CODE DOCUMENTATION

ChenYeh Chien, Wenyue Wu, Kedar N Bhoyar

Umass Boston

CONTENTS

The Front-End Part:

Page Login Page:

/src/containers/login_page/LoginPage.js

Function **LoginPage**():

Call all the child components for the login page.

Parameters:

None

Return:

HTML

/src/components/login_page/loginForm/LoginForm.js

Class **LoginForm**(props):

Sign in with the email and password.
If this is your first time to login, the verification link will be sent to your email.
After click on the link, the user will be able to login to our web app.

Parameters:

props

Return:

HTML

Function **handleChangeEmail**(userEmail):

Change state variable "userEmail" as user input to the text box by setState.

Parameters:

userEmail: local variable

Return:

None

Function **handleChangePassword**(password):

Change state variable "password" as user input to the text box by setState.

Parameters:

password: local variable

Return:

None

Function **handleSubmit**(e):

Sign in to the firebase with email and password.

If success, update the user's information to the database.
At last, reset the state variables to the initial state.
Parameters:
e: event variable
Return:
None

Function **componentDidMount**():
Prevent the page from memory leak by global variable "_isMounted".
Parameters:
None
Return:
None

Function **componentWillUnmount**():
Prevent the page from memory leak by global variable "_isMounted".
Parameters:
None
Return:
None

/src/components/login_page/loginForm/LoginInfo.js

Class **LoginInfo**(props):
Text box for the login page (signup page).
Get the props from parent components and call back when there is some changes in the text box.
Parameters:
props
Return:
HTML

Function **handleChange**(e):
Call back by the props for setting the state variable in the parent component.
Parameters:
e: event variable
Return:
None

/src/components/login_page/loginSocial/Firebase.js

Function **firebase.initializeApp**(config):

Initialize the firebase with the config variable we set in this class.
Parameters:
config: object
Return:
None

Function **Notification.requestPermission**():
Request for the notification of FCM.
Parameters:
None
Return:
None

Function **messaging.onMessage**(payload):
Set the information in the notification to the local variables and display them with FCM.
Parameters:
payload: notification object
Return:
None

/src/components/login_page/loginSocial/LoginFacebookBase.js

Class **LoginFaceBookBase**(props):
Sign in with facebook account.
If it already exists, update some neccesary information to the database;
otherwise, sign up for the account.
Redirect to the dashboard page after the process above.
Parameters:
props
Return:
HTML

Function **handleFacebookLogin**(e):
Sign in with the facebook account you chose and update to the database.
Parameters:
e: event variable
Return:
None

Function **componentDidMount**():
Prevent the page from memory leak by global variable “_isMounted”.
Parameters:
None

Return:
None

Function **componentWillUnmount**():
Prevent the page from memory leak by global variable "_isMounted".
Parameters:
None
Return:
None

/src/components/login_page/loginSocial/LoginGoogleBase.js

Class **LoginGoogleBase**(props):
Sign in with google account.
If it already exists, update some neccesary information to the database; otherwise, sign up for the account.
Redirect to the dashboard page after the process above.
Parameters:
props
Return:
HTML

Function **handleGoogleLogin**(e):
Sign in with the google account you chose and update to the database.
Parameters:
e: event variable
Return:
None

Function **componentDidMount**():
Prevent the page from memory leak by global variable "_isMounted".
Parameters:
None
Return:
None

Function **componentWillUnmount**():
Prevent the page from memory leak by global variable "_isMounted".
Parameters:
None
Return:
None

/src/components/login_page/loginSocial/LoginSocial.js

Function **LoginSocial**():

Call the child components “LoginGoogleBase” and “LoginFacebookBase”.

Parameters:

None

Return:

HTML

/src/components/login_page/LoginFooter.js

Function **LoginFooter**():

Medpick footer for login page.

Parameters:

None

Return:

HTML

/src/components/login_page/LoginLink.js

Function **LoginLink**():

The link connects between login page and sign up page.

Parameters:

None

Return:

HTML

/src/components/login_page/LoginLogo.js

Function **LoginLogo**():

Show the background logo of login page (signup page).

Parameters:

None

Return:

HTML

Page Signup:

/src/containers/signup_page/SignupPage.js

Function **SignupPage**():

Call all the child components of the sign up page.

Parameters:

None

Return:

HTML

/src/components/signup_page/SignupFooter.js

Function **SignupFooter**():

Medpick footer for signup page.

Parameters:

None

Return:

HTML

/src/components/signup_page/SignupForm.js

Class **signupForm**(props):

Including text boxes for the user to sign up.
(Name, Phone, Email, Password)
Create an user acount in the firebase.
If success, insert a new user into our database.

Parameters:

props

Return:

HTML

Function **handleChangeName**(fullName):

Change state variable “fullName” as user input to the text box by setState.

Parameters:

fullName: local variable

Return:

None

Function **handleChangePhone** (phoneNo):

Change state variable “phoneNo” as user input to the text box by setState.

Parameters:

phoneNo: local variable

Return:

None

Function **handleChangeEmail**(userEmail):

Change state variable “userEmail” as user input to the text box by setState.

Parameters:

userEmail: local variable

Return:

None

Function **handleChangePassword** (password):

Change state variable “password” as user input to the text box by setState.

Parameters:

password: local variable

Return:

None

Function **handleSubmit**(e):

Create a firebase user account with the email and password.
If success, insert a new user to the database.
At last, reset all the state variables to the initial state.

Parameters:

e: event variable

Return:

None

Function **componentDidMount**():

Prevent the page from memory leak by global variable “_isMounted”.

Parameters:

None

Return:

None

Function **componentWillUnmount**():

Prevent the page from memory leak by global variable “_isMounted”.

Parameters:

None

Return:

None

/src/components/signup_page/SignupLink.js

Function **SignupLink**():

The link connects between login page and sign up page.

Parameters:

None

Return:

HTML

Page Dashboard:

/src/containers/dashboard/Dashboard.js

Function Dashboard():

Call all the child components of the dashboard page.

Parameters:

None

Return:

HTML

/src/components/dashboard/dashboard_body/DashboardBody.js

Class **DashboardBody**(props):

Load the dashboard images and links from the database for convenience. Pass these information corresponging to the pages they need.

Parameters:

props

Return:

HTML

Function **componentDidMount**():

Prevent the page from memory leak by global variable "_isMounted". Fetch the dashboard's images and links from the database and set them to the state variables.

Parameters:

None

Return:

None

Function **componentWillUnmount**():

Prevent the page from memory leak by global variable "_isMounted".

Parameters:

None

Return:

None

/src/components/dashboard/dashboard_banner/DashboardAd.js

Class **DashboardAd**(props):

Return a auto slider with the images and links receive from the props.
Parameters:
props
Return:
HTML

/src/components/dashboard/dashboard_header/DashboardHeader.js

Class **DashboardHeader**(props):
The header used in most of the page.
It includes sidebar, pinaka logo, and user icon (links to my account page).
And only the header which is in the dashboard page will display the loaction icon (City Sele ction).
Parameters:
props
Return:
HTML

Function **openNav**():
Change the nav bar style when clicking on the open button.
Parameters:
None
Return:
None

Function **handleUser**():
Change the functionalities according to the login state of the user.
If login, display “my account” page; else, display “login” page.
Parameters:
None
Return:
None

/src/components/dashboard/dashboard_header/SideBar.js

Class **SideBar**(props):
Contains links of pages in our web app and the current username who login to our web ap p.
Parameters:
props
Return:
HTML

Function **handleUserName**():

Change the user name above the side bar according to the user’s login state.
If login, display the name of the user; else, display ”Guest”.

Parameters:

None

Return:

String

Function **closeNav**():

Change the nav bar style when clicking on the close button.

Parameters:

None

Return:

None

Function **handleLoginStatus**():

Change the link name to the login page according to the login state of the user.

Parameters:

None

Return:

String

Function **handleSignOut**():

Log out the firebase account and clear all the localStorage variables.

Parameters:

None

Return:

None

/src/components/dashboard/dashboard_header/DashboardLocation.js

Class **DashboardLocation**(props):

Show the current city you are when you come to this page.
You can also choose other city you want for delivery.
(But only few cities are able to support the delivery servie.)

Parameters:

props

Return:

HTML

Function **handleGeoLocation**(location):

Get the lat and lng from the “PharmacyAutoComplete” component and pass them to the function which will get the city name of the location.

At last, close the autocomplete text box after choosing the city.
Parameters:
location: object
Return:
None

Function **handleDropdown**():
Change the state of autocomplete text box according to the current state variable “dropdownState” value.
Parameters:
None
Return:
None

Function **openDropdown**():
Change the current state variable “dropdownState” by setState.
Then, set the style of the autocomplete text box.
Parameters:
None
Return:
None

Function **closeDropdown**():
Change the current state variable “dropdownState” by setState.
Then, set the style of the autocomplete text box.
Parameters:
None
Return:
None

Function **getCity**(addressArray):
Get the city name according to the addressArray which is given from the “PharmacyAutoComplete” component’s lat and lng.
Parameters:
addressArray []: local variable
Return:
String

Function **getCountry**(addressArray):
Get the city name according to the addressArray which is given from the “PharmacyAutoComplete” component’s lat and lng.
Parameters:

addressArray []: local variable

Return:

String

Function **getLocation**(lat, lng):

Get the location's information from lat and lng.

Then, set them to the state variables and localStorage variables.

Function **componentDidMount**():

Prevent the page from memory leak by global variable "_isMounted".

Display city information on the header according to the props value.

If true, display; otherwise, don't display.

Get the user current location if the user allow the request.(The very first time);

Or, display the city name which is already been set to the localStorage variable.

Parameters:

None

Return:

None

/src/components/dashboard/dashboard_header/PharmacyAutoComplete.js

Class **PharmacyAutoComplete**(props):

Handle the pincode autocomplete text box.

Parameters:

props

Return:

HTML

Function **handleChange**(address):

Change the state variable "address" according to the value in the autocomplete text box.

Parameters:

address: object

Return:

None

Function **handleSelect**(address):

Get lat and lng from the API method "geocodeByAddress".

Then, invoke the parent's function by call back function in the props.

At last, reset the state variable to the initial state.

Parameters:

address: object

Return:

None

/src/components/dashboard/dashboard_upload/DahboardUpload.js

Function **DashboardUpload**():

Handle upload the prescription.

Redirect the user to the upload page.

Parameters:

None

Return:

HTML

/src/components/dashboard/dashboard_product/DahboardProduct.js

Function **DashboardProduct**(props):

Receive product information from the database which is passed by the props and display them

Parameters:

props

Return:

HTML

/src/components/dashboard/dashboard_help/DahboardHelp.js

Function **DashboardHelp**():

Link to contact us page or medbox page.

Parameters:

None

Return:

HTML

Page PharmacySelection:

/src/components/pharmacy_selection/PharmacySelection.js

Class **PharmacySelection**(props):

Handle the state changes between pincode and geolocation.

Show the pharmacies according to your location or the pincode you entered in the textbox.

Parameters:

props

Return:

HTML

Function **handleGeoLocation**(location):

Set the state variables according to the value get from the “PharmactAutoComplete” component.

Parameters:

location: object

Return:

None

Function **handleLocation**():

Invoke the method “handleOffPin”.

Request for the current location of the user in order to display the pharmacies nearby. If success, insert a request information to the database and set the state variables to the value which is gotten from the “navigator.geolocation.getCurrentPosition” method.

Parameters:

None

Return:

None

Function **handleOffLocation**():

Set the state variables for the state changes.

Parameters:

None

Return:

None

Function **handleOffPin**():

Set the state variables for the state changes.

Parameters:

None

Return:

None

Function **getPharmacyInfoByPincode**():

Fetch all the pharmacies which are in the database.
Then, invoke the “handleOffLocation” for state changes.
At last, change the state variables for other methods to use.

Parameters:

None

Return:

None

Function **getPharmacyInfoByLocation**():

Fetch all the pharmacies which are in the database.
At last, change the state variables for other methods to use.

Parameters:

None

Return:

None

Function **showPharmacyInfoByPin():**

According to the state variable “isPharmacyExist” value to determine if it has to display the pharmacy’s information or not.

Parameters:

None

Return:

HTML

Function **showPharmacyByLocation():**

According to the state variable “isUserExist” value to determine if it has to display the pharmacy’s information or not.

Parameters:

None

Return:

HTML

Function **ComponentDidMount**():

If user is not login yet, redirect the user to the login page;
else if the user didn’t upload the prescription, redirect to the dashboard page.
Prevent the page from memory leak by global variable “_isMounted”.
Invoke “handleLocation” method for requesting the user’s geolocation when the user is in this page.

Parameters:

None

Return:

None

Function **ComponentWillMount**():

Prevent the page from memory leak by global variable "_isMounted".

Parameters:

None

Return:

None

/src/components/pharmacy_selection/PharmacyInfo.js

Function **PharmacyInfo**(props):

Display pharmacies according to the condition we choose in "PharmacySelection" page. (pi ncode or geolocation)

Parameters:

props

Return:

HTML

/src/components/pharmacy_selection/PharmacyButton.js

Class **PharmacyButton**(props):

After selecting the pharmacy, the order information will be sent to the "confirm pharmacy" page as props.

Parameters:

props

Return:

HTML

Function **handleSelect**():

Set the "order" object according the information we have and pass it to the "confirm_pharmacy" component.

Parameters:

None

Return:

None

/src/components/pharmacy_selection/PharmacyAutoComplete.js

Class **PharmacyAutoComplete**(props):

Handle the pincode autocomplete text box.

Parameters:

props

Return:

HTML

Function **handleClickSearch**():

Invoke “handleSelect” method with the address which the user clicked on in the dropdown list of the autocomplete text box.

Parameters:

None

Return:

None

Function **handleChange**(address):

Change the state variable “address” according to the value in the autocomplete text box.

Parameters:

address: object

Return:

None

Function **handleSelect**(address):

Get lat and lng from the API method “geocodeByAddress”.
Then, invoke the parent’s function by call back function in the props.
At last, reset the state variable to the initial state.

Parameters:

address: object

Return:

None

Page: **Forgot Password**

File: src/containers/Forgotpassword/ForgotPassword.js

Function **ForgotPassword**(props):

To return whole content of FP page.

Parameters:

props: null

Return:

HTML structure, including the labels of <UploadHeader>, <SubHeader> and <FPForm> imported from other files

File: src/components/Forgotpassword/FP-Header/Sub-Header.js

Function **SubHeader**():

Define the structure of SubHeader.

Parameters:

None

Return:

HTML structure

File: src/components/Forgotpassword/FP-Form/FP-Form.js

Class **FPForm**:

Be called by super component, ForgotPassword. The FPForm deals with user's email from input and transmit the email to firebase to perform.

Function **handleEmail**(e):

Set the value of state's email from <input>.

Parameters:

e []: state variable of email

Return:

None

Function **handleResetEmail**(e):

Preform the reset-password operation, transmit the value of email to firebase.

Parameters:

e []: state variable of email

Return:

None

Function **Render**():

To return and show the web real looks like.

Parameters:

None

Return:

HTML structure.

Page Upload:

File: src/containers/upload_prescriptions/UploadPrescription.js

Function **UploadPrescription**(props):
The main framework of Upload Prescription page calls its header and body parts(label) from other JS files.

Parameters:

Props: none

Return:

HTML structure, including <UploadHeader> and <UploadBoady>.

File: src/ components/uploadHeader/UploadHeader.js

Function **UploadHeader**(props):

Display the structure of header.

Parameters:

Props: none

Return:

HTML structure calling the <UploadSubHeader>.

File: src/ components/uploadHeader/UploadSubHeader.js

Fuction **UploadSubHeader**(props):

Display the structure of subheader.

Parameters:

Props: none

Return:

none

File: src/components/upload_presecription/uploadBody/UploadBody.js

Function **handleChange**(e):

Store the image file from the local disk in the sessionStorage.

Parameters:

e []: state variable

Return:

None

Function **handlePharmSelect**():

Check the user whether uploaded the image. If yes, push to the Pharmacy Location page.

Parameters:

None

Return:

None

Function **Render**():

To return and show the web real looks like.

Parameters:

None

Return:

HTML structure calling the <UploadButton> and <PastUploadedButton> from other JS files.

File: src/components/upload_presecription/uploadBody/PastUploadButton.js

Function **HandlePrescriptions()**:

Check user whether has logged in. If yes, push to userOrder Page.

Parameters:

None

Return:

None

Function **Render**():

To return and show the web real looks like.

Parameters:

Props: none

Return:

HTML structure, including <DashboardHeader> and <ContactUsBody>.

//use the same header with DashBoard page

File: src/components/ContactUs_Body/ContactUsBody.js

Page Contact Us:

File: src/containers/contact_us/ContactUs.js

Function **ContactUs**():

The main framework of Contact Us page calls its header and body parts(label) from other JS files.

Parameters:

Props: none

Return:

HTML structure, including<ContactUsBody>.

File: src/components/contact_us/contactUs_Body/ContactUsBody.js

Function **ContactUsBody**():

The main structure of body part

Parameters:

Props: none

Return:

HTML structure, including<UploadHeader>.

File: src/components/contact_us/contactUs_Form/ContactUsForm.js

Class **ContactUsForm**:

Handle the information from input by user, which the user wants to send to the manager.

Function **handleName**(e):

Set the name in the state.

Parameters:

e[]: the value of input

Return:

None

Function **handlePhone**(e):

Set the phone number in the state.

Parameters:

e[]: the value of input

Return:

None

Function **handleMessage**(e):

Set the message in the state.
Parameters:
e[]: the value of input
Return:
None

Function **handleSubmit**(e):
To handle the all the input of user and send the info to the back-end by post.
Parameters:
e[]: the value of input
Return:
None

Function **componentDidMount(**):
React function for performing below codes before the render().
Parameters:
None
Return:
None

Function **componentWillUnMount**():
React function for performing below codes after the render().
Parameters:
None
Return:
None

Function **render()**:
Return the main framework of the contact form.
Parameters:
None
Return:
HTML structure, including the form of input – name, email, phone number and message

Page Medbox:

File: src/containers/medbox/Medbox.js

Function **Medbox**():
The main framework of medbox page calls its header and body parts(label) from other JS files.

Parameters:
Props: none
Return:
HTML structure, including <MedboxProcess> and <MedboxBody>

File: src/components/medbox/medboxBody/MedboxBody.js

Class **MedboxBody**:
The main structure of medbox body part.

Function **handleClick**():
To handle the click operation for selecting nearest medbox. If user has logged in, get the user data from the localStorage, and then send to the back-end by post to update the shipping information.

Parameters:
None
Return:
None

File: src/components/medbox/MedboxProcess/MedboxProcess.js

Function **MedboxProcess**():
The main structure of <MedboxProcess>, called by the function Medbox().
Parameters:
None
Return:
HTML structure

Page My Order:

File: src/containers/my_orders/MyOrders.js

Function **MyOrders**():

The main framework of my orders page calls its header and body parts(label) from other JS files.

Parameters:

None

Return:

HTML structure, including <MyOrdersBody>

File: src/components/my_orders/myOrders_Body/MyOrdersBody.js

Class **MyOrdersBody**:

Function **showOrders**():

To display every order information by calling <MyOrdersInfo> from MyOrdersInfo.js

Parameters:

None

Return:

<MyOrdersInfo>, iterate every order from state orders by map().

Function **componentDidMount**():

React function, performed before render(). Call the data of user orders from the back-end by post(), and set them to the state.

Parameters:

None

Return:

None

Function **Render**():

To return and show the web real looks like.

Parameters:

None

Return:

HTML structure.

File: src/components/my_orders/myOrders_Info/MyOrdersInfo.js

Class **MyOrdersInfo**:

The main structure of order information.

Function **handleOrderView**():

To handle the click operation, push to the order information page

Parameters:

None

Return:

None

Function **handleOrderStatus**():

To display order's status, Processing, Accepted, Declined, or Delivered.

Parameters:

None

Return:

the CSS features

Function **handlePharmLength**():

To handle the situation that pharmacy name is longer than 30, the part will be replaced by "…".

Parameters:

None

Return:

pharm_name: pharmacy name from props

Function **render**():

To return the main structure of order information

Parameters:

None

Return:

HTML structure

Page Order Information:

File: src/containers/order_view/OrderView.js

Function **OrderView**(props):

The main framework of Order Information page calls its header and body parts(label) from other JS files.

Parameters:

props[]: the state variable for transmitting user orders

Return:

HTML structure, including <OrderViewBody>

File: src/components /order_view/OrderViewBody.js

Class **OrderViewBody**:

Display the order details, including prescription image, customer name, order date, order time, pharmacy name, pharmacy address, and pharmacy phone number.

Parameters:

None

Return:

HTML structure.

Page My Account:

File: src/containers/my_account/MyAccount.js

Function **MyAccount**():

The main framework of my account page calls its header and body parts(label) from other JS files.

Parameters:

Props: none

Return:

HTML structure, including <User>

File: src/components/my_account/User.js

Class **User**:
The main structure of user information.

Function **handleUpdate**(e):

To deal with the update operation, send the user info - name, phone to the back-end by post().

Parameters:

e[]: the value from user input

Return:

None

Function **handleSetName**(e):

To set the user's name for state.

Parameters:

e[]: the value from user input

Return:

None

Function **handleSetPhone**(e):

To set the user's phone number for state.

Parameters:

e[]: the value from user input

Return:

None

Function **componentDidMount**():

React function, perform before the render(). Set the name, phone, and email from localStorage to the state.

Parameters:

none

Return:

None

Function **render**():

To return the main structure of user information

Parameters:

None

Return:

HTML structure

Page Order Confirm:

File: src/containers/confirm_pharmacy/ConfirmPharmacy.js

Function **ConfirmPharmacy**():

The main framework of Pharmacy Confirm page calls its header and body parts(label) from other JS files.

Parameters:

props[]: the value of local properties

Return:

HTML structure

File: src/components/confirm_pharmacy/comfirmForm/ConfirmForm.js

Class **ConfirmForm**:

The main structure of Order Confirm page.

Function **handleName**(e):

To set user's name to the state.

Parameters:

e[]: the value from user input

Return:

None

Function **handleEmail**(e):

To set user's Email to the state.

Parameters:

e[]: the value from user input

Return:

None

Function **handlePhone**(e):

To set user's phone number to the state.

Parameters:

e[]: the value from user input

Return:

None

Function **handleSubmit**(e):

To handle user's submit operation. Collect the information to the FormData, and then send to the back-end by post().

Parameters:

e[]: the value from the state

Return:

None

Function **render**():

To return the main structure of confirm form.

Parameters:

None

Return:

HTML structure, including a <form> for collecting the user information – name, email, and phone

File: src/components/confirm_pharmacy/ConfirmInfo/ConfirmInfo.js

Class **ConfirmInfo**:

The main structure of confirm information, the uploaded prescription, the selected pharmacy name and address.

Function **componentDidMount**():

React function, performed before the render(). Set the pharmacy name, address, and prescription image from sessionStorage to the state.

Parameters:

None

Return:

None

Function **render**():

To return the main structure of pharmacy information and prescription.

Parameters:

None

Return:

HTML structure

Page Thanks:

File: src/containers/thank_page/ThankPage.js

Function **ThankPage**():

The main framework of Thanks page calls its header and body local from other JS files.

Parameters:

Props: none

Return:

HTML structure, including <ThankPageHeader>, and <ThankPageBody>

File: src/components/thank_page/thankPage_Header/thankPageHeader.js

Function **ThankPageHeader**():

The main structure of header.

Parameters:

Props: none

Return:

HTML structure, the <button> links to the DashBoard page

File: src/components/thank_page/thankPage_Body/thankPageBody.js

Class **ThankPageBody**:

The main structure of body.

Function **componentDidMount**():

React function, performed before the render(). Count the time and push to the DashBoard after 3000 ms.

Parameters:

none

Return:

None

Function **componentWillUnmount**():

React function, performed after the render(). Clear the time count.

Parameters:

none

Return:

None

Function **render**():

To return the main structure of success and thanks information.

Parameters:

None

Return:

HTML structure

Router

File: src/components/react_router/ReactRouter.js

Class **ReactRouter**:

To implement the transfer between different pages.

Function **render**():

Implement route rules by <Switch> and <Route> imported from react-router-dom.

Parameters:

none

Return:

HTML structure

The Back-End Part:

File: server/index.js

The server uses express as the framework.

Function **use**(path, router):

Express function, be used to define the route rules, when there are multiple route rules.

Parameters:

path: the path is defined to accept the variables from post and get requirements

router: the object is created by express.Router()

Return:

None

Function **get**(path, router):

Express function, be used to define the route rule, when there is a single rule.

Parameters:

path: the path is defined to accept the variables from post and get requirements

router: the object is created by express.Router()

Return:

None

Function **listen**(port, error):

Express function, be used to listen the date from a certain port.

Parameters:

port: the port number

error: error throw

Return:

None

File: server/routes/userSignup.js

Function **route**('/signup'):

Express router function, accept the data from the path with 'signup'

Parameters:

'/signup': URL

Return:

None

Function **post**(req, res, next):

Express router function, to handle the accept the user data from the requirement, insert the data into MySQL DB, and respond the result.

Parameters:

req: request

res: response,

next: nextFunction

Return:

None

File: server/routes/userOrders.js

Function **route**('/my_orders'):

Express router function, accept the data from the path with 'my_orders'

Parameters:

'/my_orders': URL

Return:

None

Function **post**(req, res, next):

Express router function, to handle the accept the user data from the requirement, search the data from MySQL DB, and respond the result.

Parameters:

req: request

res: response,

next: nextFunction

Return:

None

File: server/routes/userDetail.js

Function **route**('/userDetail'):

Express router function, accept the data from the path with 'userDetail'

Parameters:

'/userDetail': URL

Return:

None

Function **post**(req, res, next):

Express router function, to handle the accept the user data from the requirement, update the data in MySQL DB, and respond the result.

Parameters:

req: request

res: response,

next: nextFunction

Return:

None

File: server/routes/userInfo.js

Function **route**('/userInfo'):

Express router function, accept the data from the path with 'userInfo'

Parameters:

'/userInfo': URL

Return:

None

Function **post**(req, res, next):

Express router function, to handle the accept the user data from the requirement, update the data of customers in MySQL DB, and respond the result.

Parameters:

req: request

res: response,

next: nextFunction

Return:

None

File: server/routes/pharmacy.js

Function **route**('/pharmacy'):

Express router function, accept the data from the path with 'pharmacy'

Parameters:

'/ pharmacy ': URL

Return:

None

Function **post**(req, res, next):

Express router function, to handle the accept the user data from the requirement, search the data of pharmacies from MySQL DB, and respond the result.

Parameters:

req: request

res: response,

next: nextFunction

Return:

None

File: server/routes/dashboard.js

Function **route**('/dashboard'):

Express router function, accept the data from the path with 'dashboard'

Parameters:

'/dashboard ': URL

Return:

None

Function **post**(req, res, next):

Express router function, to handle the accept the user data from the requirement, search the data from Pinaka_dashboard table MySQL DB, and respond the result.

Parameters:

req: request

res: response,

next: nextFunction

Return:

None

/server/routes/insert_prescription.js

Function **admin.initializeApp**(config):

Initialize the firebase with the config variable “credential” and “databaseURL”.

Parameters:

config: object

Return:

None

Function **router.route('/insert_prescription_order').post(req, res, next)**:

Do the database operation for inserting the order which is submitted by the user in the front end page “confirm pharmacy” page.

Parameters:

'/insert_prescription_order': URL

req: request

res: response,

next: nextFunction

Return:

None

Function **emailSender**(emailAddress, fileName):

Send email to the “emailAddress” and the attachment with the “filename” specified.

Parameters:

emailAddress: String

filename: String

Return:

None

/server/routes/contact_us.js

Function **router.route('/contact_us').post(req, res, next)**:

Sending the email to the medpick with the information which is typed by the user in the “contact_us” page (front end)

Parameters:
'/contact_us ': URL
req: request
res: response,
next: nextFunction
Return:
None

Function **emailSender**(name, phoneNumber, message):
Send email to the medpick with “name”, “phoneNumber”, and “message” specified above as parameters.
Parameters:
name: String
phoneNumber: String
message: String
Return:
None

/server/routes/request_pharmacy_by_your_location.js

Function **router.route('/request_pharmacy_by_your_location').post(req, res, next)**:
Insert the request in the “pharmacy selection” page to the database.
If it is come from “Medbox” page, send an email to the medpick.
Parameters:
'/request_pharmacy_by_your_location': URL
req: request
res: response,
next: nextFunction
Return:
None

Function **emailSender**(name, phoneNumber, email, lat, lng, city):
Send email to the medpick with “name”, “phoneNumber”, “email”, “lat”, “lng”, and “city” specified above as parameters.
Parameters:
name: String
phoneNumber: String
email: String
lat: String
lng: String
city: String
Return:
None