|  |
| --- |
|  |
| Student Job Engine |
| An API platform for location-based jobs |

|  |
| --- |
| Janith Perera  5-3-2019 |

# Introduction

Student Job Engine is an API platform for any web or mobile interface which can provide a user interface and service to the user. The concept behind the platform is, with the growing number of exchange students from both international and local, the demand for student jobs get increased. As it is not a main objective for the university to facilitate part time jobs or temporary jobs for students, the problem of finding a job will get much more difficult for the student. The other problem is that the alien nature of the city, country and the society will make the problem more complicated. However, the student jobs in particular area play a huge role in student’s life, because most of the students require some financial support by their own. Not only the financial support, but also to adapt the environment, it is necessary to work with the surrounding society.

Some of the common methods of finding a job are, by newspaper, websites, university notices, course web (not in every case), and also by meetups. However, most convenient method would be finding a job via website or any related online material. However, the major problem with such a method is that, the student mostly likely will have no idea about the location, distance and time it takes to go there. The reason for this problem is, mostly these job portals are targeting the type of the job than the location.

The Student Job Engine platform is a prototype project, which propose a location-based job search and also job store for third party web or mobile portals. It uses google map services and RESTful API to act as a web service. The project is written in Golang programing language, with support of many Golang libraries and it runs as a backend service, which enables a port for the remote extensions to connect and communicate. It also includes an Object-relational Mapping platform to use variety of database types as some databases provides different features compare to another. This makes the platform more flexible in migrating to a new database system, and requires minimum alteration to the program.

The correct implementation of the project, with aid of a cloud platforms, proper database system and secure networking methods, can bring an impressive platform for multiple interfaces to serve many users concurrently. Other implementations can be also achieved by this project with minimum alteration to the structure of the platform and code, example, renting apartments, rooms, houses and also selling cars. The entire code base is released under open source license, where external contributors can join and improve the code, bring more features, enhance security and also change the targeted category. Main purpose of the making the source code available to the public is to collaborate with other ideas and bring a much more advanced platform for anyone who’s willing to use it. Even though it’s an open platform, further development can be led to in business perspective, with marketing strategies.

As it mentioned earlier, this is a prototype to bring a concept alive and it has many core implantations that require technical walk through to understand.

# Background of the Problem

The most common way of searching a Job is by googling or going to a famous Job web site. These kinds of methods are popular among the normal people, who live in a particular city, hence they know about what they are searching, where the location, etc. However, from the student perspective, the situation can be different. Let’s consider a student (Bob) comes to a city called Shire. His main purpose of being in the city is to study his bachelor’s degree in a famous university located in Shire. Upon arrival, he does his registration and residency process and starts his regular studies by visiting the university. In most of the cases students choose their hostel or room closer to the university. In this scenario, the public transport is not very convenient in the city and the university also not located in the center.

After a few months, Bob finds the expenses of that city is much higher than he estimated before he arrives and also, he wants to engage with the society outside the university environment. So, he decides to find a part time job, as he’s an active student. With the lack of experience in the city and the environment, he finds it’s difficult to land a job and also doesn’t find much resources to search on.

Not only there are no enough materials to search on, but also, he has new problems, they are that the location of the job, and how long it takes. As he’s continuing studies, he wants to manage the part time job and spend less time on the road for travel in between. The city itself has no convenient public transport and also private transport is unaffordable for a student in regular basis. By considering Bob’s situation, the new requirements arise in search of jobs, is that the search being location oriented. In most of the cases, part time jobs for students are very odd, and not in a specific industry or doesn’t require much skill set. Mostly they are training oriented and also contract basis. Because of these behaviors of the requirement, the targeted area would be the location, which should get high consideration from the student perspective.

The solution must be, a search performed with the coordinates of the currently location. Then based on those coordinates, with a special algorithm, the search should be performed and result must be displayed in the user interface.

The most job portals are not designed in this way, that they require exact current location of the user, rather they prefer a city. Other problem with popular job portals is they mainly target on more professional work, which are either fulltime, internships or long-term contract part time jobs. These are mainly provided by the companies, where they expect much documentation process to hire new employees and they always consider a fixed payment, and much more other conditions.

For the students like Bob, it is very convenient, if the University provides such a job search feature in any of their portal, where students require no other 3rd party website to depend on. It will make more trust worthy service for the students and also, they can get some help from University officials for further information. Although it sounds very soft implementation, the University might not be able to handle everything by their own, where they require an open platform to use as a service.

# PaaS (Platform as a Service)

Student Job Engine is completely a service-oriented platform, which runs as a backend service. In this section it talks about the main architecture of the platform, database layer and RESTful API layer. The platform is purely written in Go language, which is a statically typed, compiled programming language designed by Google. It is syntactically similar to C, but it includes memory safe, garbage collection, and supports for structural typing, which are not available in C, unless they are implemented within the project. As it is compiled programming language, the use of resources is minimum and the speed is higher than other competitive programing languages like Python and Java.

## Platform Architecture

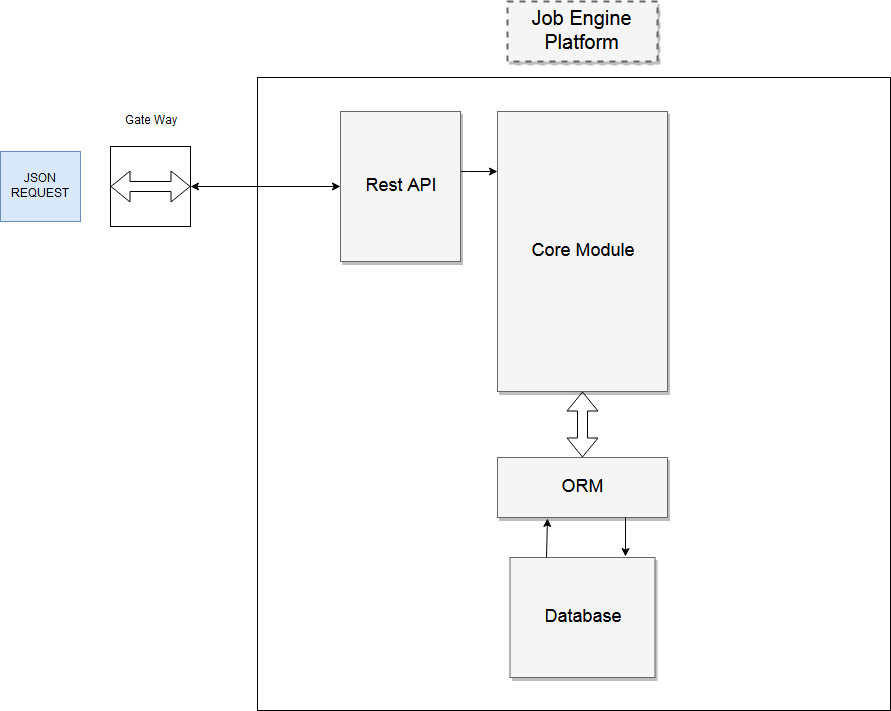


Figure 1Platform Architecture

## Rest API

This is the so-called front end of the platform, where the third-party applications can interact with the platform. It should be connected with a properly secured gateway to handle requests from outside. The Rest API is implemented with Gorilla Mux package, which is designed for Go environments. Gorilla Mux implements a request router and dispatcher for related incoming requests via web interface. The Mux stands for “HTTP request multiplexer”.

The implementation of Gorilla Mux is available as a part of core module, even though it is linked to the Rest API. The **main()** method initiates a Gorilla Mux router, and then handle the requests based on the path, and the method. The following implementation shows how it linked with user operations. Furthermore, in this implementation, it uses two Muxes for authentication required and not required operations. In later section, the authentication will be covered in details.

1. router\_noauth := mux.NewRouter()
2. router\_auth := mux.NewRouter()
4. router\_auth.HandleFunc("/users/all", ops.SelectAllUsers).Methods("GET")
5. router\_noauth.HandleFunc("/users/add", ops.InsertUser).Methods("POST")

In the example, it shows two methods, one is to get all users and the other is to add a user. With a close consideration, it can be seen that, the methods required to access these registered paths are **GET** and **POST** respectively. Also the two **mux** routers are dedicated for both authentication and non-authentication.

The REST API part of the platform consist of several subroutines for every operation that the platform has to do. Every subroutine require **http.Request** and **http.ResponseWriter** objects as arguments, where it can gather http request parameters and write back as a response. The entire reading, operation and writing back to the client is handled by within the subroutine.

The following code snippet shows an example of subroutine implementation. Each of these subroutines will be called in the **HandleFunc** of **Mux** router, with appropriate method.

1. func GetUserByFirstName(w http.ResponseWriter, r \*http.Request){
2. db := orm.Connect\_To\_Database()
3. params := mux.Vars(r)
4. first\_name := params["first\_name"]
5. var user\_array []orm.User
6. user\_array = orm.Search\_User\_From\_First\_Name(first\_name, db)
7. json.NewEncoder(w).Encode(user\_array)
8. }

## Authentication JWT