# Chong Woon Kiat
# 17-Oct-2020

# Prototype of a Dental Appointment System in Golang

# Dental Appointment System

This project is built on top of the last project in Go Advanced. Information on the implementation of data structure used can be found in the last report.

In this report, the focus will be on some key concepts of (i) Networking, (ii) Templates and (iii) Dependencies, as well as some error handling and concurrency mechanism incorporated.

## 1. Description of Features

The application is an appointment booking system for

(a) Patient to:

- View their appointment
- Add appointment
- Modify appointment

(b) Admin to:

- Modify appointment for a patient
- Remove appointment for a patient

Dummies accounts are pre-initiated in the server when the program is executed. Access the web server at URL http://localhost:5221

You may login as admin with username = “admin” and password = “password” or any dummy accounts (see instruction in section 7 below)

## 2. Networking

### 2.1. Net/http package

The net/http package is used to build a web server.
A listening socket which listen to the port 5221 is created.

```
func main() {
    http.HandleFunc("/", index)
    http.HandleFunc("/signup", signup)
    http.HandleFunc("/login", login)
    http.HandleFunc("/logout", logout)
    http.HandleFunc("/makeappointment", makeAppointment)
    http.HandleFunc("/adminedit", adminEdit)
    http.Handle("/favicon.ico", http.NotFoundHandler())
    http.ListenAndServe(":5221", nil)
}
```

It waits for and accept requests from client, and handle requests by reading the HTTP header to check if it uses POST method.
If a post method is received, the data in message body is read and pass to the handlers. In the net/http package, it can be done by simply calling the method:

```
func (r *Request) FormValue(key string) string
```

The form data are then process and return to the Client.

### 2.2. Session and cookie

Cookies are maintained by browser and web applications can access cookie information when users visit the corresponding website. Cookies have an expiry time, in this program, the expiry time is not set when a cookie is issued, this means that cookie will not be saved after the browser is closed.

An unique uuid is assigned to the cookie which serves as a session id. When the client access the webpage, if the session (cookie) if already exists, the server will return the same session to the client.

When a user logs out from the webpage, the server will change the expiry time of the cookie to -1, so that the cookie is destroyed, and the session is removed from the server side.

## 3. Templates

### 3.1. http/template package

http/template package is available in Go which helps handle templates. We make use of the ParseGlob function and Execute function to read and load templates to the webpage:

```
tpl = template.Must(template.ParseGlob("templates/*"))
```

```
tpl.ExecuteTemplate(res, "index.gohtml", pd)
```

where res is the response writer, and pd is data to be loaded to the page.

### 3.2. Loading data into template

In above line of code, pd (pageData) is a struct of information to be inserted to the templates. A new type, pageData is declared to serve as information storage for the handler functions.

```
type pageData struct {
    Title           string
    User            User
    Timeslot        Timeslot
    ErrorMsg        string
    DateList        []string
    DoctorList      []string
    SlotMap         map[string]string
    AvailabilityMap map[string]map[string][]bool
    UserMap         map[string]Timeslot
}
```

For example, in the makeappointment.gohtml file, it takes in the pd (pageData struct).
pd.User.HasBooking is loaded into the template and if the user has a booking, the date, slot and doctor information will be displayed:

```
{{template "header" .}}

<h1>Welcome to the online dental appointment portal.</h1>
{{if eq .User.HasBooking true}}
<h2>You have an appointment with {{.Timeslot.Doctor}} on {{.Timeslot.Date}}
slot {{.Timeslot.Slot}}.</h2>
<h2>Select a new timeslot you wish to change to:</h2>
{{end}}
```

## 4. Dependencies Management

Go dependency management system is used to make dependency information explicit and easier to manage.

A go.mod file is first created with command “go mod init” and in the file it shows all the dependencies in this module.
In this program, the package uuid and crypto are needed to generate session id for cookies as well as hashing users’ password:

```
module appointmentsystem

go 1.15

require (
    github.com/satori/go.uuid v1.2.0
    golang.org/x/crypto v0.0.0-20201016220609-9e8e0b390897
)
```

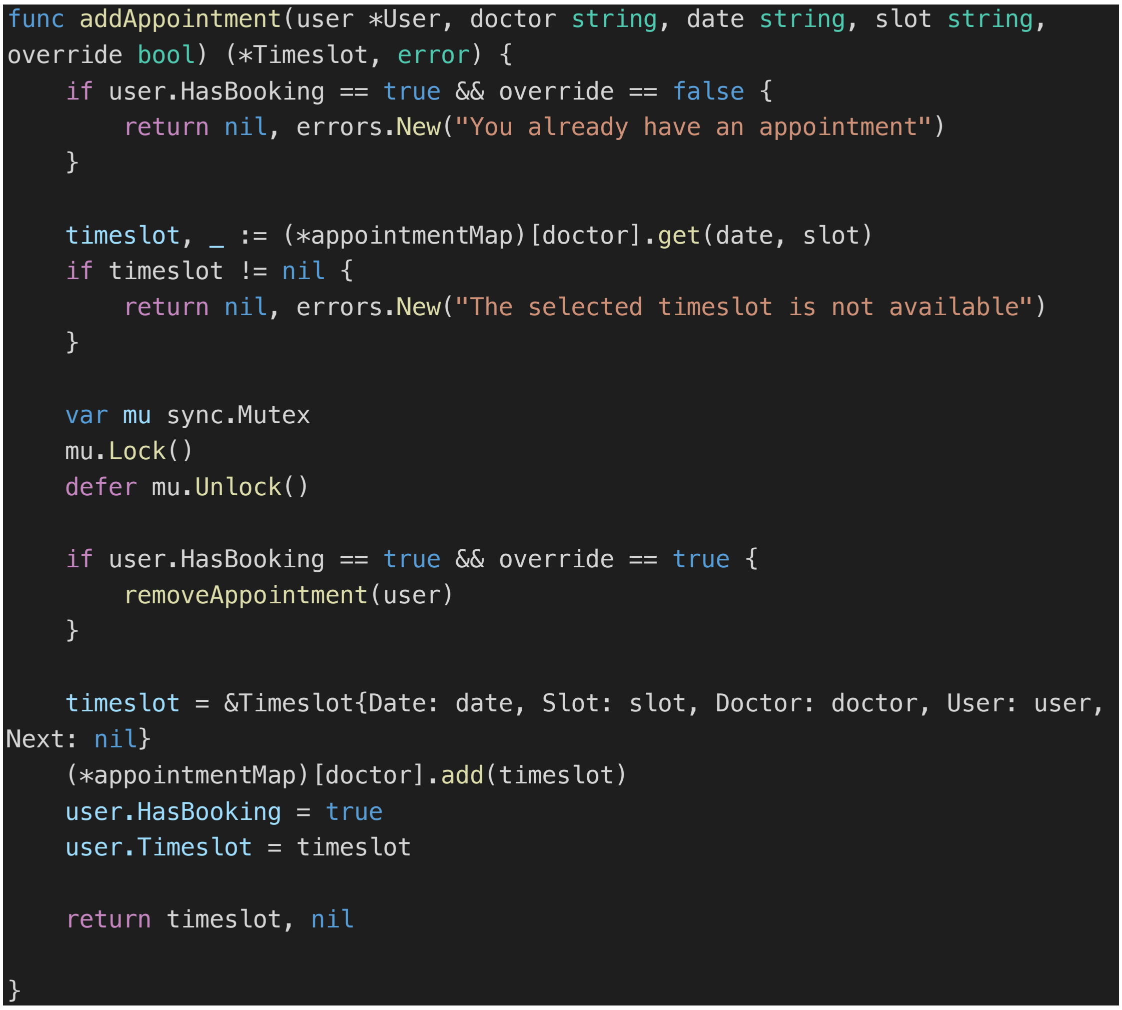
Note: There is a known bug where uuid.NewV4() function returning 2 variables but when shifting to modules, it took the latest stable release which had 1 variable
Read more from https://github.com/golang/go/issues/34280
Make sure to use v1.2.0 when running this program.

After running go build, go.sum gets created. The purpose of this file is to keep track of which codebase of the package is being used. It uses cryptographic hash to every dependency in the program:

```
github.com/satori/go.uuid v1.2.0
h1:0uYX9dsZ2yD7q2RtLRtPSdGDWzjeM3TbMJP9utgA0ww=
github.com/satori/go.uuid v1.2.0/go.mod
h1:dA0hQrYB0VpLJoorglMZABFdXlWrHn1NEOzdhQKdks0=
golang.org/x/crypto v0.0.0-20190308221718-c2843e01d9a2/go.mod
h1:djNgcEr1/C05ACkg1iLfiJU5Ep61QUkGW8qpdssI0+w=
golang.org/x/crypto v0.0.0-20201016220609-9e8e0b390897
h1:pLI5jrR7OSLijeIDcmRxNmw2api+jEfxLoykJVice/E=
golang.org/x/crypto v0.0.0-20201016220609-9e8e0b390897/go.mod
h1:LzIPMQfyMNhhGPhUkYOs5KpL4U8rLKemX1yGLhDgUto=
golang.org/x/net v0.0.0-20190404232315-eb5bcb51f2a3/go.mod
h1:t9HGtf8HONx5eT2rtn7q6eTqICYqUVnKs3thJo3Qplg=
golang.org/x/sys v0.0.0-20190215142949-d0b11bdaac8a/go.mod
h1:STP8DvDyc/dI5b8T5hshtkjS+E42TnysNCUPdjciGhY=
golang.org/x/sys v0.0.0-20190412213103-97732733099d/go.mod
h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/text v0.3.0/go.mod
h1:NqM8EUOU14njkJ3fqMW+pc6Ldnwhi/IjpwHt7yyuwOQ=
```

## 5. Concurrency

Mutex is used to lock writing access to the map containing link lists of all the appointments of the doctors, whenever a client submits a POST request to add or remove appointment.
This is to prevent two users booking the same time slot at the same time.



```
func addAppointment(user *User, doctor string, date string, slot string,
override bool) (*Timeslot, error) {
    if user.HasBooking == true && override == false {
        return nil, errors.New("You already have an appointment")
    }

    timeslot, _ := (*appointmentMap)[doctor].get(date, slot)
    if timeslot != nil {
        return nil, errors.New("The selected timeslot is not available")
    }

    var mu sync.Mutex
    mu.Lock()
    defer mu.Unlock()

    if user.HasBooking == true && override == true {
        removeAppointment(user)
    }

    timeslot = &Timeslot{Date: date, Slot: slot, Doctor: doctor, User: user,
Next: nil}
    (*appointmentMap)[doctor].add(timeslot)
    user.HasBooking = true
    user.Timeslot = timeslot

    return timeslot, nil

}
```

## 6. Error Handling

(a) The two main functions addAppointment and removeAppointment returns the time slots added/removed as well as the error, in the case when

- two users select the same time slot, but one submits faster than the another
- user clicks the back button after booking an appointment, resubmits with the same time slot
- user clicks the back button after removing an appointment, resubmits the form to remove again

```
func addAppointment(user *User, doctor string, date string, slot string,
override bool) (*Timeslot, error)
```

```
func removeAppointment(user *User) (*Timeslot, error)
```

These errors are then return to the Clients with http.Error function:

```
_, err := addAppointment(myUser, doctor, date, slot, true)

if err != nil {
       http.Error(res, err.Error(), http.StatusForbidden)
       return
}
```

(b) Error when users sign up with username that has been taken, log in with invalid username and password.

```
http.Error(res, "Username and/or password do not match",
http.StatusForbidden)
```

(c) Error when user submitting blank form.

```
http.Error(res, "Please select a timeslot", http.StatusForbidden)
```

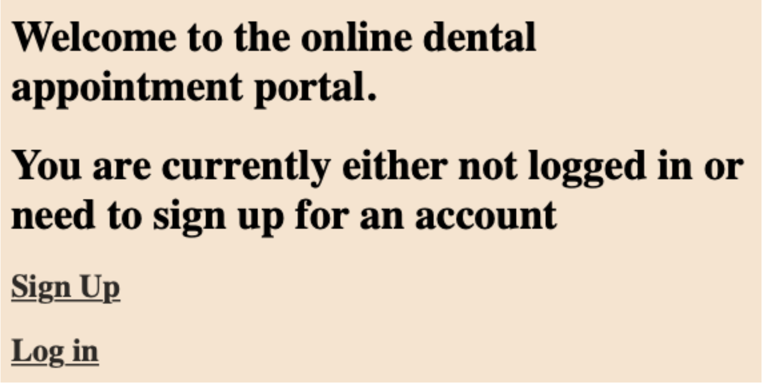
## 7. Instructions on running the application

Run the executable in the folder (one for MacOS, one for Windows)

OR

Run command “go build”. It downloads all the missing dependencies specified in go.mod. A new executable file will be created.

Go to http://localhost:5221



Welcome to the online dental appointment portal.

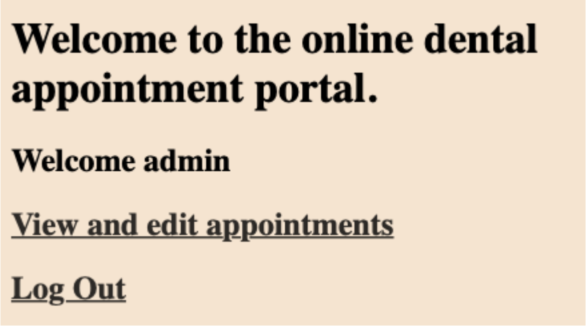
You are currently either not logged in or need to sign up for an account

Sign Up

Log in

### 7.1. Login as admin

Username: admin
Password: password



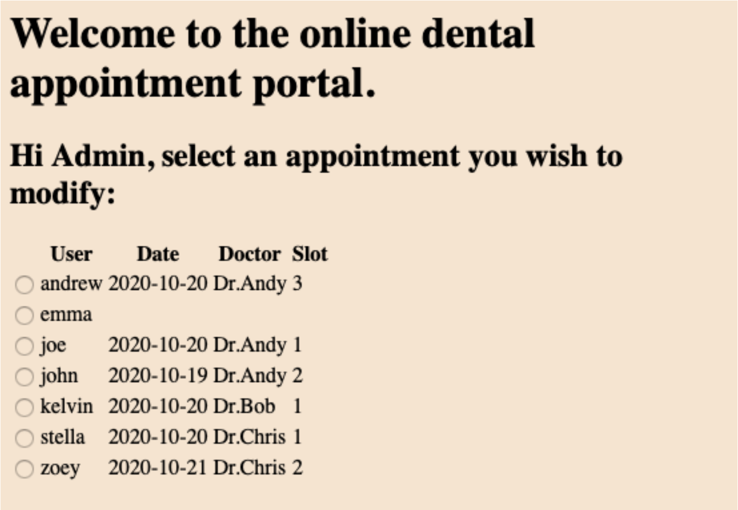
Welcome to the online dental appointment portal.

Welcome admin

View and edit appointments

Log Out

You may modify/ remove appointment for any dummy account.

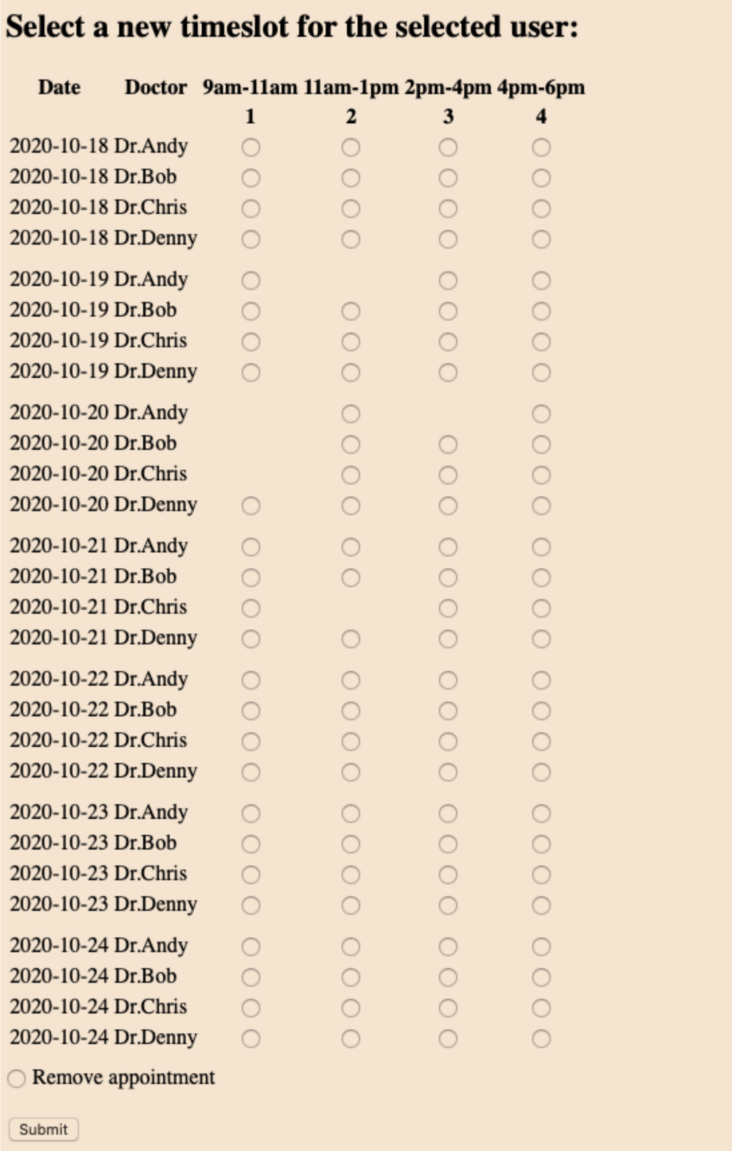


# Welcome to the online dental appointment portal.

## Hi Admin, select an appointment you wish to modify:

| | User | Date | Doctor | Slot |
|---|---|---|---|---|
| ○ | andrew | 2020-10-20 | Dr.Andy | 3 |
| ○ | emma | | | |
| ○ | joe | 2020-10-20 | Dr.Andy | 1 |
| ○ | john | 2020-10-19 | Dr.Andy | 2 |
| ○ | kelvin | 2020-10-20 | Dr.Bob | 1 |
| ○ | stella | 2020-10-20 | Dr.Chris | 1 |
| ○ | zoey | 2020-10-21 | Dr.Chris | 2 |

A radio button will only be available for time slots that are available.



## Select a new timeslot for the selected user:

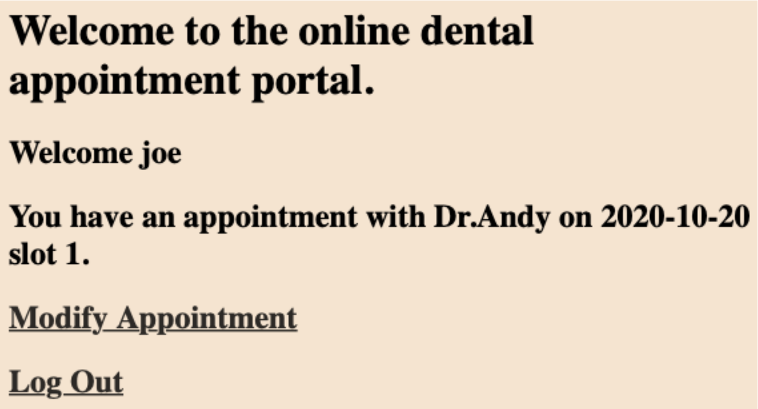
| Date | Doctor | 9am-11am | 11am-1pm | 2pm-4pm | 4pm-6pm |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 |
| 2020-10-18 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-18 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-18 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-18 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-19 | Dr.Andy | ○ | | ○ | ○ |
| 2020-10-19 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-19 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-19 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-20 | Dr.Andy | | ○ | | ○ |
| 2020-10-20 | Dr.Bob | | ○ | ○ | ○ |
| 2020-10-20 | Dr.Chris | | ○ | ○ | ○ |
| 2020-10-20 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-21 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-21 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-21 | Dr.Chris | ○ | | ○ | ○ |
| 2020-10-21 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Denny | ○ | ○ | ○ | ○ |

○ Remove appointment

Click submit to modify/remove the appointment
Logout to end the session (cookie will be destroyed)

## 7.2. Login as dummy user

Username: andrew/emma/joe/john/kelvin/zoey/stella
Password: password

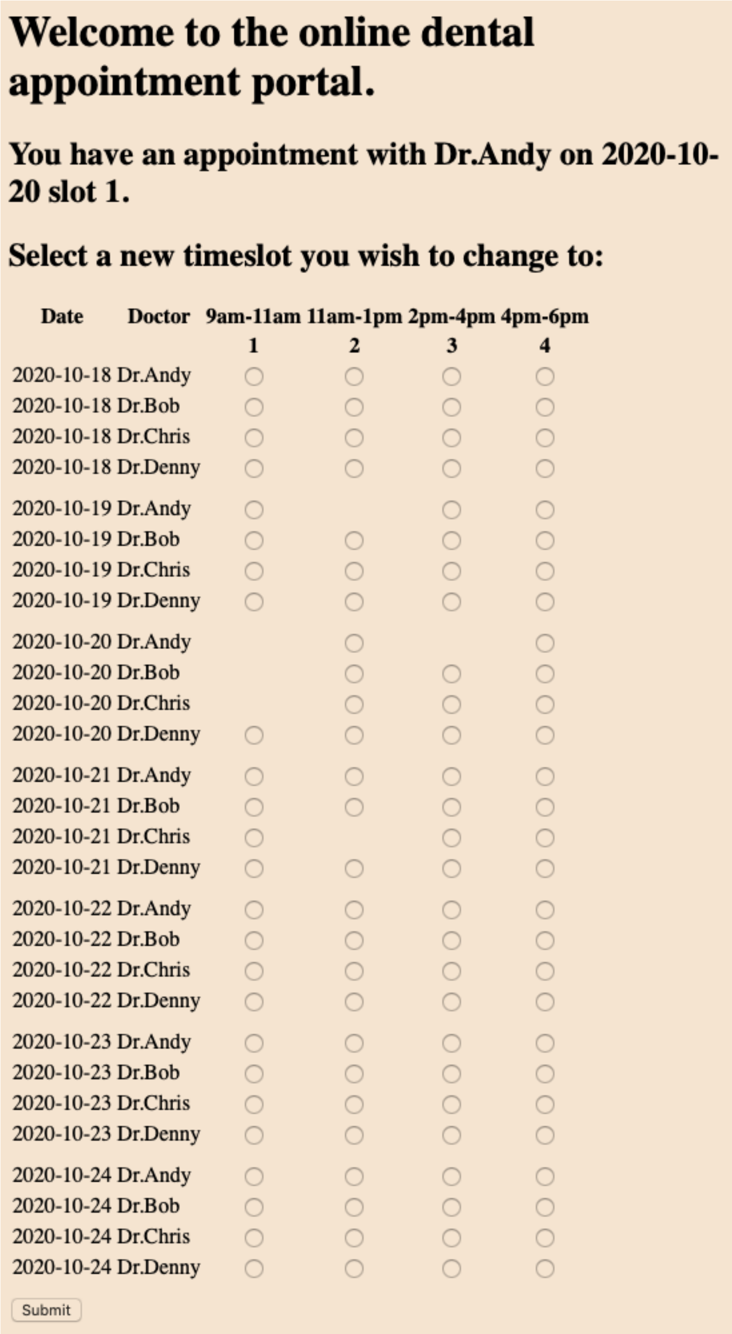


# Welcome to the online dental appointment portal.

### Welcome joe

### You have an appointment with Dr.Andy on 2020-10-20 slot 1.

User will be greeted with his/her appointment (if any), and has the option to modify/add an appointment



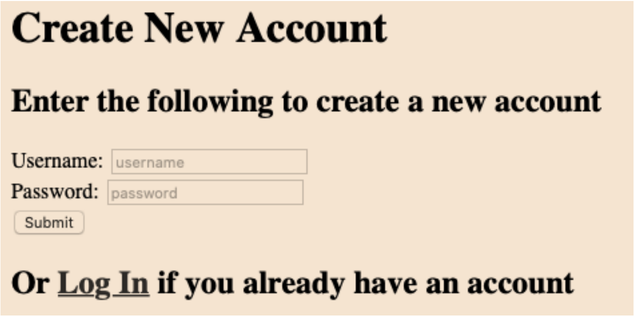
# Welcome to the online dental appointment portal.

## You have an appointment with Dr.Andy on 2020-10-20 slot 1.

## Select a new timeslot you wish to change to:

| Date | Doctor | 9am-11am | 11am-1pm | 2pm-4pm | 4pm-6pm |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 |
| 2020-10-18 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-18 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-18 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-18 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-19 | Dr.Andy | ○ | | ○ | ○ |
| 2020-10-19 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-19 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-19 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-20 | Dr.Andy | | ○ | | ○ |
| 2020-10-20 | Dr.Bob | | ○ | ○ | ○ |
| 2020-10-20 | Dr.Chris | | ○ | ○ | ○ |
| 2020-10-20 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-21 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-21 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-21 | Dr.Chris | ○ | | ○ | ○ |
| 2020-10-21 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-22 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-23 | Dr.Denny | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Andy | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Bob | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Chris | ○ | ○ | ○ | ○ |
| 2020-10-24 | Dr.Denny | ○ | ○ | ○ | ○ |

### 7.3. Sign up for a new account



# Create New Account

## Enter the following to create a new account

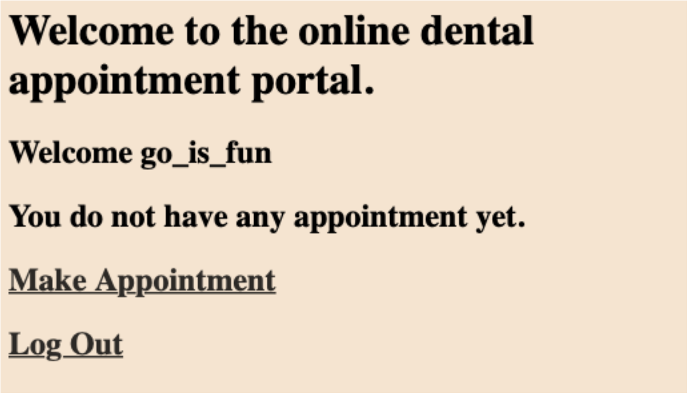
Username: username

Password: password

Submit

## Or Log In if you already have an account

New user has no appointment



# Welcome to the online dental appointment portal.

## Welcome go_is_fun

## You do not have any appointment yet.

## Make Appointment

## Log Out

## 8. Remarks

This application was built on MacOs, the executable created via "go build" is not executable on Windows.

A separate exe file for Windows was create with command line "GOOS=windows GOARCH=amd64 go install".

Run the correct exe file according to the operating system.