**Scheduling program**

1. **Description**

The program will be used to schedule meeting between provider and customer for example dog walker and dog owner or a doctor and a patient.

The doctor will have a calendar with days and slots.

The patient will have slots to choose from and will select the slot.

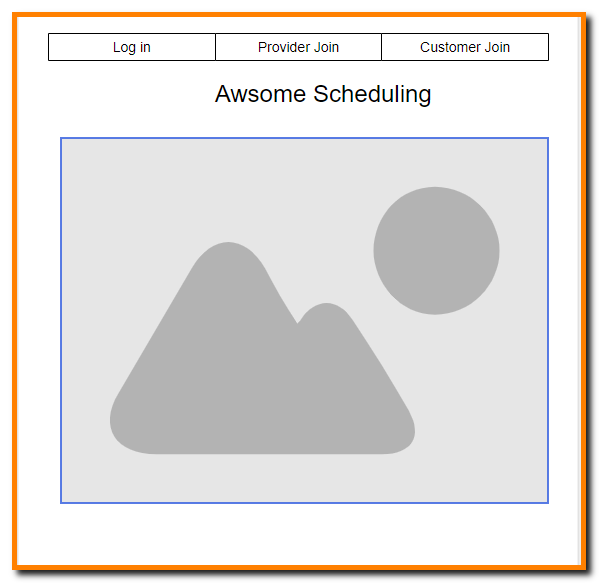
1. **Scenarios**
2. Provider Join
3. Starts the program.
4. Selects the Menu option of Provide Join
5. Enters the data of the provider.
6. Customer Join
7. Starts the program.
8. Selects the Menu option of Customer Join
9. Enters the data of the customer.
10. Setting a session

Done by a customer.

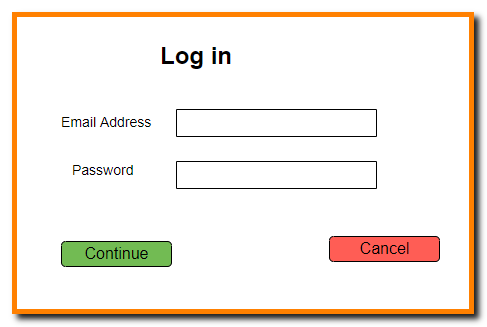
1. Log in using email and password.
2. The program opens on the customer schedule.
3. Selects a new session
4. Enter the data.
5. Cancelling a session

Done by customer.

1. Log in using email and password.
2. Program opens to his session list
3. Selects the session to be deleted.
4. Deletes the session.
5. Managing provider sessions
6. Log in using email and password.
7. Opens his session list
8. If needed deletes a session.
9. Sending reminders
10. When a session is scheduled by a customer, the program sends an email to the provider
11. When a session is canceled by a customer, the program sends an email to the provider.
12. When a session is cancelled by a provider, the program sends an email to the customer.
13. When a session is about to start, the program sends an email to both customer and provider.
14. **Records**
15. Provider
16. Customer
17. Service
18. session
19. **Screens**
20. Main Screen
21. Provider Joining Screen
22. Customer Joining screen
23. Login
24. Provider Screen
25. Customer screen
26. Session
27. **Main screen**



1. Navigation
2. Log in: opens log in screen
3. Provider Join: opens Provider Join Screen
4. Customer Join: opens Customer Join Screen
5. Exit:
6. **Log in Screen:**



1. Continue button:

The program looks for the user if not found gives an error message,

Else if the user is customer, it opens Customer screen if the user is Provider, it opens provider screen.

1. Cancel button:

Returns to Main Screen.

1. **Provider Join**



1. Save button:

The program verifies that the record is valid if not valid gives an error message,

Else opens the Provider screen.

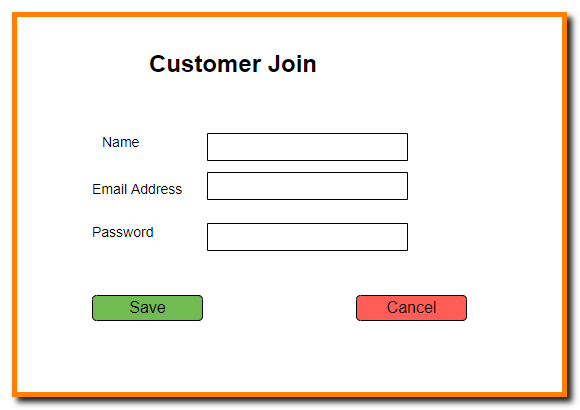
If the Service is not in the Service Table, the program adds it.

When saving, the Service ID is saved to the provider table.

1. Cancel button:

Returns to Main Screen.

1. **Customer Join**



1. Save button:

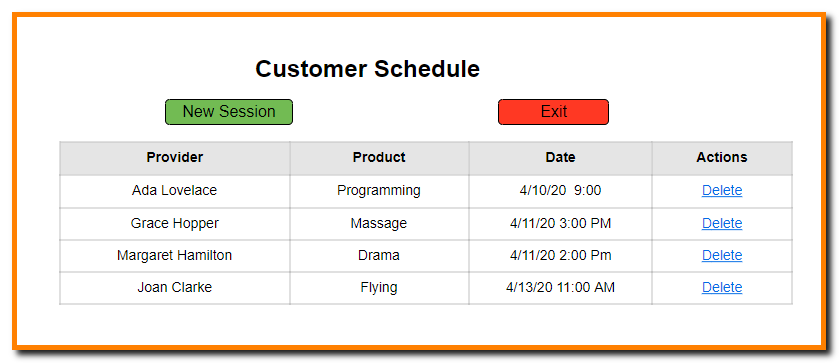
The program verifies that the record is valid if not valid gives an error message,

Else saves and opens the Customer screen

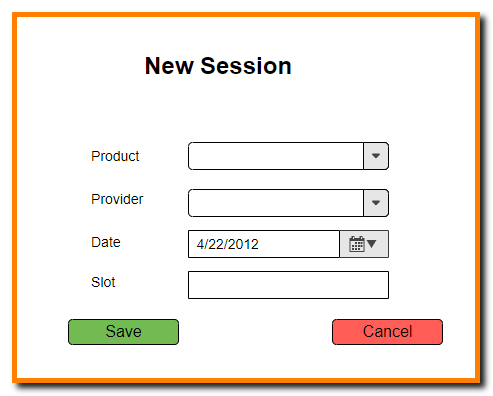
1. Cancel button:

Returns to Main Screen.

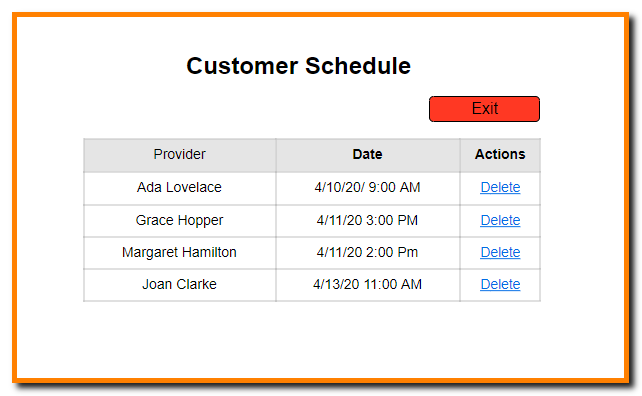
1. **Customer screen**



1. List of sessions each Session includes
2. List of grid items
3. Provider.
4. Service
5. Date
6. Time
7. Buttons
8. New Session: Opens the New Session Screen.
9. Exit Return to the main menu.
10. Delete: Deletes the session.
11. **New Session**:



1. Fields:
2. Service: The drop down includes all the Services, This items needs to be selected first.
3. Provider: If no Service selected needs to select Service. The dropdown includes all provider that offer the Service.
4. Date: If no provider selected needs to select provider.
5. Slot: 1 to the number of slots that the Provider offers.
6. **Provider screen**



1. List of sessions each Session includes
2. Customer
3. Date
4. Time
5. Button for delete
6. Buttons
7. Exit Return to the main menu.
8. Delete: Deletes the session.
9. **Database records**
10. Provider
11. id
12. name
13. email
14. password
15. service\_id
16. daily\_slots.
17. Customer
18. id
19. name
20. email
21. password
22. Session
23. id
24. customer\_id
25. provider\_id
26. service\_id
27. date
28. slot
29. Services
30. id
31. name
32. provider\_id

query all provider

for each one, go into services

loop through services array to see if any matches service id

return those provider

vs

query services

return all matching providers

1. Comments
2. Id of user is the email address
3. Future expansion is using Zoom API to schedule a Zoom conference for each session so when creating a session, the program will create a zoom conference and send the invite to the provider and the customer.

ReactDate: <https://www.npmjs.com/package/react-datepicker>

React stepper: <https://www.npmjs.com/package/react-stepper-horizontal>

Todo list

1. Division of work
2. Front end Kevin
3. Backend Shlomo
4. Working in Branches
5. Adding the lint
6. Following the database
7. Need to make sure that email is unique in both customer and provider.

React Outline

1. Containers – class components, ComponentDidMount loads user data from db
   1. Navbar – “const location = useLocation()”, active classes
   2. Home – Landing page, app information, user instructions, links to about us and get started
   3. About – Talks about the project, link to repo, link to github profiles
   4. Dashboard – After a user is signup and booked, this is where they see their exisiting appointments
   5. Provider Registration – Stepper form, 2 steps
      1. Provider screen
      2. Provider Dashboard.
   6. Customer Registration
      1. Customer screen
      2. Customer Dashboard
   7. New Session
      1. Select Service
      2. Select Provider
      3. Select Session.
   8. Login
      1. Login screen
      2. Dashboard.
      3. Pick Account Type – select customer or provider
      4. Signup/Login – email password
      5. User Details – render form for customer or provider
      6. Pick Services – query all services
      7. Pick Provider – query all provider for selected services
      8. Schedule Time – query all time slots for selected provider
      9. Confirmation – showing booked appointment
      10. Redirect to Dashboard
   9. Scheduling
      1. Pick Provider + Pick Services
      2. Schedule Time
      3. Confirmation
      4. Redirect to Dashboard
2. Components
   1. Header – app title
   2. Footer – github usernames
   3. PageTitle
   4. Form – versioned out for each registration step
   5. Appointment – render each scheduled session (on confirmation page or on dashboard)
   6. Calendar – show all scheduled Appointments (render component for each session)
   7. Profile – after user enters all information, a card to display user’s information
   8. Scheduler – date picker to actually select calendar date value for appointment

React Folder Structure

* Client
  + Public
  + Src
    - Components
      * Header
        + Index.js
        + Styles.css
    - Containers
      * Home
        + Index.js
        + Styles.css
      * Dashboard
        + Index.js
        + Styles.css
      * About
        + Index.js
        + Styles.css
      * Navbar
        + Index.js
        + Styles.css
      * Registration
        + Index.js
        + Styles.css

REPO SETUP

1. Clone manny’s boilerplate
2. Delete .git file (rm -rf .git)
3. Create new repo
4. Set up collaborators on github
5. Set up master branch rules on github
6. Link repo and local folder ( get init…)
7. Add .gitignore (node\_modules, package-lock.json, DS\_Store, .env, eslint file as well)
8. Install Travis and link to github – install in server
9. Install ESLint – install as dev dependency on server ( -dev)
10. Apply Airbnb eslint style – see link in slack – each person does it
11. Install and setup dotenv (environment variables and secret api keys)
12. Create and update .env file in root folder (each person)
13. Setup JawsDB on Heroku (live mysql db)
14. Create a Heroku app
15. Push to Heroku
16. Add collaborator on Heroku
17. Pick react library for date picker and install inside client folder

Switch( case)

Case 1:

Return (

<RegType /> // Step1

)

Form Component

Const CustomerForm = [

{

Label: “”,

Validate: [],

Name: “”

}

]

Const ProviderForm = [

{

Label: “”,

Validate: [],

Name: “”

}

]

Function Form() {

Return (

If (regType === “customer”) {

Return <Form fields={customerForm} />

}

If (regType === “provider”) {

Return <Form fields={providerForm} />

}

)

}

DROP DATABASE IF EXISTS schedule\_db;

CREATE DATABASE schedule\_db;

USE schedule\_db;

CREATE TABLE provider (

  id INT AUTO\_INCREMENT NOT NULL,

  name VARCHAR(60) NOT NULL,

  email VARCHAR(60) NOT NULL,

  password VARCHAR(20) NOT NULL,

  service\_id INT,

  daily\_slots INT,

  PRIMARY KEY (id)

);

CREATE TABLE customer (

  id INT AUTO\_INCREMENT NOT NULL,

  name VARCHAR(60) NOT NULL,

  email VARCHAR(60) NOT NULL,

  password VARCHAR(20) NOT NULL,

  PRIMARY KEY (id)

);

CREATE TABLE session (

  id INT AUTO\_INCREMENT NOT NULL,

  customer\_id INT NOT NULL,

  provider\_id INT NOT NULL,

  service\_id INT NOT NULL,

  date DATE NOT NULL,

  slot INT NOT NULL,

  PRIMARY KEY (id)

);

CREATE TABLE service (

  id INT AUTO\_INCREMENT NOT NULL,

  name VARCHAR(50) NOT NULL,

  PRIMARY KEY (id)

);