Shivam Singh **Zendesk Take Home Exercise**

Language/tools used: Golang, ReactJS, Docker, Postman client
Time Taken: 2-4 Hours.

Project Setup

1. Golang
   (a) Install Golang. (If using docker than can jump directly to Method 2/3 to run the Golang project)
   (b) Download the archive and extract it into **/usr/local**.
   (c) tar –C /usr/local –xzf go$VERSION.$OS-$ARCH.tar.gz
   (d) Add **/usr/local/go/bin** to the PATH environment variable by adding it into **$HOME/.profile**

   export PATH=$PATH:/usr/local/go/bin
   (e) Setup workspace by creating a directory **$HOME/go** and add it as path variable
   (f) **Export GOPATH=$HOME/go** in ~/.bash_profile file.
   (g) Create directory structure like **$GOPATH/src/github.com , $GOPATH/bin** and **$GOPATH/pkg** and **$GOPATH/src/golang.org** at $GOPATH
   (h) Create or clone a subdirectory in src folder with the username of github account (for ex: my username is napster11) like $GOPATH/src/github.com/napster11

   Ex**: cd $GOPATH/src && git clone https://github.com/napster11/zendesk.git**

2. React JS.
   (a) Install Node (using homebrew install node)
   (b) Clone the project
   git clone https://github.com/napster11/zenUI.git
   (c) Go to the project path and Do "npm install" to install the dependencies.

3. Run the program.

Method 1

(a) Go to project folder like src/github.com/napster11/zendesk and type
go get all && go install && go run main.go

(b) Launch rest client like postman and use the API documentation given below to test the service.

(c) go to zenUI project path and type “npm run” to run the project.

(d) Go to browser and type localhost:3000

Method 2

(a) Launch Docker Hub in Mac

(b) Type **docker pull shivam30/shivamzendeskservice**

(c) Now **docker run –t –p 8080:8080 {ImageID}**

(d) Test the service on Rest client using API documentation or try it using curl request provided below.

Method 3

(a) Go to the path of Dockerfile in project folder

(b) Type **docker build –t zendesk .**

(c) Type **docker run –t –p 8080:8080 zendesk**

(d) Test the service on Rest client.

Fetch Ticket List API

Name	**Quotes API**
Description	Provide the Ticket List of an account
Type	GET

Endpoint	/ticketList
Query Params	per_page and page
Example Endpoint	http://localhost:8080/ticketList?per_page=25&page=1
Success Response	{ "tickets": [{ "id": 1, "description": "Hi Shivam,\n\nEmails, chats, voicemails, and tweets are captured in Zendesk Support", "assignee_id": 360884808592, "brand_id": 360000149052, "created_at": "2018-02-12T21:26:48Z", "updated_at": "2018-02-12T21:26:49Z", "tags": ["sample", "support", "zendesk"], "submitter_id": 360884808592, "status": "open", "url": "https://singh782.zendesk.com/api/v2/tickets/1.json", "requester_id": 360891689372 }

	```
	    ]
	}
	```

Error Response	1. When null ticketList returned
	```
	{
	    "meta": {
	        "code": 400,
	        "msg": "No more Tickets Found"
	    }
	}
	```
	2. When Username and Password Mismatch
	```
	{
	    "meta": {
	        "code": 401,
	        "msg": "Something Went Wrong"
	    }
	}
	```
	3. When Auth is not provided in API
	```
	{
	    "meta": {
	        "code": 401,
	```

	"msg": "Username or Authtoken is missing" } }

Curl:

```
curl -X GET \
  'http://localhost:8080/ticketList?per_page=1&page=10' \
  -H 'Authorization: Basic {Auth_Token}
```

User AuthToken is using URL/token otherwise use password.

UI Flow:

1. Go to localhost:3000 and type username (email ID in my case) and password.
2. Instead of password can use auth_token but for this need to append/token in username.
3. I'm using first half of the email address to get the account name and appending it in the base URL like singh782@umn.edu is email address and API endpoint base for zendesk API is **https://singh782.zendesk.com/**

4. Returns List of tickets with Ticket ID, Ticket Status, Description, Created_At and Last Updated_At attributes.

Code Flow:

1. BootRouter function in zendeskService/zendeskRouter.go starts the service on port 8080 using mux library.
2. getTicketList in zendeskService/zendeskHandler.go is the handler function which will return the response.

3. Test cases are written in zendeskHandler_test.go

   a. First test case to check the happy path.
   b. Second test to check if ticket list is empty or not.
   c. Third test case to check if authentication is mandatory to access the API or not.