

REPUBLIC OF TURKEY
HASAN KALYONCU UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT



SNACK JOBS
CO-OP FINAL REPORT

BY
Mustafa Alperen POLAT

AUGUST 2020

ABSTRACT

Today's, most of people can not find a fully employed job. However, they have to work somewhere temporarily until they find a formal job. And Employers sometimes need a daily employee in any sector. But the important fact here is it is only needed for one day.

This project plays a role as an intermediary between employers and employee. This app will allow to employers to find daily employees for their need. And employees have a chance of finding a daily job by choosing employers' register. My aim is to make to find a daily job easily to unemployed people and employers as soon as possible to easily uses.

In this project, I have firstly planned to written a web service which contains my business logic. And my service responds client side application. I have planned to create a web application. If a mobile application is needed in the future, my service is ready for this aim. Now I have developed a single page web application. Thus, an employee will be able to find a daily job by choosing job applications.

TABLE OF CONTENTS

ABSTRACT	2
TABLE OF CONTENTS	3
LIST OF TABLES	5
LIST OF FIGURES	5
LIST OF SYMBOLS	7
CHAPTER 1	8
INTRODUCTION	8
1.1 Purpose and Scope.....	8
1.2 Problem Statement.....	8
1.3 Solution Statement	8
1.4 Contribution.....	9
CHAPTER 2	10
IMPLEMENTATION	10
2.1 Back-End.....	10
2.1.1 Developing Database	12
2.1.2 End-points	14
2.2 Front End.....	15
2.2.1 Main Page.....	15
2.2.2 Sign Up.....	16
2.2.3 Sign In	17
2.2.4 Profile Page	18
2.2.5 Add Location Form	19
2.2.6 Use Current Location Form	20
2.2.7 Register an Application Page.....	21
2.2.8 The Nearest Works Page	22

2.2.9 Application Detail Page	23
2.2.10 My Done Applications Page	24
2.2.11 My Given Applications Page.....	25
2.2.12 Vote and Comment Form.....	26
2.2.13 Public Profile Page.....	27
2.2.14 Admin Panel	28
CHAPTER 3	29
DISCUSSION AND CONCLUSION	29
REFERENCES	30
APPENDIX	31

LIST OF TABLES

Table 1. DoneApplication Table.....	12
Table 2. GivenApplication Table.....	13
Table 3. Location Table	13

LIST OF FIGURES

Figure 1. A Bird's eye view from End Points	11
Figure 2. Main Page.....	15
Figure 3. Register Form.....	16
Figure 4. Login Form.....	17
Figure 5. Profile Page	18
Figure 6. Add Location Form	19
Figure 7. Use Current Location Form.....	20
Figure 8. Register a Job Form	21
Figure 9. The Nearest Works.....	22
Figure 10. Application Detail Page	23
Figure 11. My Done Applications Page	24
Figure 12. My Given Applications Page	25
Figure 13. Vote and Comment Form.....	26
Figure 14. Public Profile Page	27
Figure 15. Admin Panel.....	28
Figure 16. Database Schema	31
Figure 17. Web-service Code Map.....	31

Figure 18. Distance Calculation Codes 32

Figure 19. Transform Geolocation to Display Address and Display address to Geolocation
..... 33

LIST OF SYMBOLS

API	Application Programming Interface
EF	Entity Framework
JS	Javascript
SQL	Structured Query Language
NOSQL	not only SQL
Geolocation	Geographic Location
UI	User Interface
URL	Uniform Resource Loader

CHAPTER 1

INTRODUCTION

1.1 Purpose and Scope

The aim of this project is to bring employers and employees together. But the important fact here is it is only needed for one day. And it is location based application because the time of work is just for one day. In addition, the scope of this project is a study for employee and employers and it is possible for other people to benefit from this project.

1.2 Problem Statement

Most of people have become unemployed and they had hard time to find a job during the pandemic process [1]. There are some people in a difficult situation about finding a job. Especially, it is hard to find a job for students due to their school. Also, unqualified people have hard time to find job due to their unskillful things. There are many employment application, and agency. However, this situation was not the case for daily job. Considering this problem, I wanted to make a location based application for their need and favor.

1.3 Solution Statement

People cannot find a daily job by their location on the internet. Considering this problem, i found the following solution. Firstly, I have developed a database. Then, I have developed a restful web service. Eventually, I have developed a nice single page web application for employee and employers. In addition, it is a mobile-friendly website which means the project works properly on mobile browser that is easy to use for the users. Also, my application is maintainable for mobile application. Employer will register on app then job description will be published as name, title, location, date and price. Then user will access to system and will see the nearest job. Then it will apply for it and then employers choose one of them.

In this way, Employers will be able to publish application form by their needs. And employees will be able to apply a daily job. Eventually, employers will be able to choose an employee. They will not pay a money for this service except the internet connection. Finally, both of them meet their need.

1.4 Contribution

In a side of employees, contribution of my project is a chance of finding a near job temporarily. In a side of employers, contribution of my project is a chance of hiring an employee. Employees might catch an opportunity for formal job after they worked. In addition, I completely put at the disposal of people freely. This means that this application is free.


CHAPTER 2

IMPLEMENTATION

In this section, we describe what we have done for Snack Jobs web application including screen-shoots, detail information about each screen, flow and endpoints that we have used per each page individually.

2.1 Back-End

I used asp.net core, which is a web-development framework, in back end technology. I created a restful API which serves my client side. Also, I have developed a database on asp.net core thanks to its nice features. I preferred using multitier architecture that provides multilayer programming [2]. Thus, I avoid complexity of code and. It provides reusable, maintainable codes. Most of my end points handles database according to inputs.


Swagger
OpenAPI Specification

Select a definition
Our API

SNACK JOBS API v1/swagger.json

Authorize

Account

GET /api/Account

POST /api/Account/Register

POST /api/Account/AccessToken

POST /api/Account/RefreshToken

DoneApplication

POST /api/DoneApplication/MakeApplication

PATCH /api/DoneApplication/DenyApplication

PATCH /api/DoneApplication/AcceptApplication

PATCH /api/DoneApplication/CompleteApplication

GET /api/DoneApplication/GetEmployeeApplications

GivenApplication

POST /api/GivenApplication/RegisterApplication

GET /api/GivenApplication/GetApplications

GET /api/GivenApplication/GetApplicationDetail/{givenApplicationId}

GET /api/GivenApplication/GetPublicApplicationDetail/{givenApplicationId}

POST /api/GivenApplication/GetApplicationsByDistance

Location

PUT /api/Location/Create

User

GET /api/User/GetUserWithRole

GET /api/User/GetUserRole

GET /api/User/GetUserDetail

GET /api/User/GetPublicProfile/{userId}

GET /api/User/GetAllUsers

DELETE /api/User/DeleteUser/{userId}

Schemas

RegisterUserModel >

LoginModel >

RefreshTokenModel >

MakeApplicationModel >

DenyApplicationModel >

AcceptApplicationModel >

CompleteApplicationModel >

RegisterApplicationModel >

ApplicationsByDistance >

CreateLocationModel >

Figure 1. A Bird's eye view from End Points

2.1.1 Developing Database

I used MSSQL for the database of my project. And when I created my database, I want to use object relational mapping. For this purpose, there is a library called entity framework core [3]. I created my database model on classes. And it is automatically converting my classes to a database table.

I specified my needs according to my application. Firstly, there will be users for employee and employers and admin. They are stored in a one table. But I will recognize them according to their role this means that I need role table. Also, I will manage authentication with token. I need a refresh token table. It is necessary when token lost focus. Asp.net core identity contains all of my needs about users [4].

Then, employee and employers have some functions. Employers will register on app. This means that I need a given applications table which contains employers' registers. And I need a done applications for employees' applications. In addition, I store comment and vote columns in done applications table. These columns provides employers to vote and comment on the employee who got the job after the job is completed. Also, I need a location which store users' location. Location will be used for calculating distance between employees and employers.

In addition, I created a database for cities, counties, and neighborhoods by using Firebase. This means I stored addresses in a different database system, NOSQL.

Table 1. DoneApplication Table

Name	Type
UserId	uniqueidentifier
GivenApplicationId	uniqueidentifier
DoneApplicationType	int
Vote	real
Comment	nvarchar(MAX)

Table 2. GivenApplication Table

Name	Type
Id	uniqueidentifier
UserId	uniqueidentifier
Name	nvarchar(MAX)
Title	nvarchar(MAX)
IsActive	boolean
Price	decimal(18, 2)
Description	nvarchar(MAX)
TotalEmployee	int
StartDate	datetime2(7)
CreationDate	datetime2(7)

Table 3. Location Table

Name	Type
Id	uniqueidentifier
UserId	uniqueidentifier
Latitude	float
Longitude	float

2.1.2 End-points

I mainly have five main end points. And they contain sub-endpoints. They are designed according to database tables. And Most of them is protected by some authorizations rules.

First one is account. It serves authentication for users. This main end point serves access token, refresh token end register. For example, a user gets a token when he or she logs in the application. This token defines the user. This token contains some claims, so users successfully make a request if they have permissions.

Second one is user. It serves users operations such as, profile informations, edit informations, and delete user by admin.

Third one is location. It serves only putting a location for users.

Fourth one is givenapplication. It forms employer's given applications. Given applications are created, retrieving in here. Also, calculating distance between employee and employer is here.

Last one is doneapplication. The doneapplication handles employees' applications such as, applying an application. Also, there are some sub end-points which serve employers such as, deny, accept, complete, voting and comment on done applications.

2.2 Front End

I used vue.js, which is a JavaScript framework, in front end technology. It has many features for building a single page web application. It focuses UI. It gives a chance of developing component based web applications [5]. Also, I used third party JavaScript libraries in Vue.js to make developing easy and useful. One of them is Axios. Axios provides http client requests [6]. Other one is Vuex. It is a state management JavaScript library. It provides centralized store for whole components [7]. Other one is Vue Router. It provides URL management [8]. For warning, I used Vue Toasted which provides you to make warning easily [9]. Last one is Argon which contain built in components [10].

2.2.1 Main Page

This is the domain page of the application. When a user enters to the site, this page welcomes to the user. Also, user can search. If she or he wishes, she or he can register or login on this page.

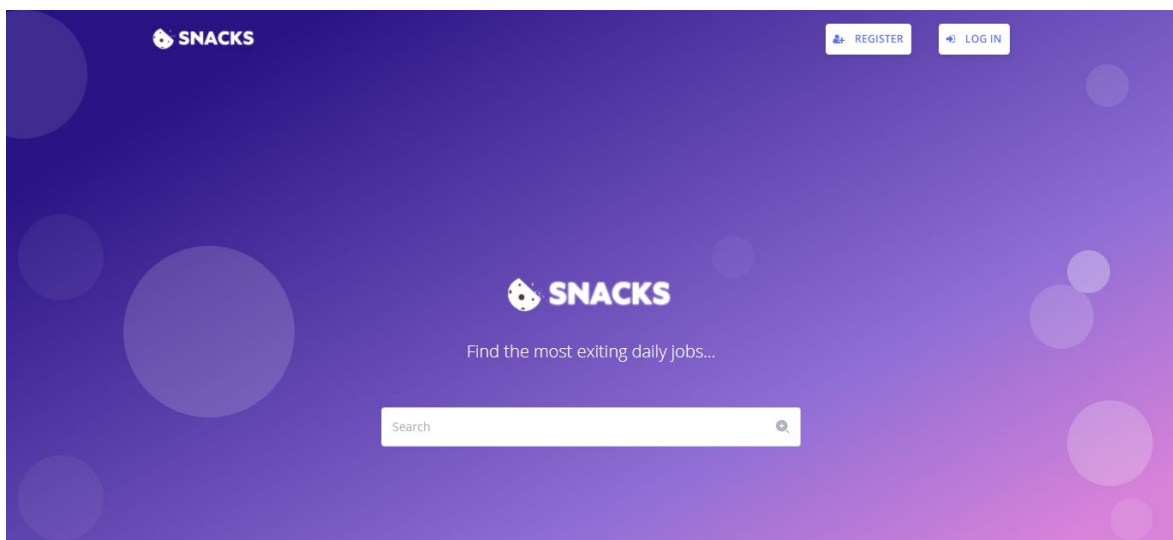


Figure 2. Main Page

2.2.2 Sign Up

In this page, user can sign up to the application with name, surname, username, email, phone number, gender, class, password and re-password. There are two options for class. These are employee and employer. If employer is selected, company name area will be appeared. Also, the system checks if fields are valid. If there is a mistake, warning will be shown on the right bottom.

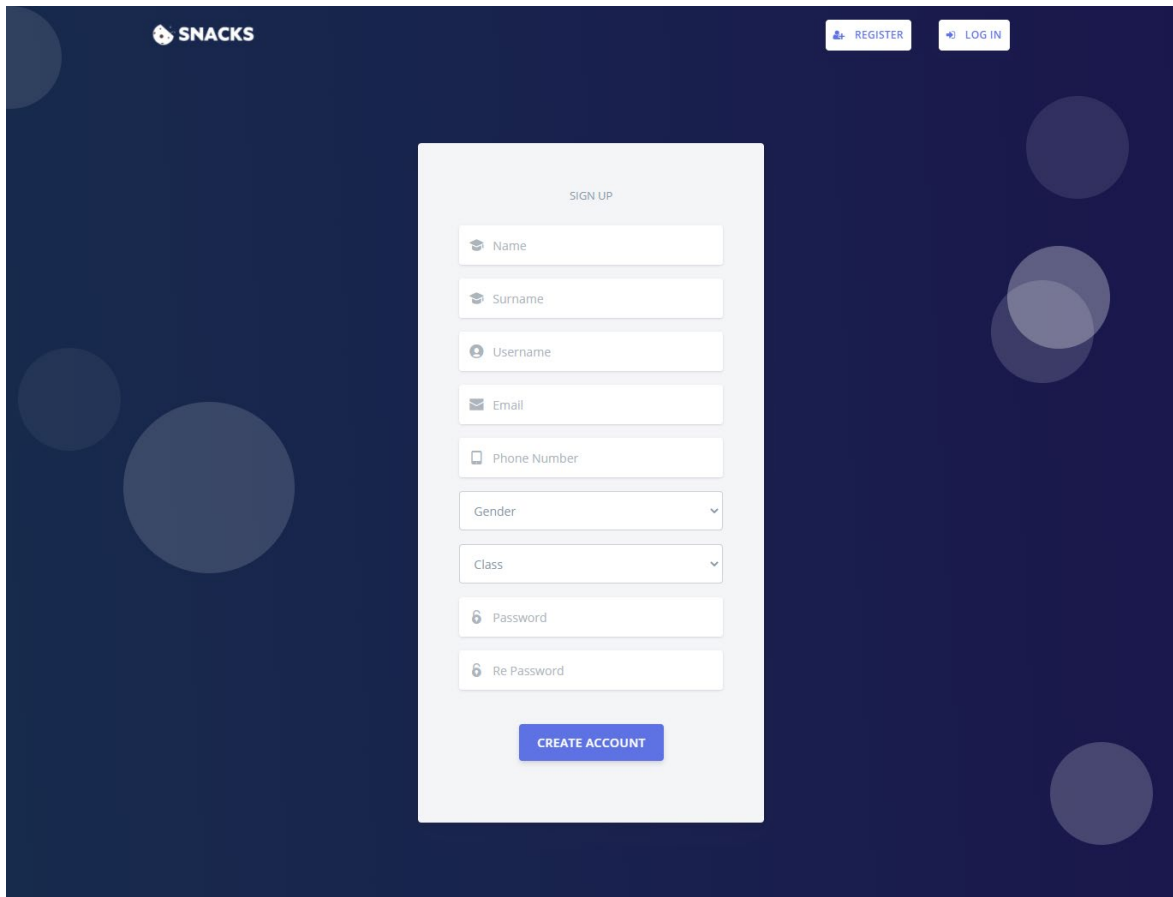
The image shows a web application interface for signing up. At the top left is the 'SNACKS' logo. At the top right are two buttons: 'REGISTER' and 'LOG IN'. The main content is a 'SIGN UP' form centered on a dark blue background with large, faint circular patterns. The form is a light gray rectangle containing several input fields: 'Name', 'Surname', 'Username', 'Email', 'Phone Number', 'Gender' (a dropdown menu), 'Class' (a dropdown menu), 'Password', and 'Re Password'. Each field has a small icon to its left. At the bottom of the form is a blue button labeled 'CREATE ACCOUNT'.

Figure 3. Register Form

2.2.3 Sign In

In this page, three types of user which are admin, employer and employee can sign in to the system with email and password. When a user login to the system, user will be redirected to main page. After that, they can make whatever their authority. And navigation bar will be changed. In addition, if a user fills the wrong email and password, error message will be shown.

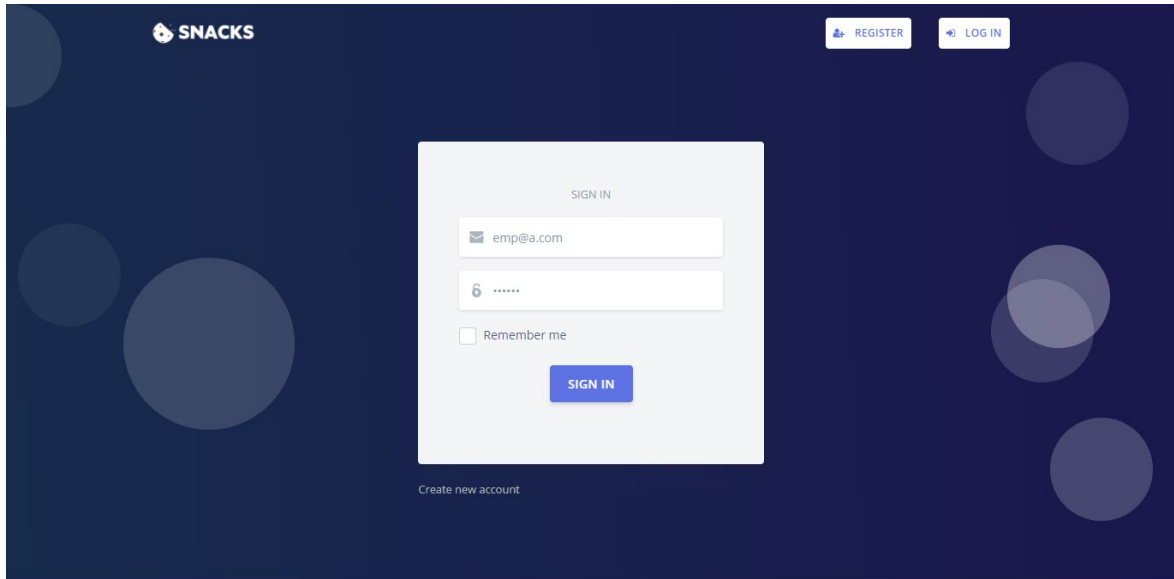
The image shows a web application interface for logging in. At the top left, there is a logo with a fork and knife icon followed by the text "SNACKS". At the top right, there are two buttons: "REGISTER" with a user icon and "LOG IN" with a key icon. In the center, there is a white rectangular box containing the login form. The form has the title "SIGN IN" at the top. Below the title, there is an email input field with a mail icon and the placeholder text "emp@a.com". Below the email field is a password input field with a lock icon and placeholder dots. Under the password field is a checkbox labeled "Remember me". At the bottom of the form is a blue button with the text "SIGN IN". Below the form box, there is a link that says "Create new account". The background of the page is dark blue with several large, semi-transparent circles of varying shades.

Figure 4. Login Form

2.2.4 Profile Page

In this page a user can see his detail and he can add a location. There are two options to add a location.

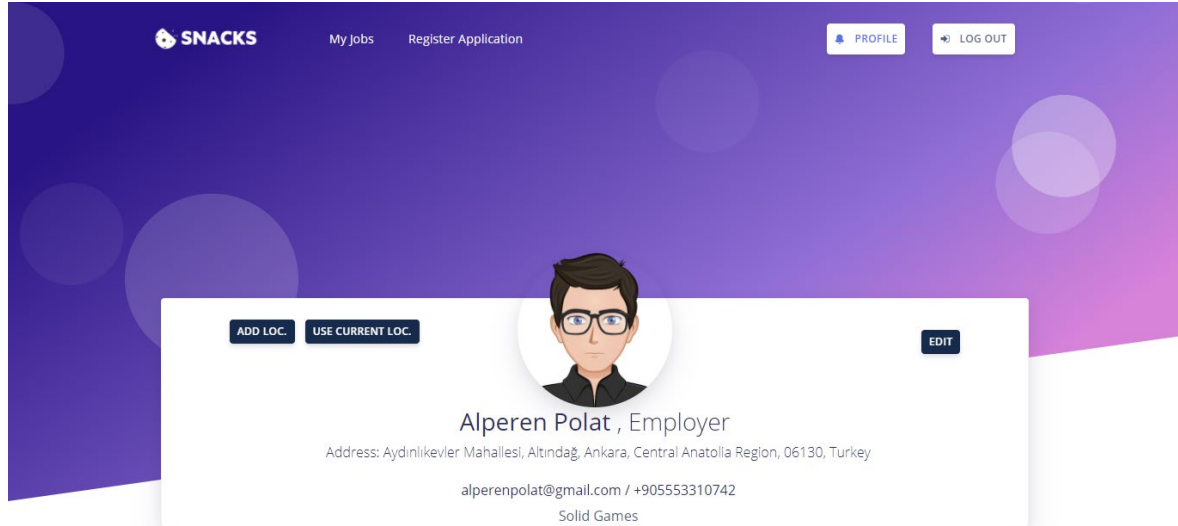


Figure 5. Profile Page

2.2.5 Add Location Form

To add a location, first option is called adding a location. This means that you will custom add your location by selecting city, county, and neighborhood in order. When you complete, you can put a location by clicking add button.

In the background, I stored users' locations as a coordinates. This means that I have to convert display address to coordinate. For this purpose, I used a converter API called Locationiq. It transforms display address to geocoding or opposite [11]. Thus, I can store users' address as a coordinate which is latitude and longitude. I would like to remind that coordinates are necessary to calculate distance between two points, origin and destination.

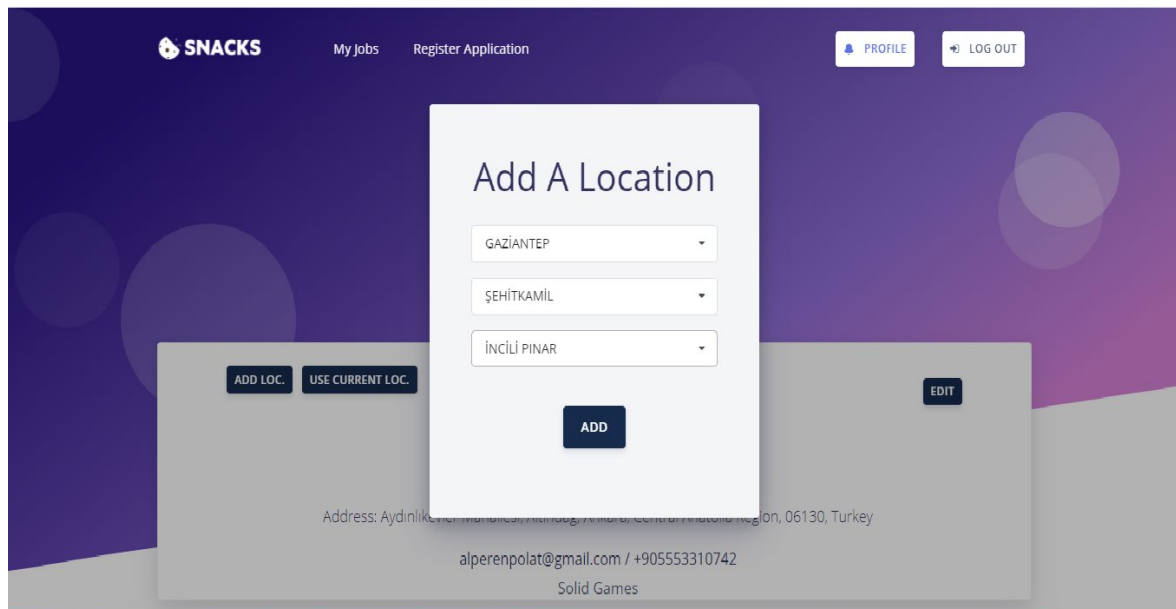
The image shows a web application interface for 'SNACKS'. At the top, there's a navigation bar with the logo, 'My Jobs', 'Register Application', and buttons for 'PROFILE' and 'LOG OUT'. The main content area features a modal form titled 'Add A Location'. This form has three dropdown menus for location selection: 'GAZIANTEP', 'ŞEHİTKAMİL', and 'İNCİLİ PINAR'. Below these is an 'ADD' button. In the background, a user profile card is visible, showing an 'ADD LOC.' button, a 'USE CURRENT LOC.' button, an 'EDIT' button, and contact information: 'Address: Aydınlikent, 06130, Turkey', 'alperenpolat@gmail.com / +905553310742', and 'Solid Games'.

Figure 6. Add Location Form

2.2.6 Use Current Location Form

To add a location, second option is called using current location. This option finds your location automatically by using Geolocation. The Geolocation api allows clients to find their current location [12]. But users need to allow geolocation to use this service when permission window is prompted. When you allow this permission window, you can put a location by clicking button called use this location.

In the background, this API, Geolocation, gives latitude, and longitude coordinates. Coordinates are unmeaning address for users. To show display address, I use locationiq API, as I mentioned below. Thus, I can show display address for users. Also, I can store users address as a latitude and longitude.

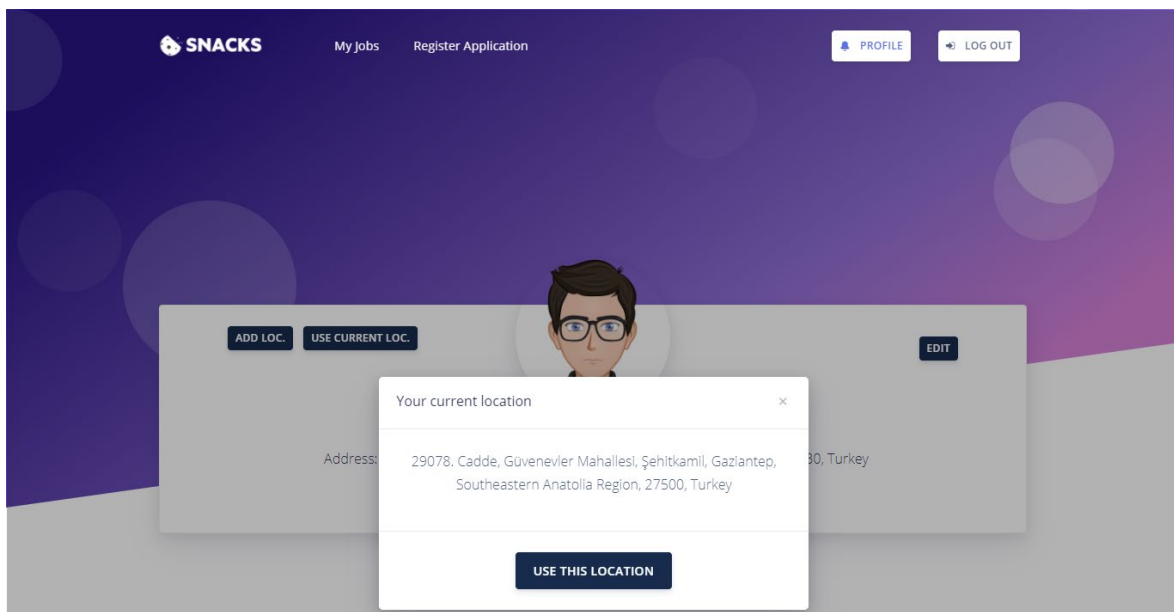


Figure 7. Use Current Location Form

2.2.7 Register an Application Page

In this page, employers can publish a job application by entering required fields. They can enter name, title, description, price, total personal, and start date. The location of the work is the user's location.

The screenshot displays the 'Register An Application' page. At the top, a purple navigation bar contains the 'SNACKS' logo, 'My Jobs', and 'Register Application' links. On the right side of the bar are 'PROFILE' and 'LOG OUT' buttons. Below the navigation bar, the page title 'Register An Application' is centered. The main content area features a 'Form' box with a light blue header. Inside the form, there are six input fields: 'Name', 'Title', 'Price', 'Total Employee', 'Starting Date', and 'Description'. Each field is preceded by a small icon. At the bottom of the form is a blue 'CREATE APPLICATION' button.

Figure 8. Register a Job Form

2.2.8 The Nearest Works Page

Employees can find the nearest works in this page. They can see the details of works such as, distance, price, name, title, and starting date. Also, there is sidebar for some settings. You can set maximum distance by using distance slider. Distance slider provide you to limit seek area for works. You can set min price by using price slider. Price slider provide you to limit minimum price for works. You can set date range by using range slider. Range slider provides you to limit between two dates. If user clicks the title of application, he is redirected to application detail page.

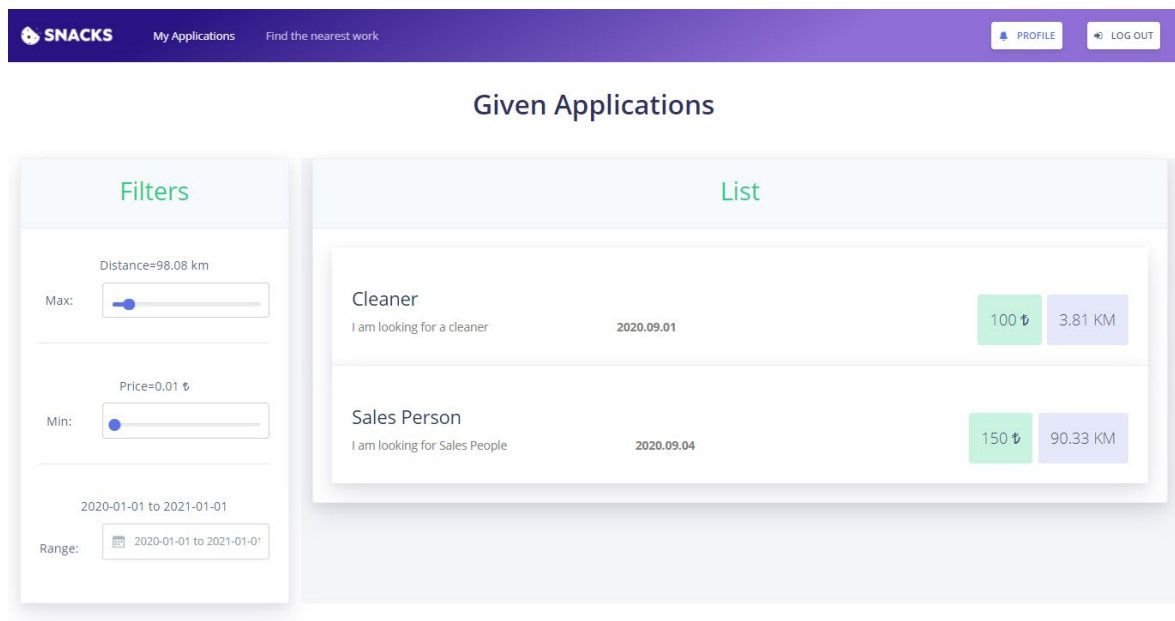


Figure 9. The Nearest Works

2.2.9 Application Detail Page

User can see the detail of application in this page. And if user wants to apply this work, user can apply by clicking button called apply on this page. If he clicks button called apply, he is redirected to my done applications page.

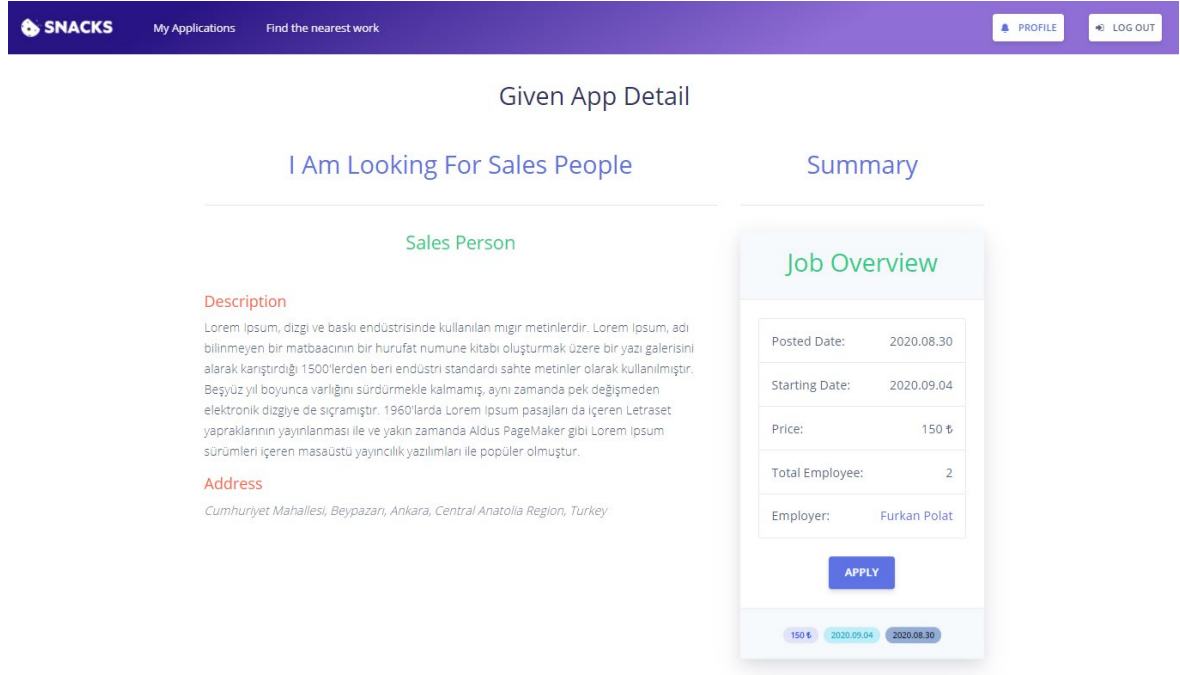


Figure 10. Application Detail Page



2.2.10 My Done Applications Page

Employee users can follow done applications thanks to this page. And there are some settings to filter. First filter called search by name provides you to search by application name. Second filter is called select status. You can select a status type and it brings applications which you selected option type. There are five option in select status. These are pending, denied, accepted, completed, and all. Detail page action (blue button) in the list provides you to go to the detail of the application. Remove application (red button) in the list provides you to remove the application.

SNACKS My Applications Find the nearest work PROFILE LOG OUT

My Done Applications

Search By Name select status

Application Name	Job Starting Date	Price	Status	Comment	Vote	Actions
Sales Person	2020.09.04	150 ₺	PENDING	NOT COMPLETED	NOT COMPLETED	 

Dear, my lecturers
Thank you for your efforts during my education life.

© 2020 Mustafa Alperen Polat & 171501583

Figure 11. My Done Applications Page

2.2.11 My Given Applications Page

Employer users can list and follow their applications thanks to this page. If employer clicks detail button on the card, he can see the done applications for his application. And he can take some actions. Employer user can deny done application by clicking red button (deny application). And employer user can accept the application by clicking green button (accept button). When user click the green button (accept application), blue button (complete application) will be activated. This means that employee finished the work. Thus, employer can vote and comment the application by clicking blue button (complete application). And employer user can go public profile page by clicking employee's name.

SNACKS My Jobs Register Application PROFILE LOG OUT

My Given Applications

List Content

Cleaner

I AM LOOKING FOR A CLEANER

I need a cleaner.

Total Employee = 1
Accepted Employee = 0
Total Application = 0

EDIT DETAIL

100 \$ 2020.09.01 - 2020.08.30




Full Name	Email	Phone Number	Status	Actions
mustafa polat	mustafapolat@gmail.com	+905553310710	PENDING	  

Figure 12. My Given Applications Page

2.2.12 Vote and Comment Form

If employee completes the work, employer can vote and comment thanks to this form. Employee can rate in the range of 0.25 and 5. And employee can leave a comment. Eventually, employer completes the application by clicking vote button. Automatically, the status type of the application will be changed to complete status.

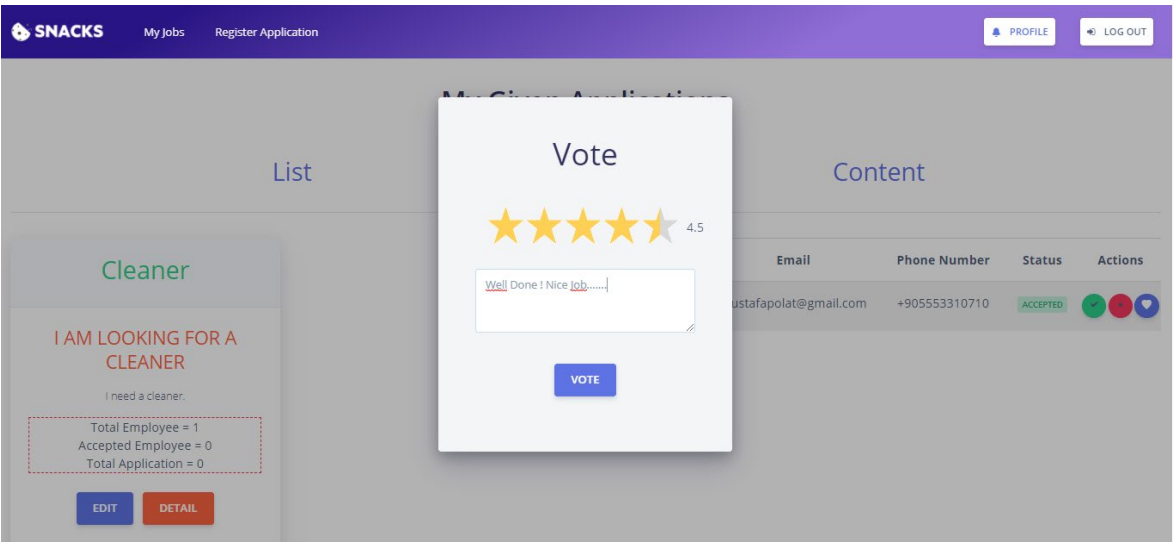


Figure 13. Vote and Comment Form

2.2.13 Public Profile Page

Employer wants to see an employee who apply his work before employer employs. Employer can view employee's old jobs. So, employer has an estimate about employee. This page is necessary for these requirements.

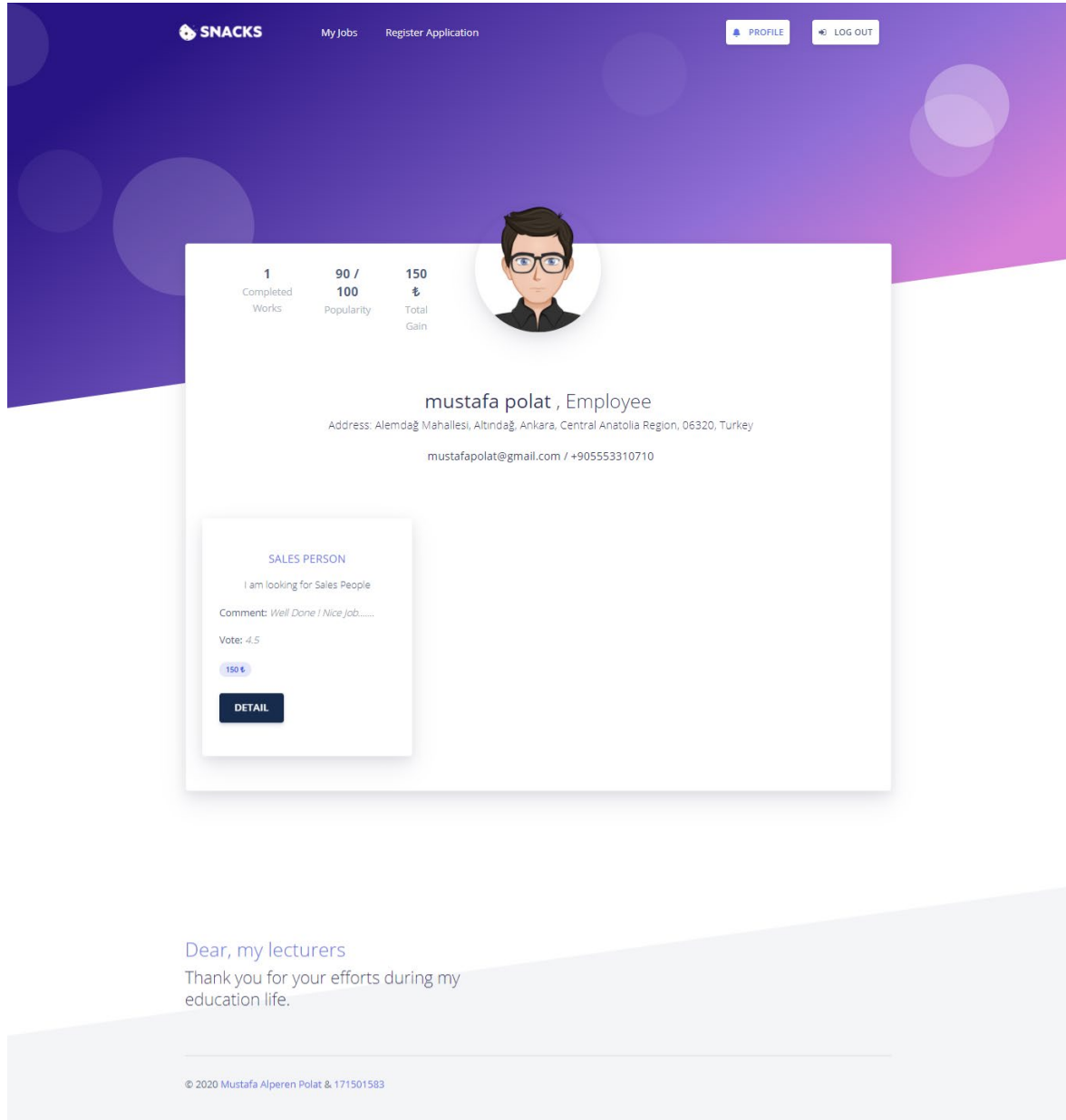


Figure 14. Public Profile Page

2.2.14 Admin Panel

Admin can list users. If he wants, he can delete a user by clicking delete action (red button).



The screenshot shows the Admin Panel interface. On the left is a purple sidebar with the 'SNACKS' logo and a 'Users' link. The main area is titled 'Users' and contains a table with three rows of user data. Each row has a red circular delete button in the 'Actions' column.




Full Name	Role	Email	Phone Number	Actions
Alperen Polat	Employer	alperenpolat@gmail.com	+905553310742	
mustafa polat	Employee	mustafapolat@gmail.com	+905553310710	
Furkan Polat	Employer	furkanpolat@gmail.com	+905553310720	

Figure 15. Admin Panel

CHAPTER 3

DISCUSSION AND CONCLUSION

As a conclusion, I believe that this project has helped me to think functional mindedness. Because I had to work front end and back end section at the same time. I had been learning javascript framework which I used in this project, Vue.js. On the other hand, I was applying what I learnt. Also, I benefit external web services. I combined them together. In addition, I learnt how to prepare real documentation for software projects thanks to our lecturers. With these software projects, I am available to prepare documentation.

REFERENCES

- [1]. Tetlow, G., Pope, T., & Dalton, G. Coronavirus and unemployment.
- [2].
https://en.wikipedia.org/wiki/Multitier_architecture#:~:text=In%20software%20engineering%2C%20multitier%20architecture,management%20functions%20are%20physically%20separated.
- [3]. <https://docs.microsoft.com/en-us/ef/core/> .
- [4].
[https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio.](https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio)
- [5]. <https://en.wikipedia.org/wiki/Vue.js>.
- [6]. <https://github.com/axios/axios>.
- [7]. <https://vuex.vuejs.org/>.
- [8]. <https://router.vuejs.org/>.
- [9]. <https://github.com/shakee93/vue-toasted>.
- [10]. <https://github.com/creativetimofficial/vue-argon-design-system>.
- [11]. https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API.
- [12]. <https://locationiq.com/>.

APPENDIX

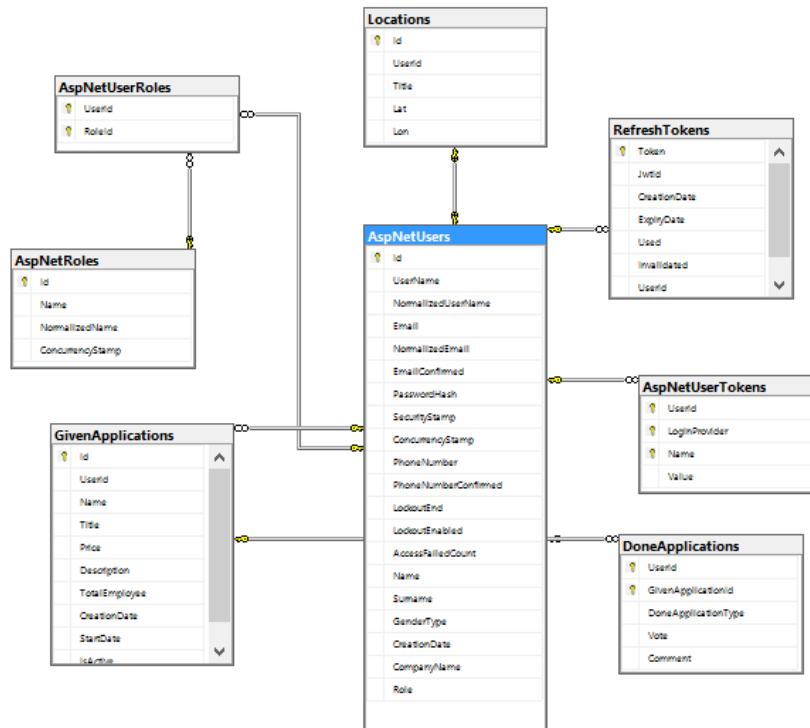


Figure 16. Database Schema

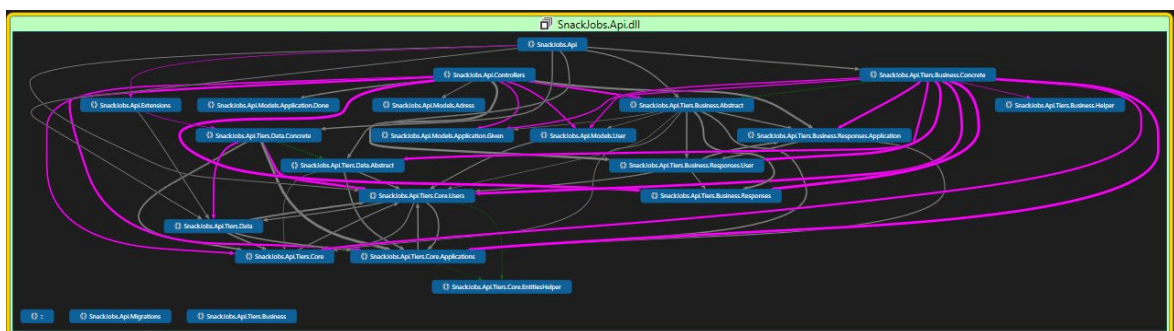


Figure 17. Web-service Code Map

```

public static class GFG
{
    4 references | 0 exceptions
    private static double toRadians(
        double angleIn18thofaDegree)
    {
        // Angle in 18th
        // of a degree
        return (angleIn18thofaDegree *
            Math.PI) / 180;
    }
    1 reference | 0 exceptions
    public static double distance(double lat1, double lon1, double lat2, double lon2)
    {
        // The math module contains
        // a function named toRadians
        // which converts from degrees
        // to radians.
        lon1 = toRadians(lon1);
        lon2 = toRadians(lon2);
        lat1 = toRadians(lat1);
        lat2 = toRadians(lat2);

        // Haversine formula
        double dlon = lon2 - lon1;
        double dlat = lat2 - lat1;
        double a = Math.Pow(Math.Sin(dlat / 2), 2) +
            Math.Cos(lat1) * Math.Cos(lat2) *
            Math.Pow(Math.Sin(dlon / 2), 2);

        double c = 2 * Math.Asin(Math.Sqrt(a));

        // Radius of earth in
        // kilometers. Use 3956
        // for miles
        double r = 6371;

        // calculate the result
        return (c * r);
    }
}

```

Figure 18. Distance Calculation Codes


```

import locationAxios from "../../custom_axios_locationiq";
const actions = [
  ...convertToCoordinateAsync({ commit, dispatch }, data) {
    let key = "e28676994a2f68";
    let q = data.neighborhood + ", " + data.county + "/" + data.city;
    let link = "/search.php?key=" + key + "&q=" + q + "&format=json";

    return locationAxios
      .get(link)
      .then(result => {
        console.log("convertToCoordinate works");
        return result.data[0];
      })
      .catch(err => {});
  },
  ...convertToAddressAsync({ commit }, data) {
    let key = "e28676994a2f68";
    let link =
      "/reverse.php?key=" +
      key + "&lat=" + data.lat + "&lon=" + data.lon + "&format=json";
    return locationAxios
      .get(link)
      .then(result => {
        console.log(result.data.display_name);
        return result.data;
      })
      .catch(err => {});
  },
];
export default {
  actions
};

```

Figure 19. Transform Geolocation to Display Address and Display address to Geolocation