đại học về chuy...	Thỏa thuận	phuongtram@vnpi
- Phát triển và chăm sóc khách hàng là các nhà ...	Cộng tác viên tư vấn tài chính	- Kỹ năng giao tiếp, đàm phán - Kỹ năng phân tí...	Thỏa thuận	thanhtdn@vgx.vn
- Hiểu rõ ưu, nhược điểm sản phẩm của Công ty...	Cộng Tác Viên	- Có laptop để hướng dẫn khách hàng. - Sử dụn...	3 - 5 triệu	thanhtdn@vgx.vn
- Phát triển khách hàng mới. - Chăm sóc khách ...	Nhân viên kinh doanh và Cộng tác viên	- Ứng viên không phân biệt nam hay nữ. - Tuổi ...	3 - 5 triệu	thanhtdn@vgx.vn
Responsibilities - Coordinate all new packaging ...	Packing Engineer	Requirements - BS in Packaging Engineering, Ma...	15.000.000	thao.nguyen@mar
- Ứng viên không phân biệt nam hay nữ. - Tuổi ...	Trưởng phòng Kinh doanh	Có kinh nghiệm thị trường Forex là lợi thế. Tôn t...	7 – 10 triệu	thanhtdn@vgx.vn
• Tìm kiếm, tiếp cận và phát triển khách hàng • ...	Nhân viên kinh doanh	- Kỹ năng vi tính văn phòng thành thạo - Đã tìm...	3 - 5 triệu	linhnt@vgx.vn
- Report to Marketing Director - Manage and lea...	Marketing Manager	- University Degree. - 3 -5 years experience in ...	1500-2000$	hong.nguyen@ma
Tìm kiếm khách hang tiềm năng, khách đoàn, kh...	Marketing, sale tour du lich	Trung thực, nhanh nhẹn, thảo vát, có khả năng...	Thỏa thuận	director@sunsmile
Responsibilities - Monitors the performance of q...	Assistant Production Manager (Cosmetic or Pha...	Requirements - Degree in Mechanical Engineerin...	1000$	thao.nguyen@mar
Responsibilities - Responsible for instruction wor...	Warehouse Supervisor (Cosmetic/Pharma/FMCG)	Requirements - Graduated university - Minimum ...	1000$	thuy.nguyen@mar
Responsibilities - Cooperate with QA and other ...	Senior R and D (Pharma and Cosmetics)	Requirements - Bachelor in Chemistry / biochemi...	1000$	thao.nguyen@mar
Giao dịch với các nhà sản xuất nước ngoài, làm ...	Nhân viên xuất nhập khẩu kiêm trợ lý giám đốc	Nữ, học các khối kinh tế, - Biết tiếng Anh, thàn...	thỏa thuận	joey.hsu2@gmail.c

Figure 44: Data is successfully imported into MySQL

I also import data on Solr. Solr will build an indexer to store all the words it can handle from the text content. When searching, it will rely on this index. The text will be split into tags based on spaces and other characters (StandardTokenizer).

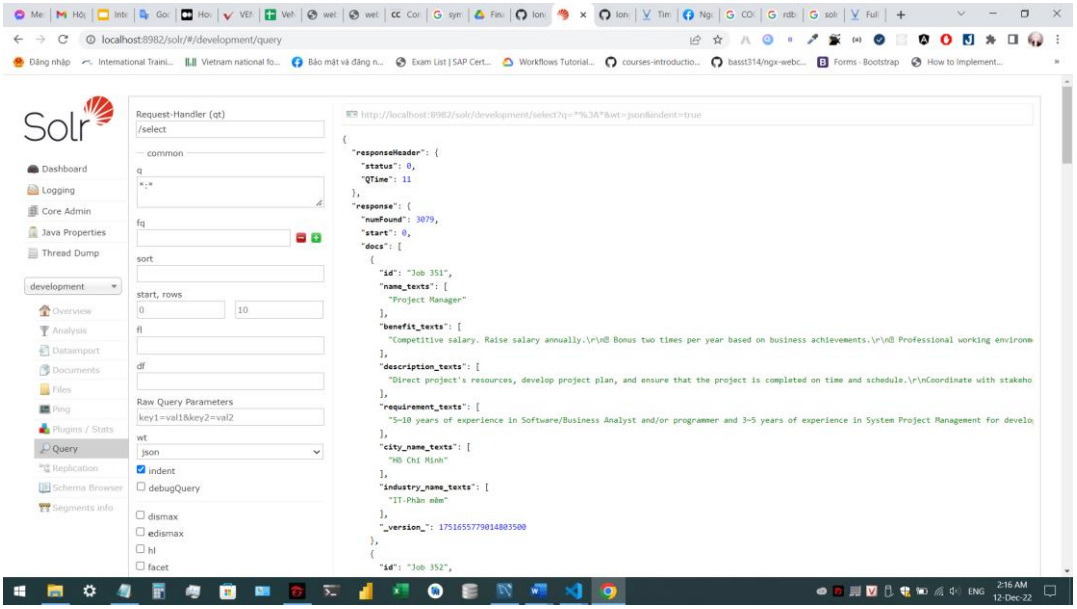


Figure 45: Solr localhost

6.7. Controller and view

After successfully importing the data, I continue to write code on the controller of the data objects in the project based on the descriptions and requests and define the routes.

As required, the homepage should list the total number of jobs available, listing the 5 newest jobs, the top 9 cities with the highest number of jobs, and the top 9 occupations with the highest number of jobs. The homepage must have a search bar for users to find jobs. I used Solr for the job for searching. Write codes for “Job” controller:

```
require './lib/slug_work.rb'
class JobsController < ApplicationController
    def index
        @jobs_count = Job.count
        @favoritesCount = Favorite.select("count(job_id)").where("user_id = ",
current_user.id) if user_signed_in?
        @jobs_latest = Job.order(created_at: :desc).limit(5)
        @cities_top = City.select(:id, :name, :job_count).order(job_count:
:desc).limit(9)
        @industries_top = Industry.select(:id, :name,
:job_count).order(job_count: :desc).limit(9)

        @query = Job.search do
            fulltext params[:search]
            paginate page: params[:page], per_page: 20
        end

        @params_var = params[:search]
        @jobs_search = @query.results
        @total_search = @query.total

    end

    def show
        @job = Job.find(params[:id])
        if user_signed_in?
            if History.exists?(job_id: @job.id, user_id: current_user.id)
                History.where(user_id: current_user.id, job_id:
@job.id).destroy_all
            end
            @user_histories = History.new(user_id: current_user.id, job_id:
@job.id )
            @user_histories.save!
        end
    end

    def search
```

```
        @click = request.path.include?('/jobs/city/') ?
City.find(SlugWork.new.slug_work(params[:city])) :
Industry.find(SlugWork.new.slug_work(params[:industry]))
        @jobs = request.path.include?('/jobs/city/') ?
Job.includes([:city]).where('city_id = ?', @click) :
Job.includes([:industry]).where('industry_id = ?', @click)
        @query = @jobs.page(params[:page]).per(20)
    end

end
```

Depending on the needs and circumstances, I would write controllers for each data object similar to what I did with "Jobs".

I will also define routes (paths, pointers to actions, HTTP request methods...) to match the request from the management. ID1 (Top page – Job main page) is the root path of webpage.

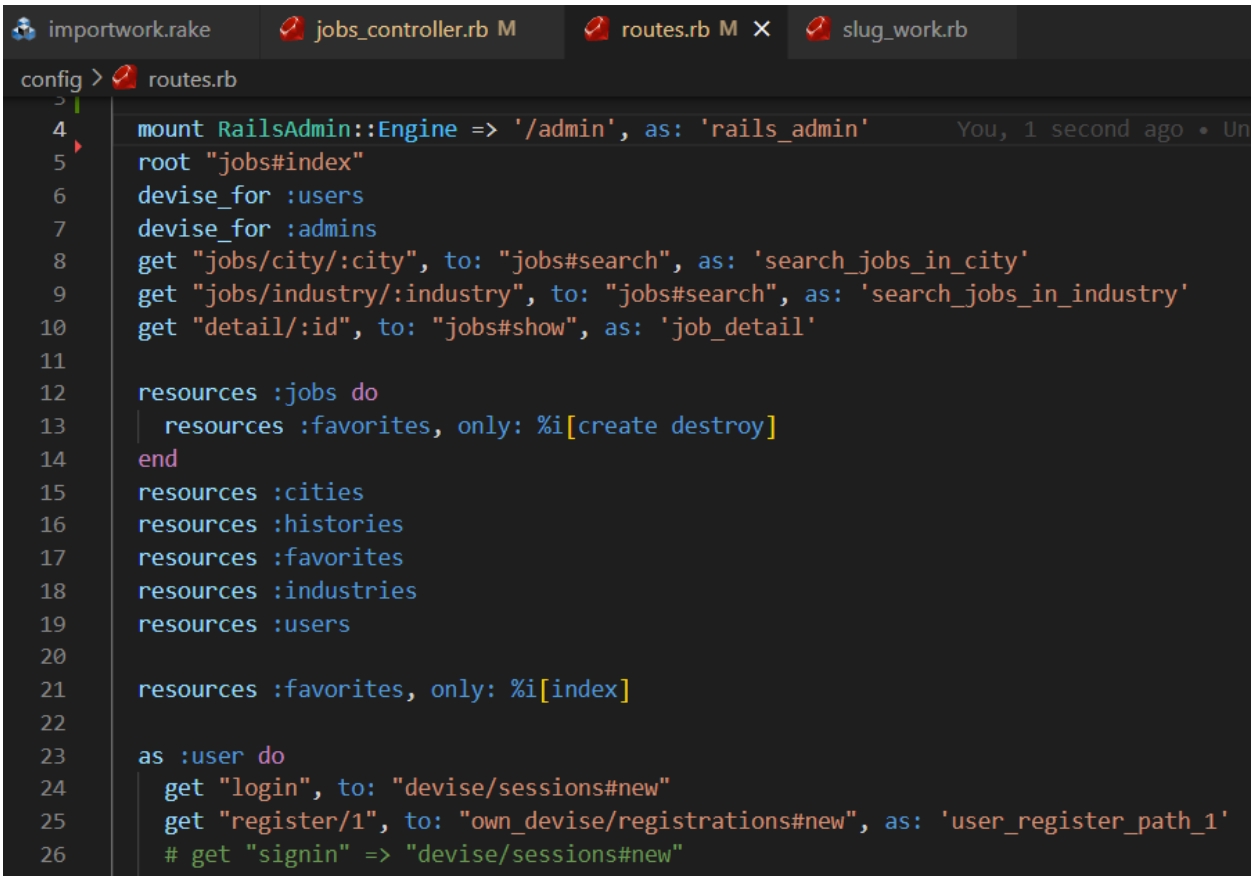


Figure 46: Config routes in "routes.rb"

Within the content of a layout, yield determines the sections where content from the view can be added. The simplest way is to use the yield keyword, which will add the entire current content of the view that is being rendered. Yield can also insert content thanks to the symbols passed to it.

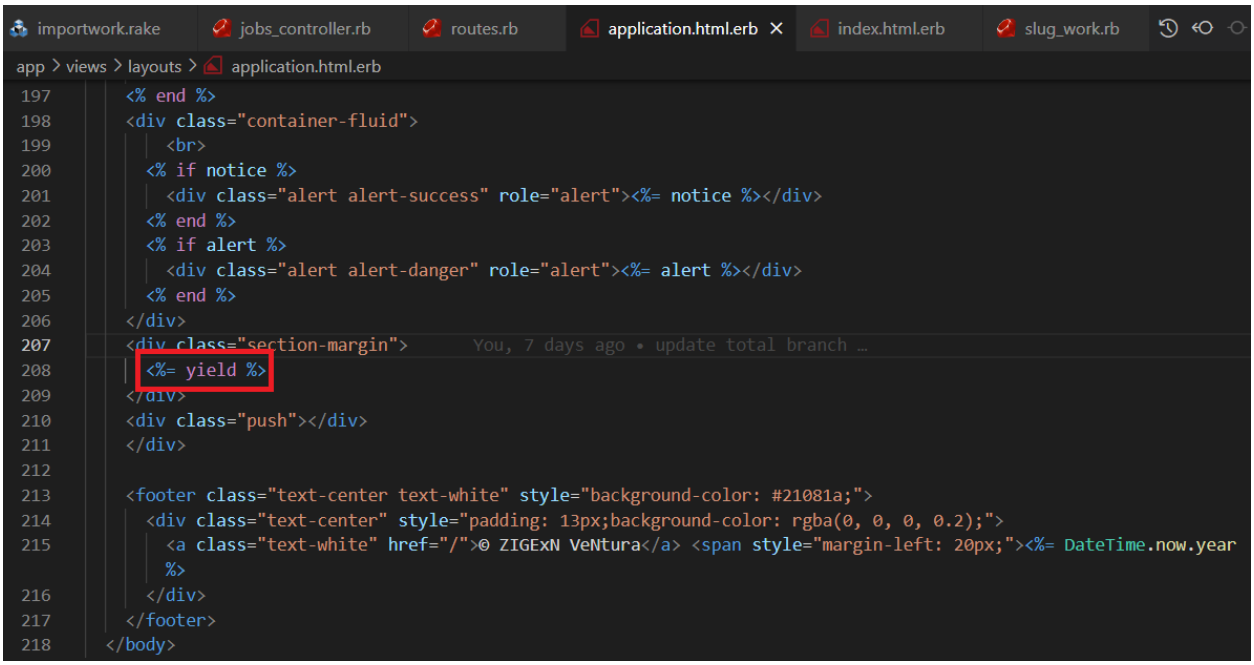


Figure 47: Yield in application.html.erb

6.7.1. TOP (Job main root page)

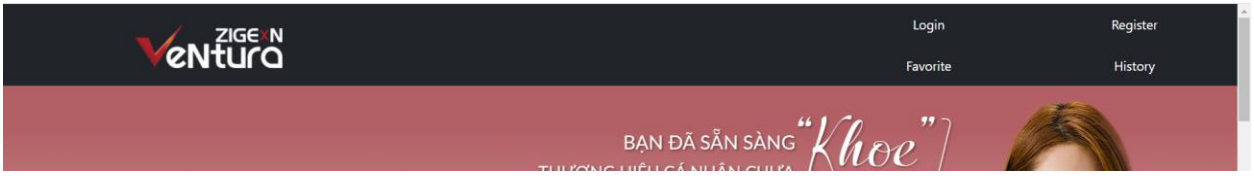


Figure 48: The header bar when the user is not logged in

Figure 49: The header bar when the user is not logged in (mobile view)

Figure 50: The header bar when user is logged in

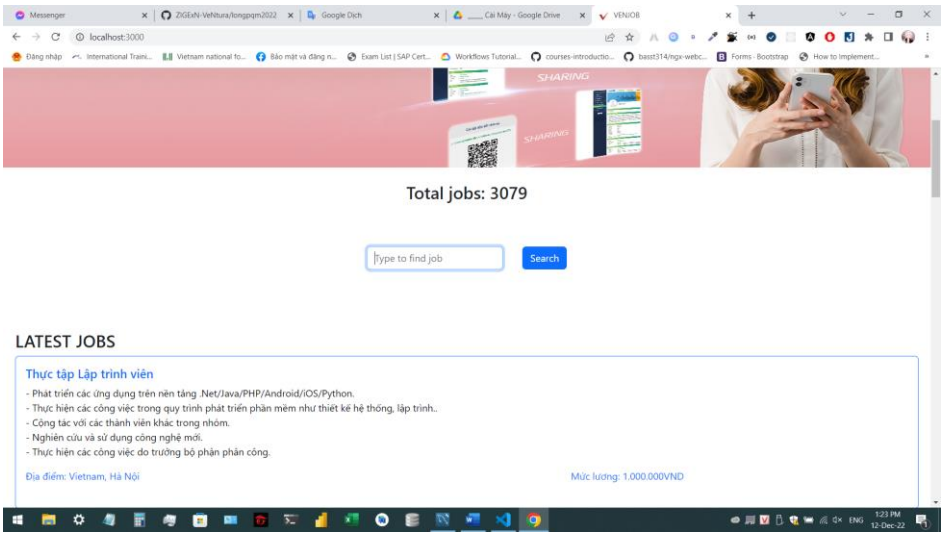


Figure 51: Key visual image, Total jobs, search bar and 5 Latest jobs in main page

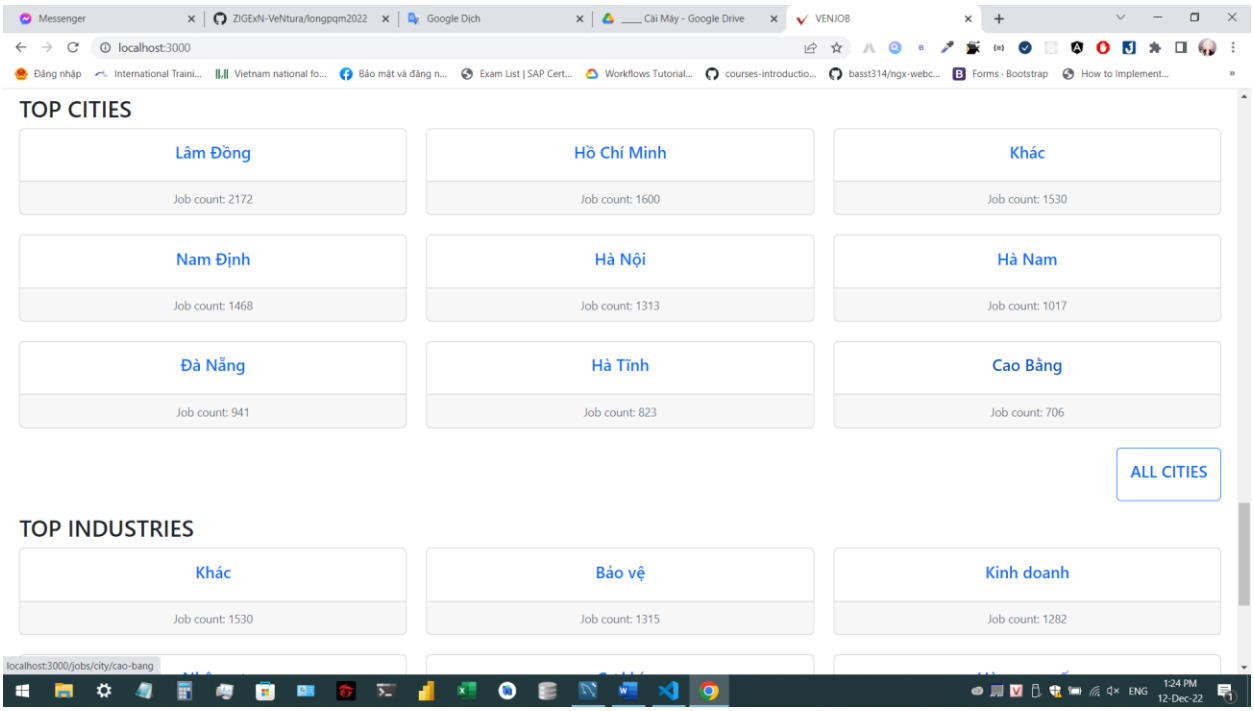


Figure 52: "Top Cities" and "Top Industries" with each job count

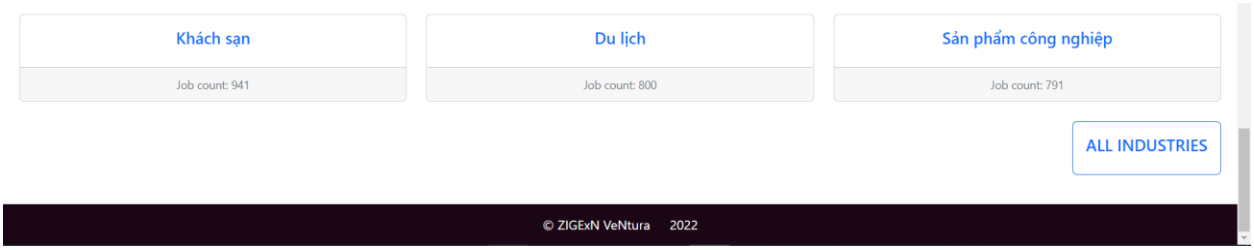


Figure 53: Simple footer

6.7.2. City list + Industry list + Job list

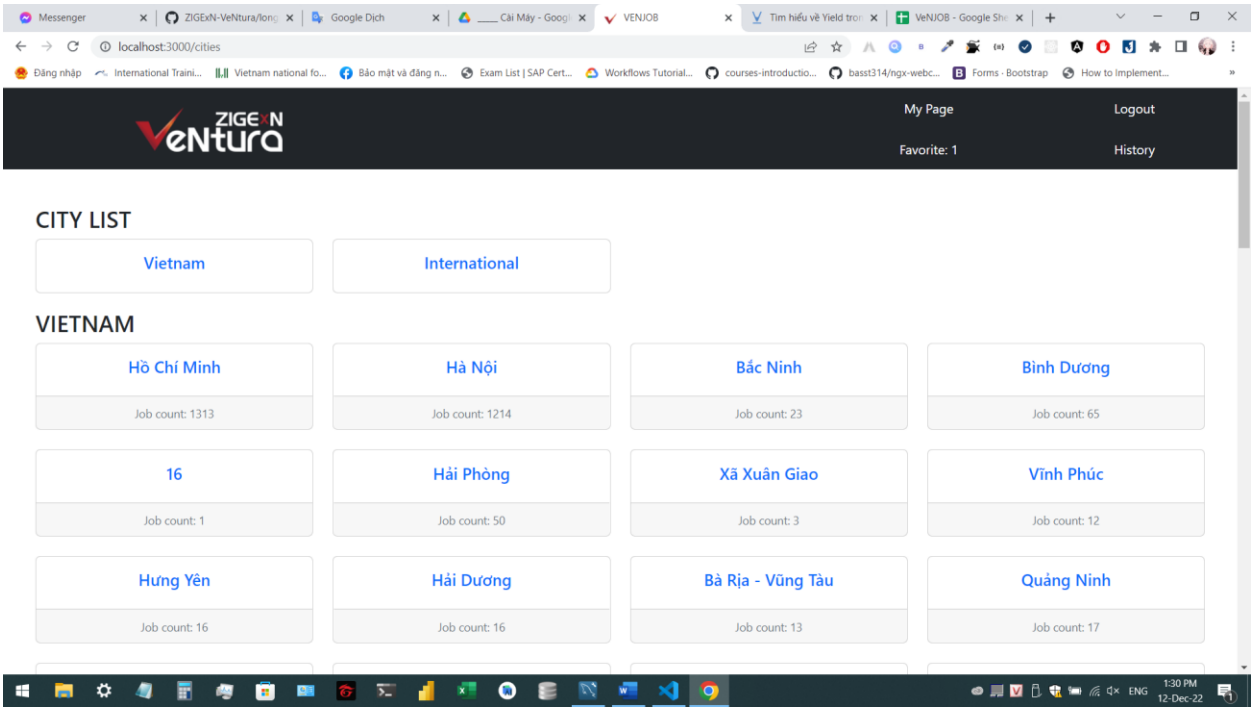


Figure 54: City list page

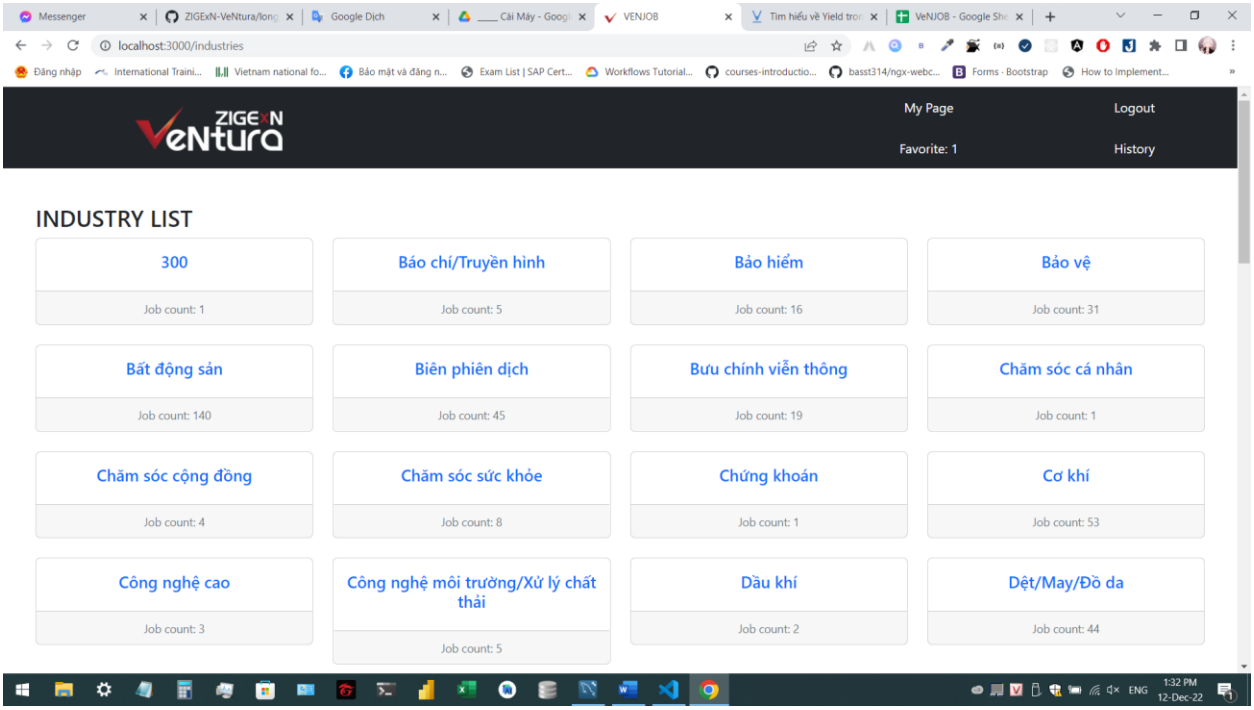


Figure 55: Industry list page

When we click on a city or an industry, it will show all the jobs for that city or industry.

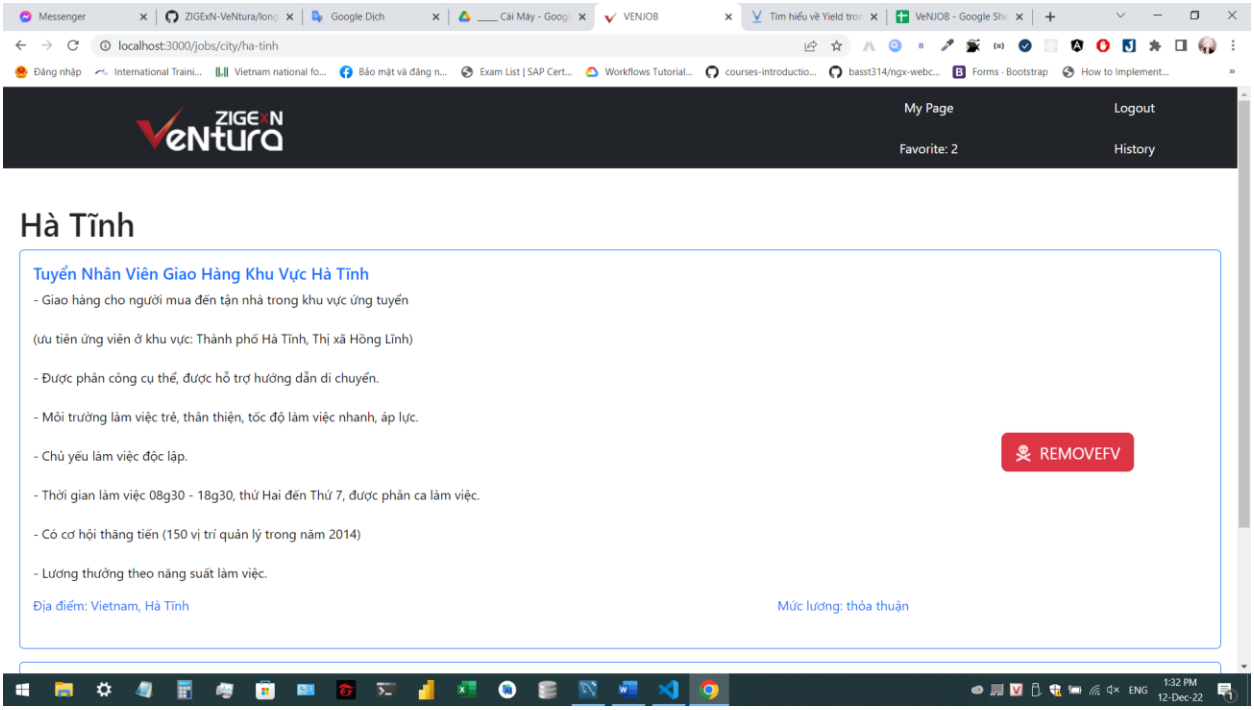


Figure 56: Show all jobs in the city (or industry)

Or if we search for jobs by certain keywords in the search bar, it will also return job results based on the keywords we type (Solr did those tasks). There will be pagination if the results are more than 20, similar to when we search for jobs by city or industry (we specify a maximum of 20 jobs for 1 page).

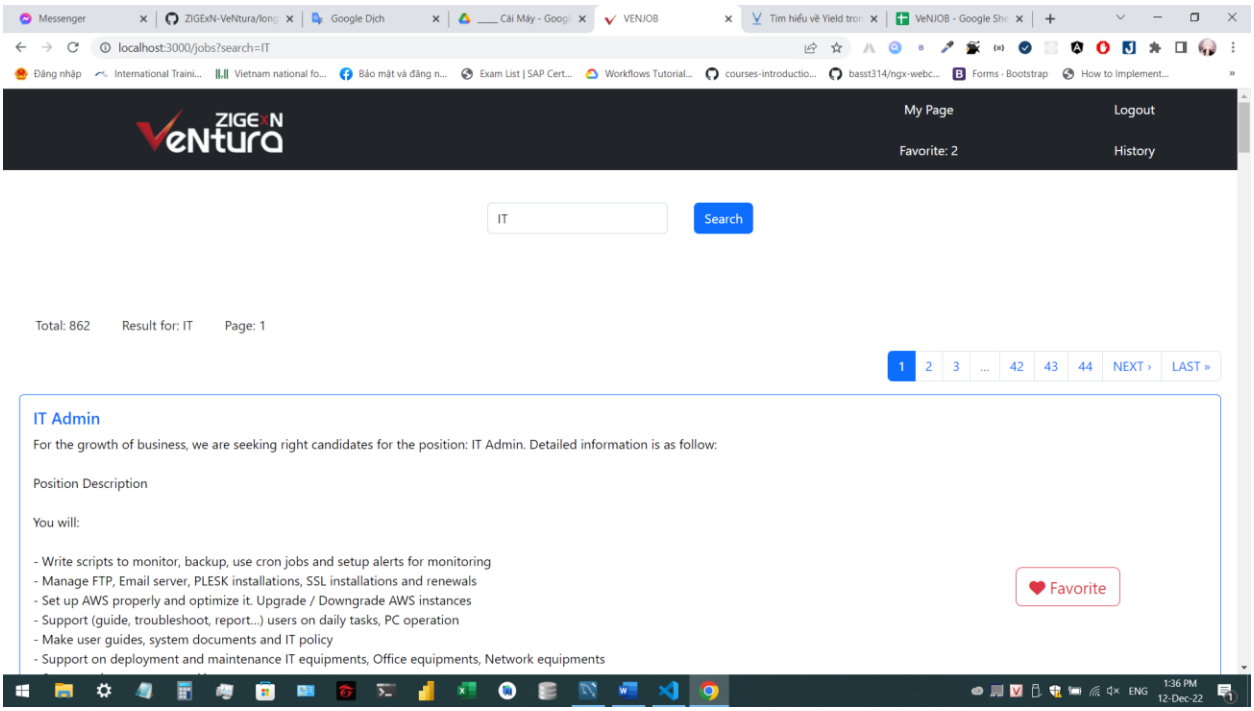


Figure 57: Search results

6.7.3. Job Detail

When we click on the job name, it will redirect us to the job description detail page.

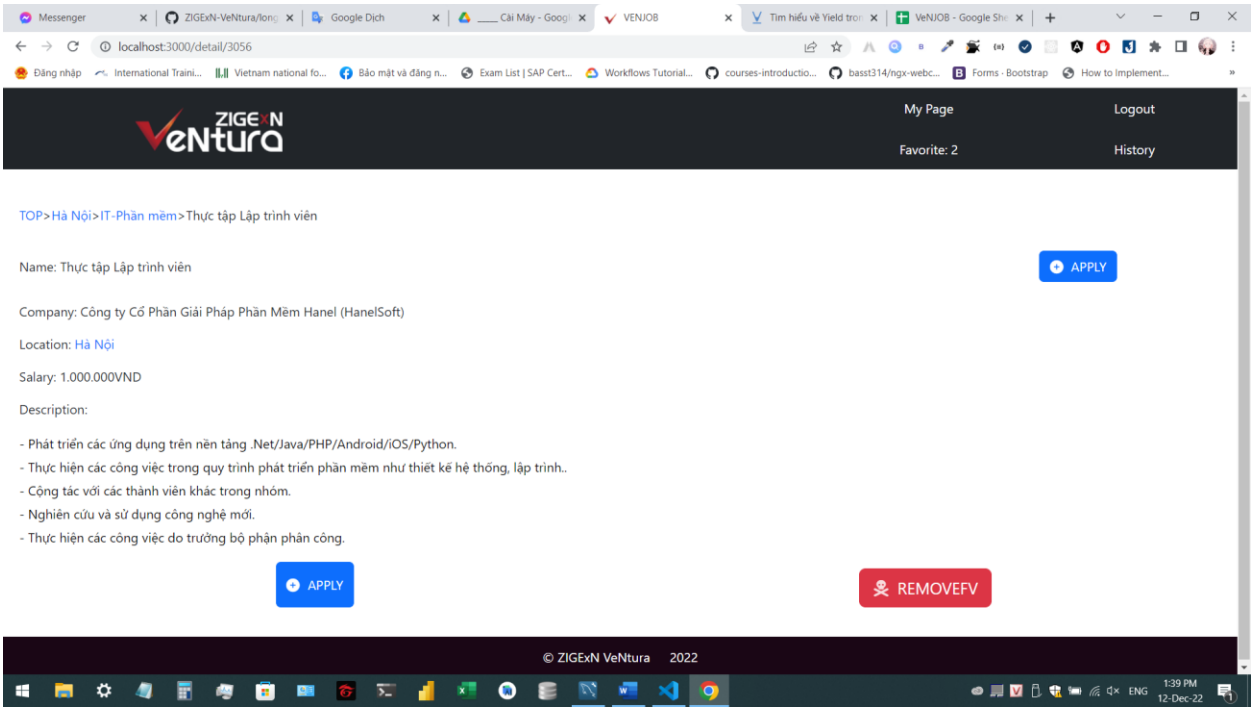


Figure 58: Job Detail page

On some pages, there will be apply and favorite buttons. If the user is not logged in and clicks, the system will switch to the login page, otherwise, it will perform the actions of those buttons.

6.7.4. Favorite jobs + History jobs

When a user has logged in and pressed the favorite button for a job, it will save the information to the database and display it on the favorites page for each of those users. User can also delete favorite jobs in favorites page by clicking **REMOVEFV** button, it will display successfully deleted message to user.

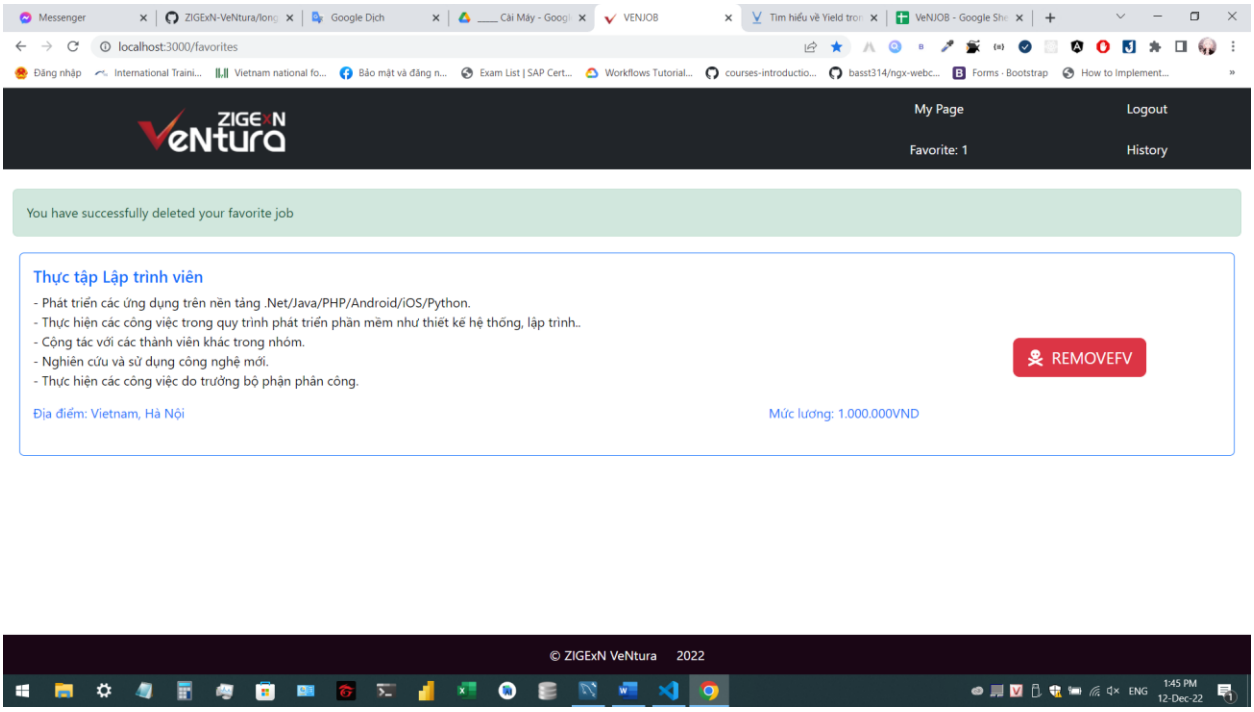


Figure 59: Favorite page

When the user has logged in and clicks on the details page of the job, the system will save that history in the database and display it on the history page (the way it works is quite similar to favorite).

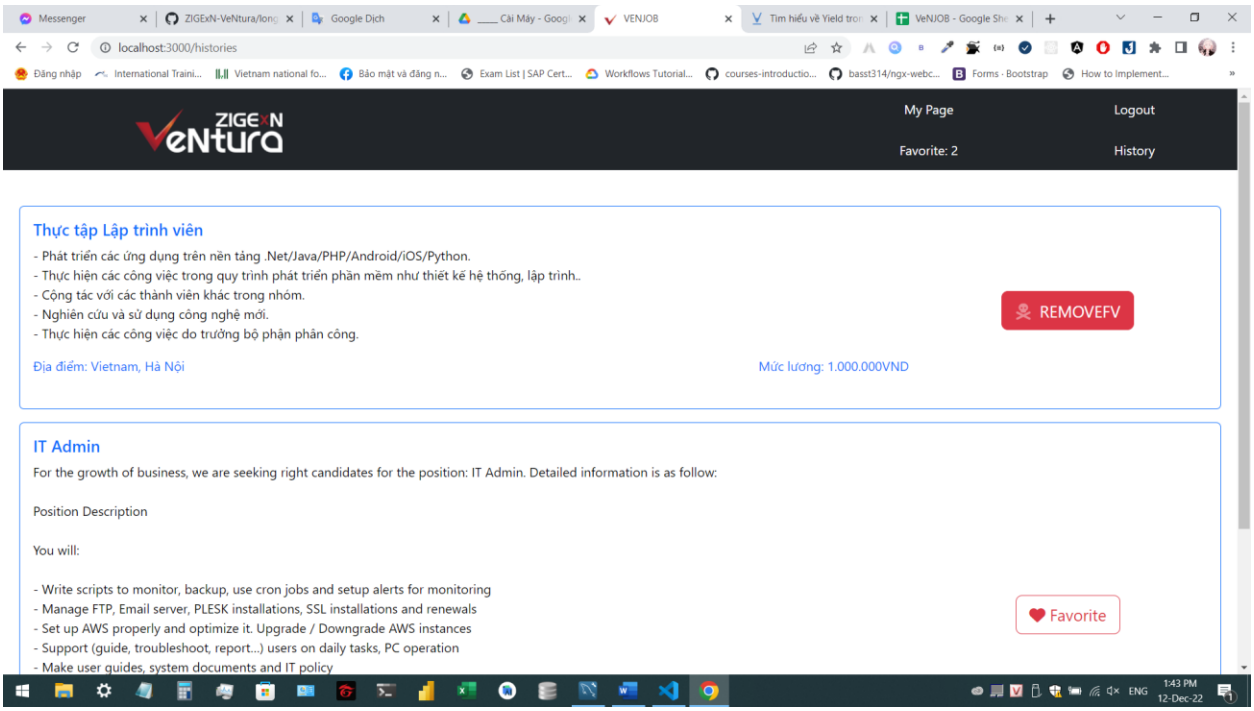


Figure 60: History page of each user

6.7.5. Admin page

There is a login page for admin (http://localhost:3000/admins/sign_in).

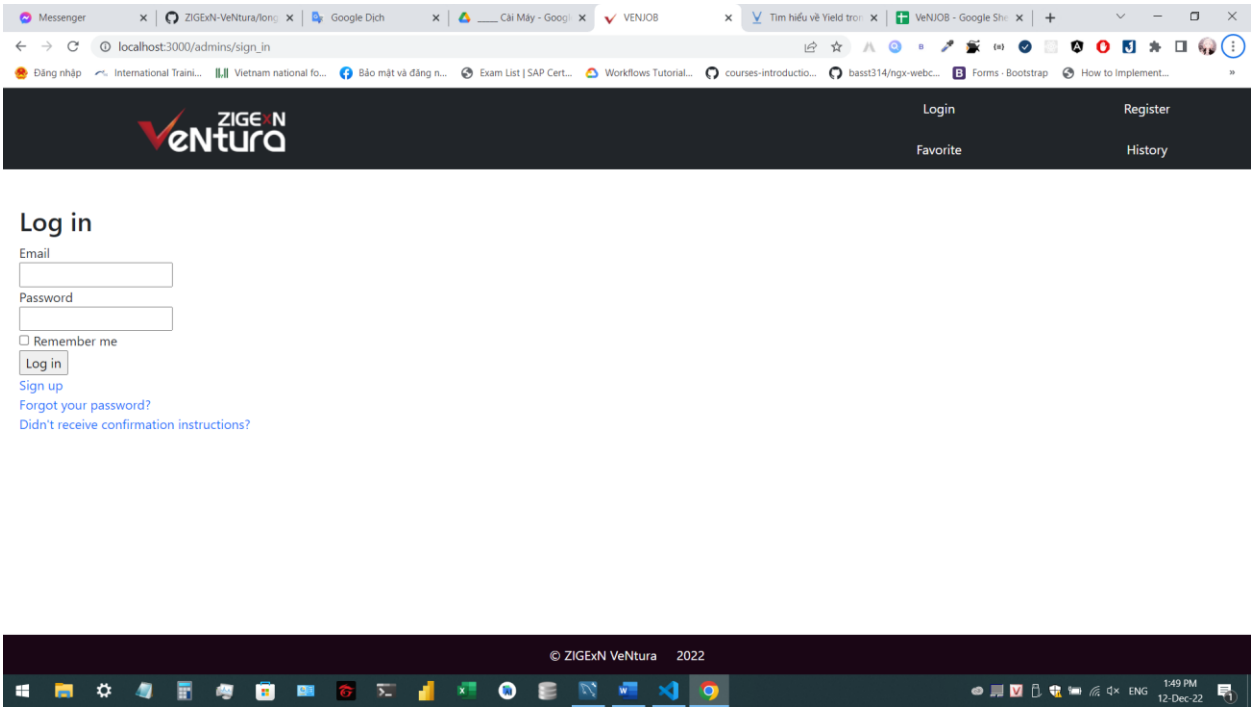


Figure 61: Admin login page

When the admin is successfully logged in, the page will navigate to the admin's own page.

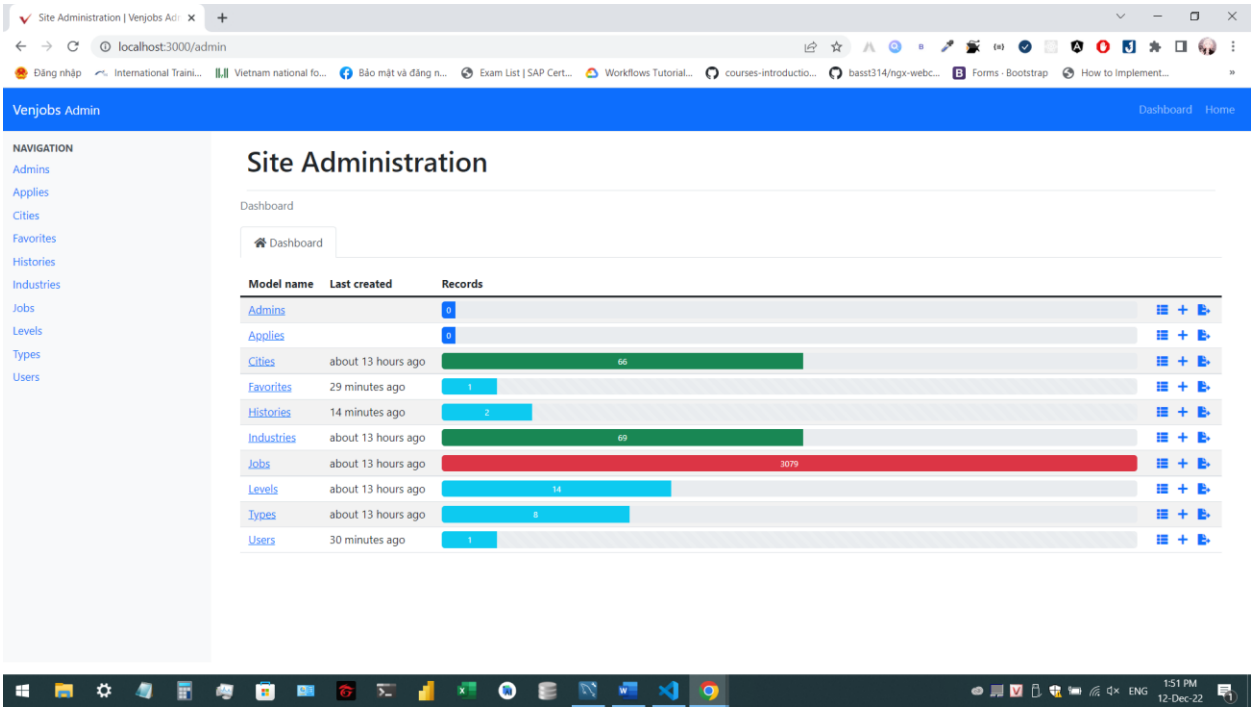


Figure 62: Admin page

This site is powered by the "rails-admin" gem. It will help the admin to customize the site's data at will. Since I have just started the admin section, it is still quite simple, there will be adjustments in the future.

6.7.6. User Account

6.7.6.1. Registration (Sign up) + Login (Sign in)

The functions of registration, login, forgot password... are supported by the "devise" gem. This page will support registration for users. When the user enters missing information and presses register, it will display a message asking for sufficient input (similar to the cases where the specified password does not match).

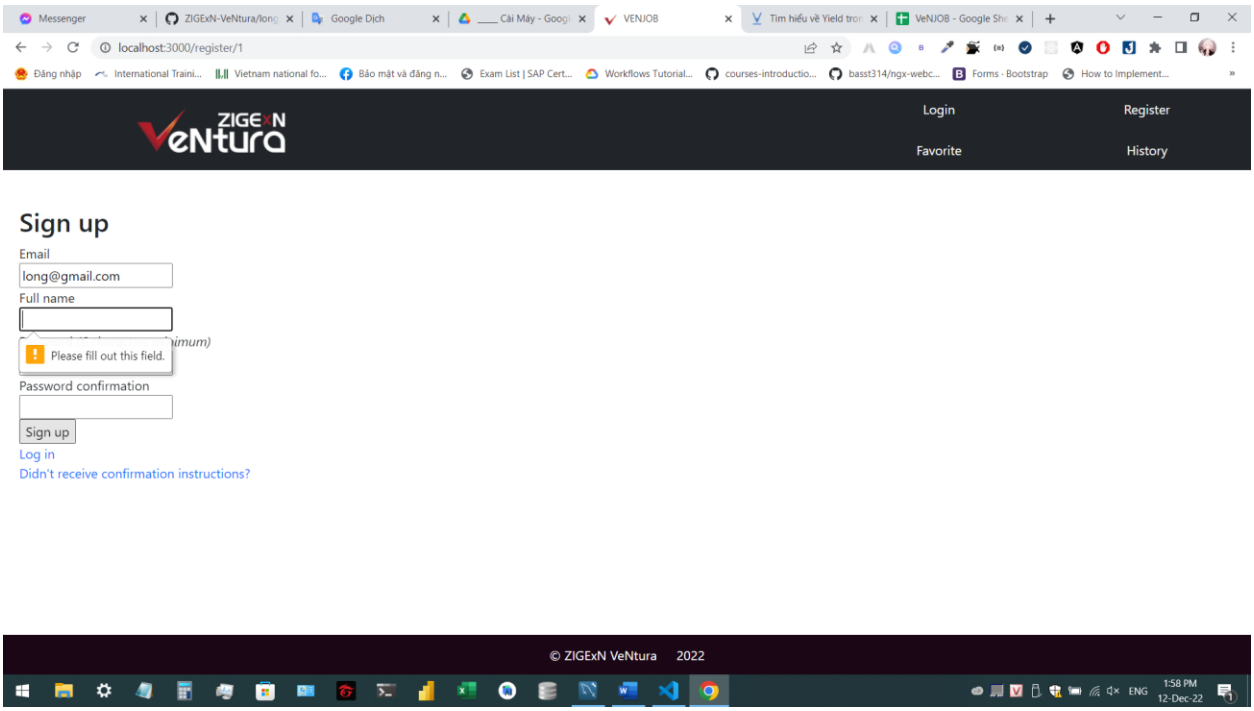


Figure 63: Sign up failed because of empty or wrong fields

Upon successful registration, the website will display a message to check email for users to click on the verification link. The email will be displayed in the console ("letter-opener" gem supports).

The system will send the verification link via email (in the console) like this:

```
Devise::Mailer#confirmation_instructions: processed outbo
und mail in 21.8ms
Delivered mail 6396d1c8b672e_431c12e443419d@DESKTOP-K7QIF
ER.mail (819.3ms)
Date: Mon, 12 Dec 2022 14:01:28 +0700
From: demoappvilike2@gmail.com
Reply-To: demoappvilike2@gmail.com
To: long2@gmail.com
Message-ID: <6396d1c8b672e_431c12e443419d@DESKTOP-K7QIFER
.mail>
Subject: Welcome To VeNJOB! Confirm Your Email
Mime-Version: 1.0
Content-Type: text/html;
 charset=UTF-8
Content-Transfer-Encoding: 7bit

<p>You're on your way long2@gmail.com!</p>

<p>Let's confirm your email address.</p>

<p>By clicking on the following link, you are confirming
your email address and agreeing to VeNJOB's Terms of Serv
ice.</p>

<p><a href="http://localhost:3000/users/confirmation?conf
irmation_token=B2JLCtbSb_oEFWw4xaWb">Confirm my account</
a></p>
```

Figure 64: Email confirmation displayed in console

When the link is clicked and the authentication is successful, the website will display the following message:

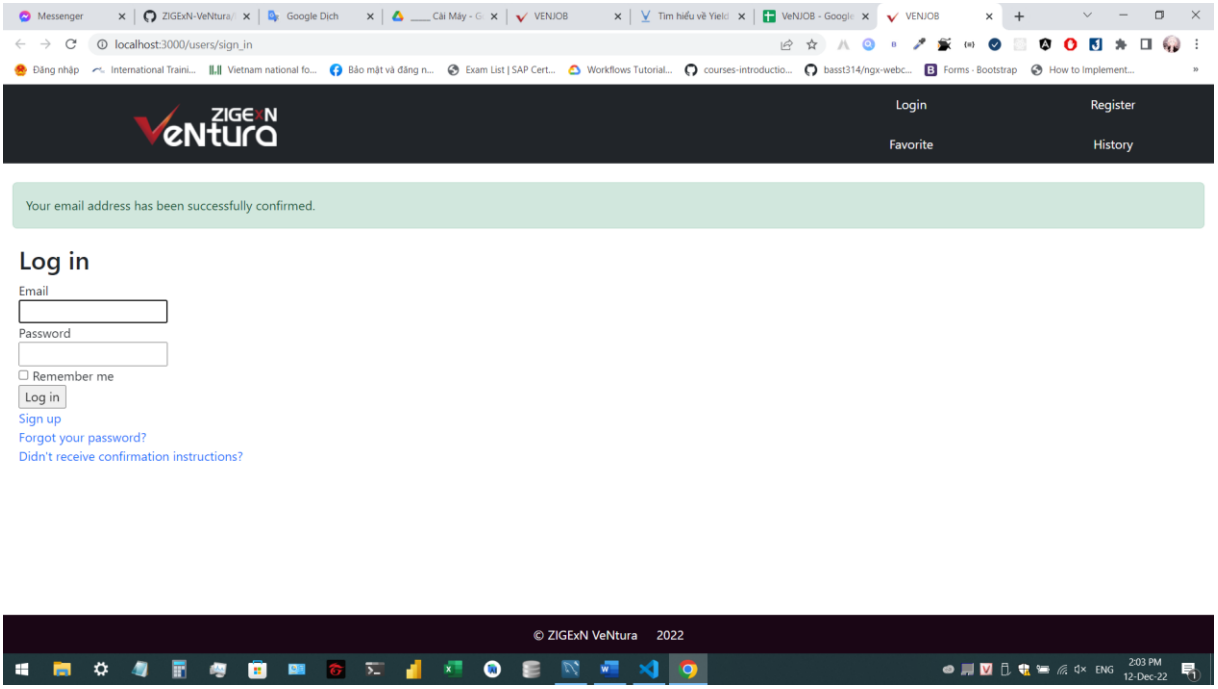


Figure 65: User authentication successful (Login page)

When the user enters the wrong information (does not match the data in the database), the website will display an alert warning the user.

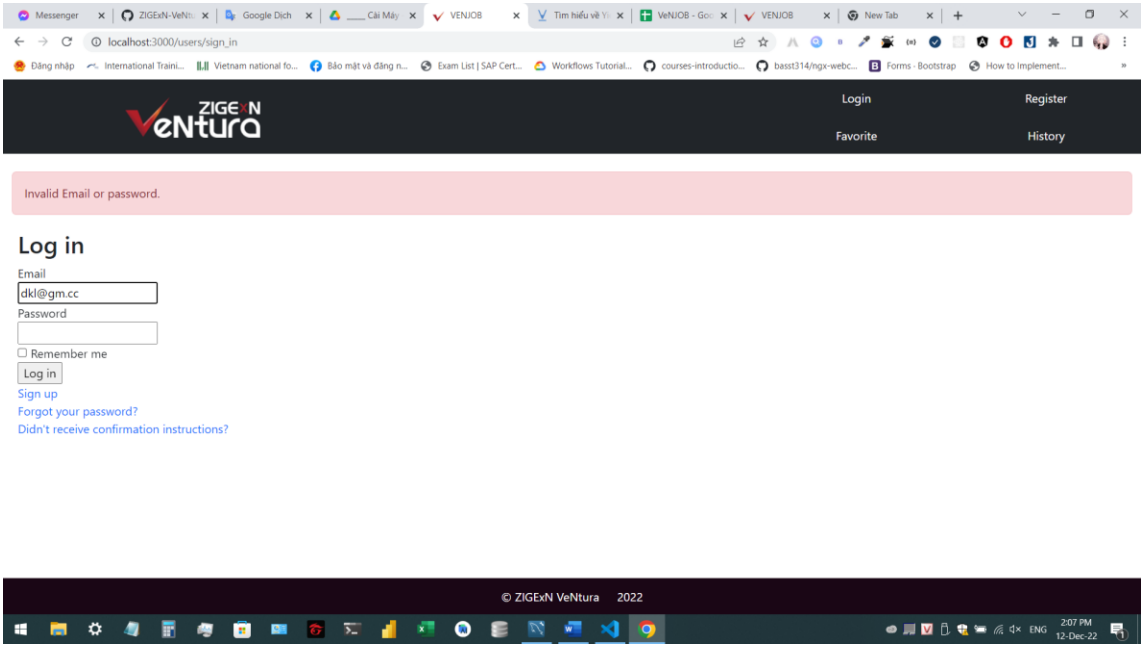


Figure 66: Login failed alert

And when the login is successful, it will switch to the homepage with a success message. The header will also change when the user is successfully logged in.

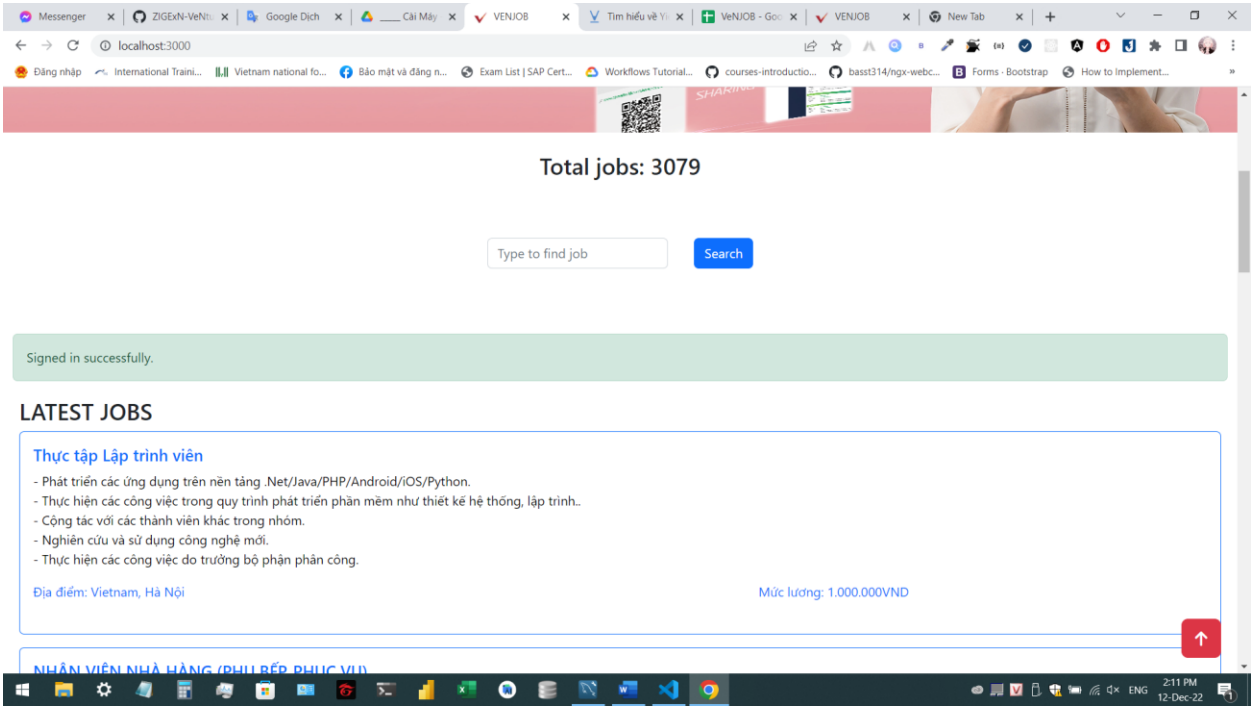


Figure 67: Login successfully

6.7.6.2. My Page

Once logged in, users can go to "My Page" to edit information as well as view the jobs they have applied for. Once logged in, users can go to "My Page" to edit information as well as view the jobs they have applied for. When the information entered is incorrect, or uploading a file that is more than 5MB in size or is not in the required file format, even in case of successful update, it will display messages similar to the following:

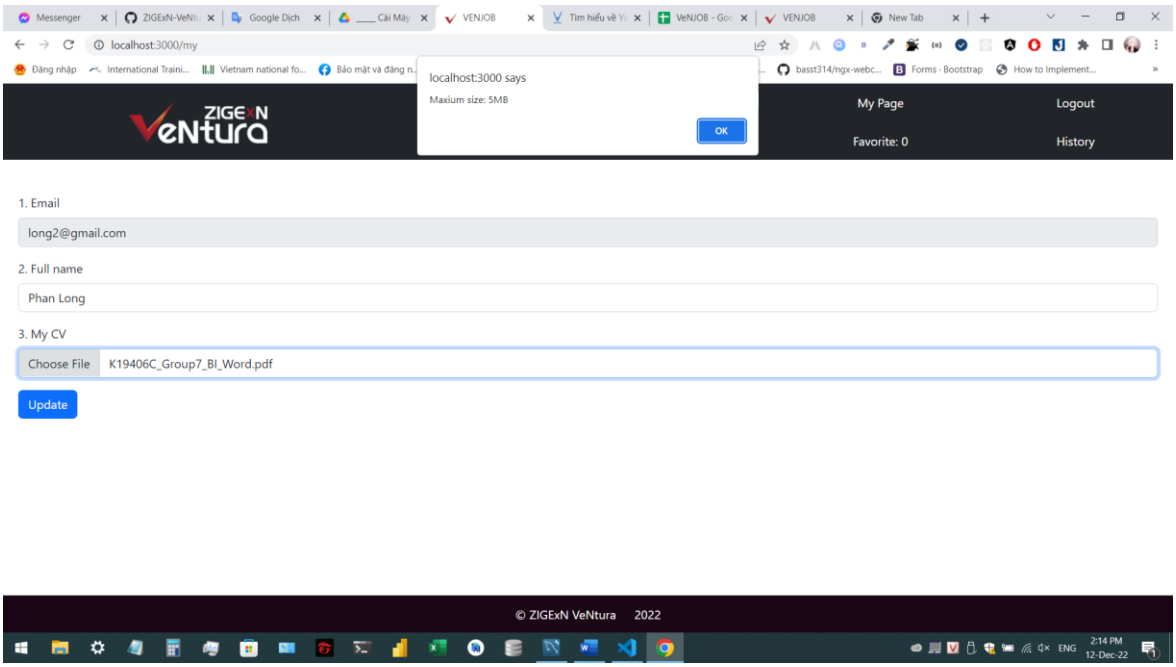


Figure 68: My Page section (error alert)

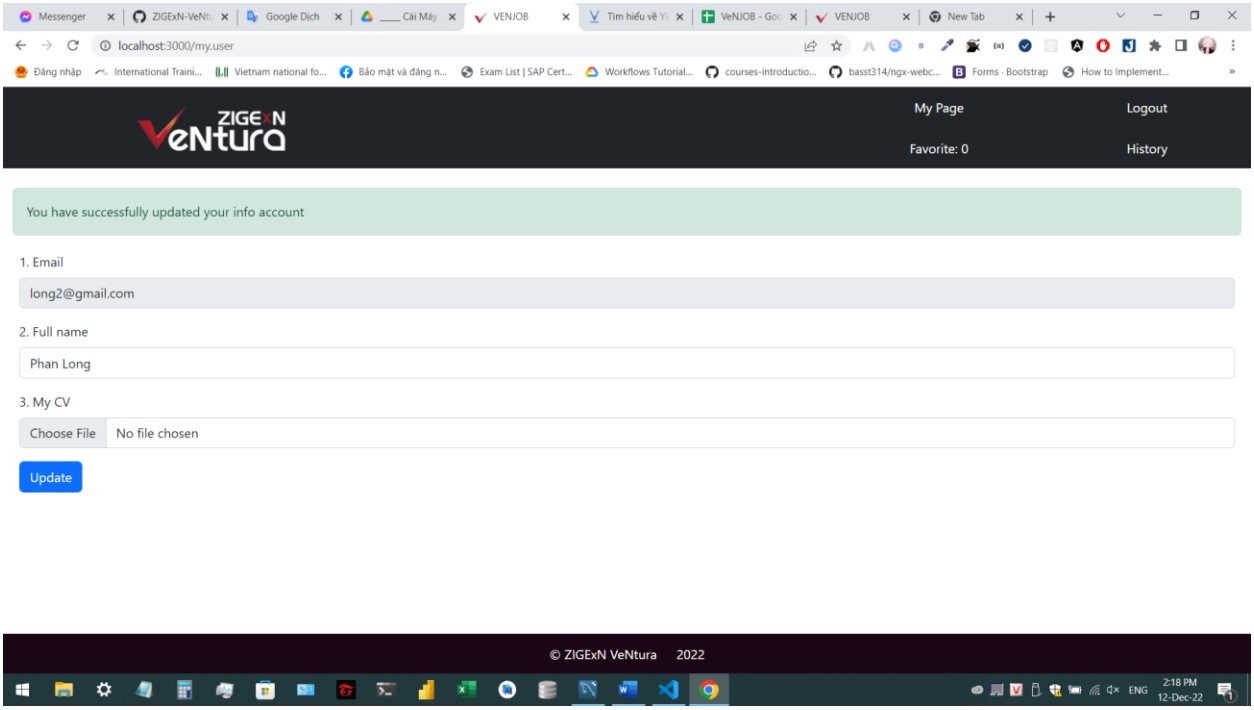


Figure 69: My Page section (Update successfully)

6.7.6.3. Forgot Password

When users forget their password, they can go to the "Forgot password" page to check and change.

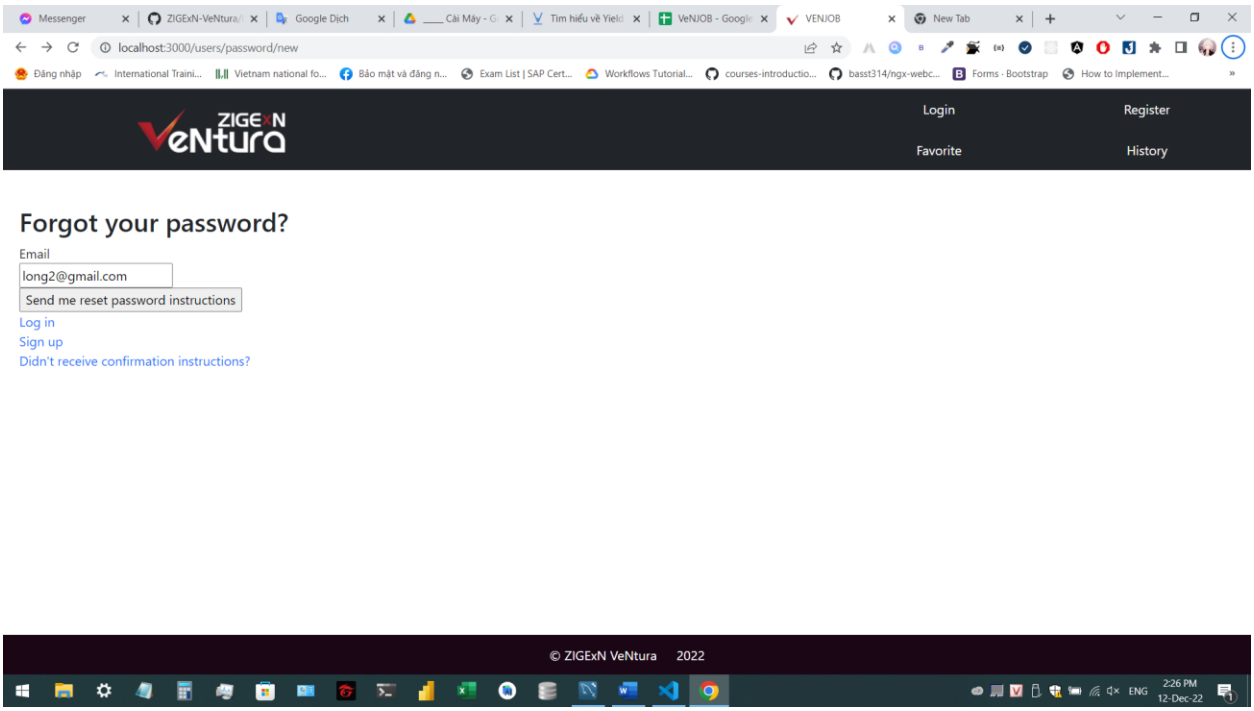


Figure 70: Forgot password form page

It will send a confirmation link to change password via console (similar to registration).

```
Delivered mail 6396d7f311855_431c12df43441a@DESKTOP-K7QIF
ER.mail (822.5ms)
Date: Mon, 12 Dec 2022 14:27:47 +0700
From: demoappvilike2@gmail.com
Reply-To: demoappvilike2@gmail.com
To: long2@gmail.com
Message-ID: <6396d7f311855_431c12df43441a@DESKTOP-K7QIFER
.mail>
Subject: Reset password instructions
Mime-Version: 1.0
Content-Type: text/html;
 charset=UTF-8
Content-Transfer-Encoding: 7bit

<p>Hello long2@gmail.com!</p>

<p>Someone has requested a link to change your password.
You can do this through the link below.</p>

<p><a href="http://localhost:3000/users/password/edit?res
et_password_token=Xbf_WBibau98VgNc3Q5v">Change my passwor
d</a></p>

<p>If you didn't request this, please ignore this email.<
/p>
<p>Your password won't change until you access the link a
bove and create a new one.</p>
```

Figure 71: Email about changing password in console

When clicking on the confirmation link to agree to change the password, the system will go to the "Change your password" page and the user can enter the new password and confirm the new password.

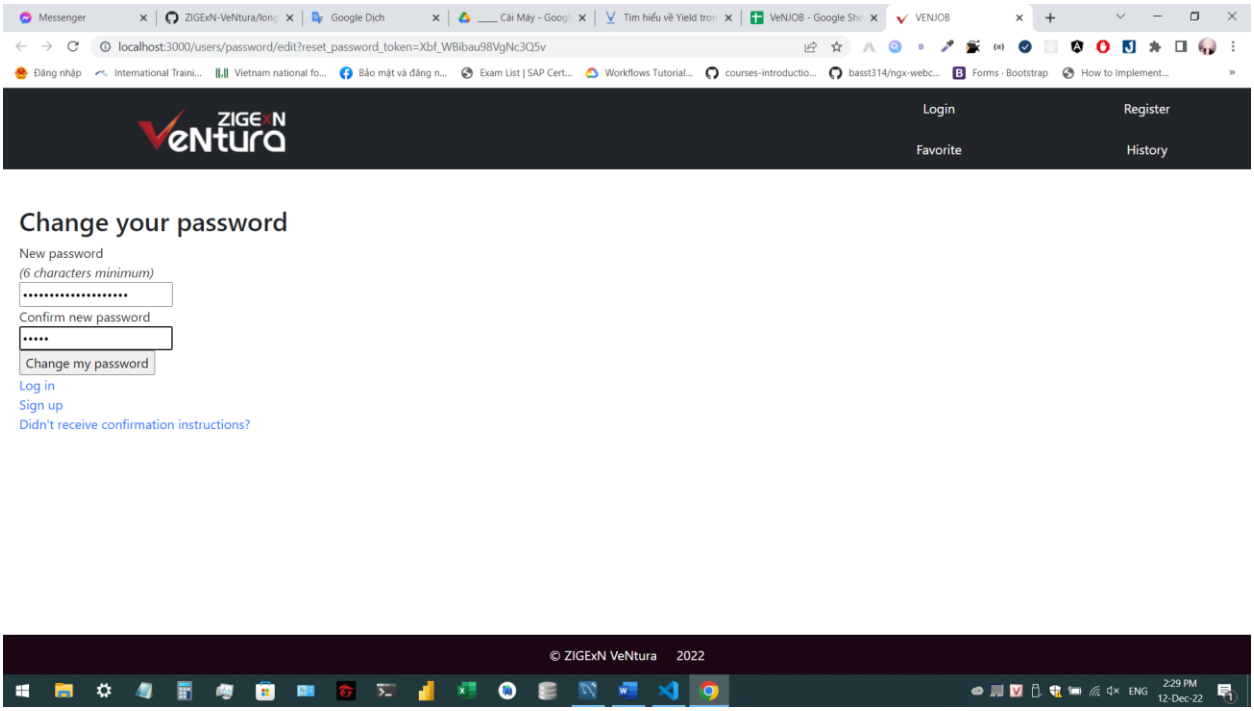


Figure 72: "Change your password" page

6.8. Manage project source code on Github

I use Github to manage the source code for my project, create branches for programming, and make pull requests for managers to review and merge branches. Some pictures of my project activities on Github:

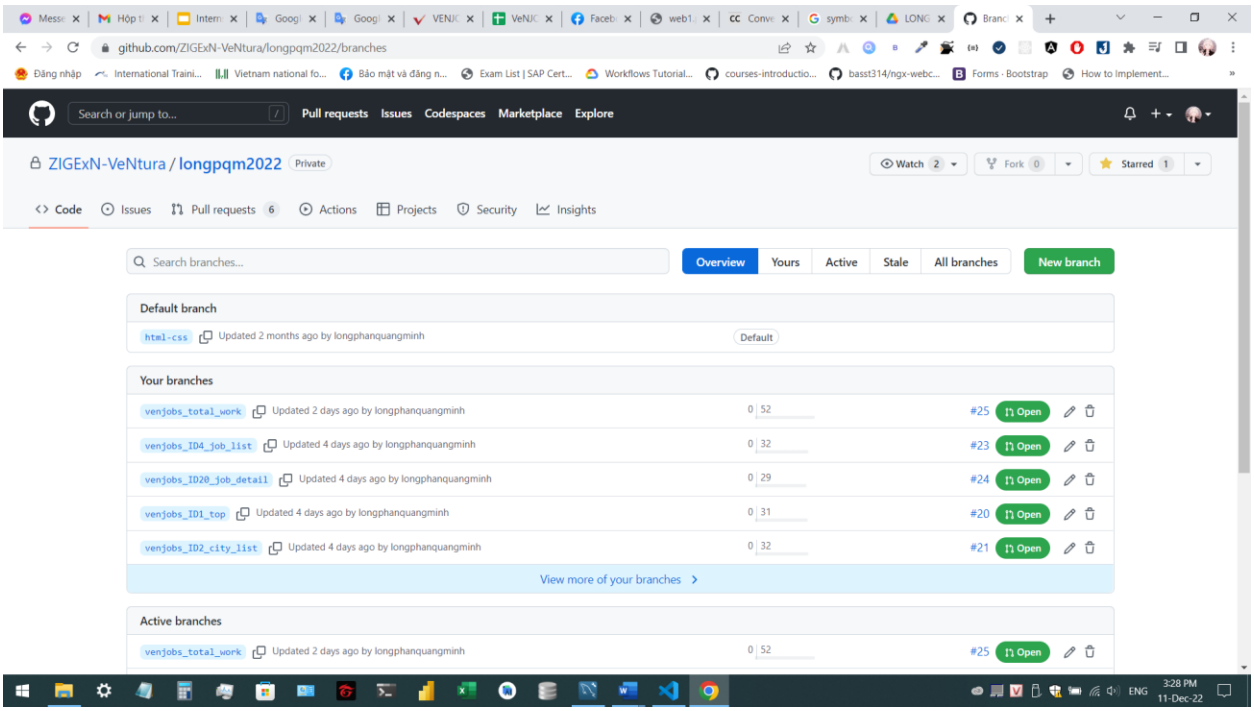


Figure 73: Project's branches

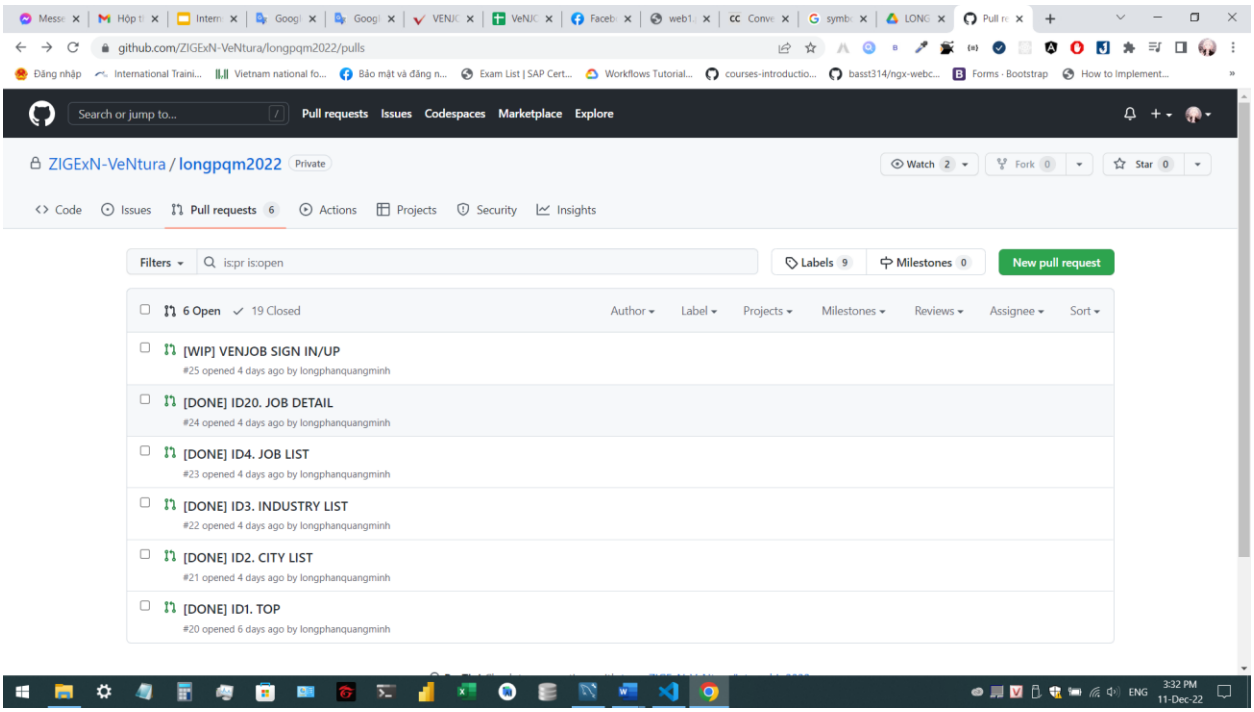


Figure 74: Project's pull requests

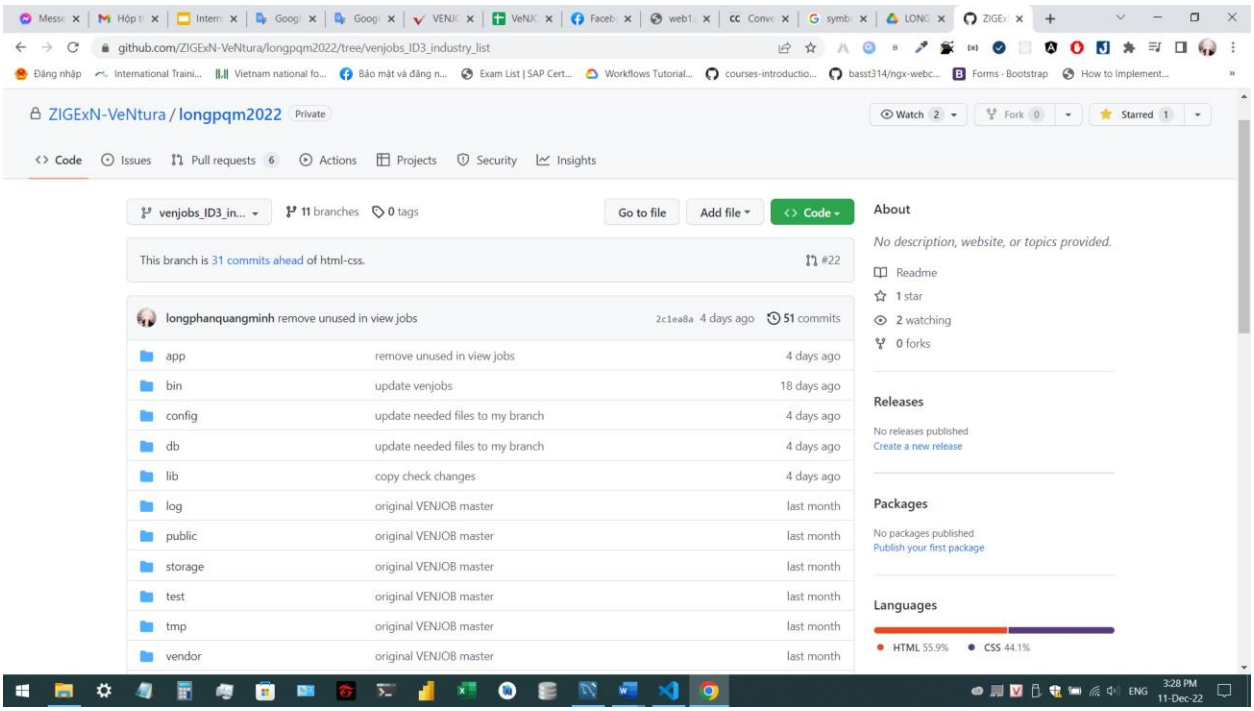


Figure 75: Source code in a branch of the project

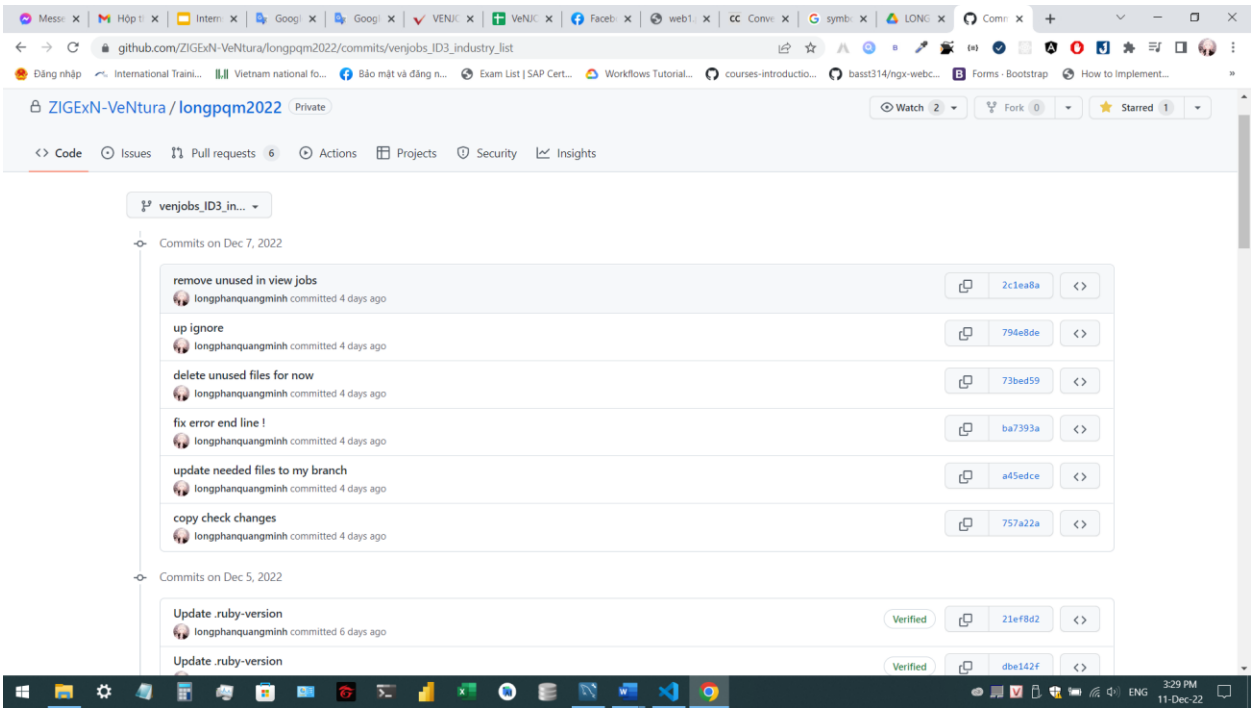


Figure 76: Project's commits

CHAPTER 7: CONCLUSION

7.1. Result

During the implementation of the thesis, I have found some results as follows:

- Successfully built a simple website to introduce and search for jobs.
- Understand the job search, and application process.
- Understand the scale and process of the MVC model.
- Know advanced knowledge of Ruby on Rails MVC.
- Friendly interface, easy to see, easy to use.
- The web application has helped users save time and effort to find a job easily. Easily manage user details. View detail information easily to save jobs and apply online quickly and effectively.

7.2. Limitations

First time trying to build a website with Ruby on Rails framework, I don't have a lot of really in-depth knowledge in this field, so this project is still limited:

- Haven't really designed a job search website application with a complete interface, there are still many shortcomings.
- Due to the limitation of the length of the report and the commitment to confidentiality with the company, some important information as well as documents, wireframes, detailed tasks, diagrams and functional requirements of the website are not presented in this report.
- There are still many functions that can be developed but not yet completed, will complete the rest until the end of the company's internship program (12/2022).

7.3. Future works

With many ideas still cherished, in the future I will try to complete the topic to achieve the goal of building a complete job search application. In which, there are the following goals:

- Especially developing the administrator functions, the functions that can interact well with users and some functions that have not been done, giving users the best experience with VENJOB.

- Integrate upgrade user account type.
- Post blogs, articles by categories on the website.
- Create CV online.
- And other future plans…

REFERENCES

[1] Ruby on Rails framework. Retrieved November 21st 2022, from https://rubyonrails.org.

[2] ZIGExN VeNtura, About ZIGExN VeNtura, https://zigexn.vn, Retrieved November 10th, 2022.

[3] Abbas, Mohammed. (2022). Design Web-Based System for Employee Registration: An Adaptive Application. 3. 14-21. 10.47310/iarjet.2022.v03i05.003.

[4] David Allen. (May 3rd, 2018). How to Build a Blog with Ruby on Rails. Retrieved November 10th, 2022, from Medium: https://deallen7.medium.com/ruby-on-rails-app-build-blog-3d9975a999ae .

[5] Nguyễn Hưng. (April 15th, 2022). Bootstrap là gì? Hướng dẫn cách dùng Bootstrap cơ bản. Retrieved November 10th, 2022, from Vietnix: https://vietnix.vn/bootstrap-la-gi

[6] Molefe, M. (2019), Ubuntu and Development, An African Conception of Development, in Africa Today, Vol. 66 No. 1, October 2019, https://www.researchgate.net/publication/337356649_Ubuntu_and_Development_An_African_Conception_of_Development, [22.9.2020].

[7] Tran Thi Tinh. (June 5th, 2018). Ruby_Cơ bản về Ruby & Ruby on Rails. Retrieved November 10th, 2022, from Viblo: https://viblo.asia/p/ruby-co-ban-ve-ruby-ruby-on-rails-L4x5xR9rZBM.

APPENDIX

Appendix 1. Source Code

Repository link: https://github.com/longphanquangminh/intern-project-DIY-K194060852 (The project folders are pushed to my own Github repository so everyone can see and understand my project making process).