Feature Wish List:

1. Expand to have multiple mechanics and track data accordingly
2. Allow for account creation with hashed passwords
   1. Accounts are secure
   2. Two account Types with different permissions
   3. Different data shown per account
3. Setup Mock Payments System
4. Find a way to include an API and connect it to my database
5. Dynamic drop down search results

* Goals for Today:
  + Finish getTicketById function
    - Will require getting item display figured out
    - Might need to split into two get functions
  + Search for items and populate on search
    - Each matching item has a add button (or just click div)
    - If no items exist show “No matching items”
  + Create item button
    - Shows create item form
    - On submit creates new item in DB and adds it to ticket
* To Do List
  + CSS for home page
    - Try to make sure it works across pages
  + All database queries, posts and such need to be formatted

This site will be from a mechanics point of view, they are creating, modifying, deleting service tickets for customers.

**Minimum Viable Page:**

Home Page

* Header
  + Company logo plus name
  + Nav links
* Main Body
  + Automatically displays 10 pre created Service Tickets
    - Tickets will contain basic info
      * Customer name
      * Bike type
      * Due date
      * Price
    - Each Ticket Item will have a delete button
      * Delete button will delete from server (or DB)
      * There will also be a drop down menu for each that can update the status of the ticket
* Footer
  + Create new Ticket form
    - Inputs for the 4 data types and a submit button
    - Adds new ticket item to backend and will display on main section

Notes:

* Database would be a local .json file for simplicity

Three Main Features (could leave out one, new ticket or dropdown if they cause trouble)

1. Getting ticket info and displaying it on the main section
2. Delete button deletes ticket
3. Status update dropdown menus
4. Create new ticket form and show it on main section

**Main goal for the Site:**

All information is handled with a database

**Home Page**

* Nav Bar, Home Page, Create new Ticket button, Highlighted Bikes Page
* Ticket searching *(this functionality could be further expanded if there is time)*
  + Just under the header will be a search form
    - Three elements of search
      * Input for name
      * Dropdown for status filter (default to all)
      * Submit button
* Off the bat the page will display ticket items with the closest due dates showing first
  + This will repopulate upon a search query
    - *(might be nice to have a way to view more items)*
* Ticket Display Format
  + Tickets will be individual row style divs
  + The div can be clicked on to go to a ticket editing page
    - Each div will have a clear border and change color when hovered over to indicate it is a button
* Footer with company information

**Create New Ticket Page**

* Nav Bar, Home Page, Create new Ticket button, Highlighted Bikes Page
* Search bar at the top for finding customer information
  + Upon search populate a drop down list or side bar with clickable profile options (name/number)
  + If one is clicked, then autofill the new ticket form with their information
    - If no customers exist, then their information will be saved when the form is submitted
  + Search could also create a side bar with options which appears and then disappears after a name is selected, or x’d out of
* New ticket form
  + Customer info, Bike info, Repair info
    - Certain fields will be required
    - Phone number will need to be reformatted
  + Bike status
  + Mechanic doing work
  + Scheduled date for work
  + Ticket Item Table (lists out all individual charges)
* Ticket Item Table Format
  + Small form to search for existing items
  + Each ticket item will be a row in a table
    - Each will have a delete button
* Footer with company information

**Ticket Editing Page** - (only accessible by clicking on an existing ticket)

* Nav Bar, Home Page, Create new Ticket button, Highlighted Bikes Page
* Basically same layout as create new ticket
  + Remove customer search bar
  + Customer and bike info are displayed nicely, an additional edit button will turn them into input fields that can be edited
  + Ticket items table is ready to be edited right away
  + Ticket status ready to be edited right away
* Footer with company information

**Highlighted Bikes Page** - (optional, only if everything else is flushed out)

* Nav Bar, Home Page, Create new Ticket button, Highlighted Bikes Page
* Page will contain bike cards of highlighted bikes
  + Maybe include rating system?
  + With extra time could include, add, delete and modify functionality
  + Potentially grab info from an API?
* Footer with company information

**Files Layout:**

* Front End
  + HTML files
    - home.html
    - newTicket.html
    - highlighted.html
  + CSS files
    - Stayles.css
  + JS files
    - Home.js
    - newTicket.js
    - highlighted.js
* Back End
  + Back-end files
    - Server.js
      * Require dot env
      * Require express
      * Require cors
      * Require seed
      * Require controller
    - Controller.js
      * Require dotenv
      * Require sequelize
      * Module.exports
    - Seed.js
      * Require dotenv
      * Require sequelize
      * Module.exports
  + Database
    - Data.io
* Packages
  + Initialize project
  + Express
  + Cors
  + Sequelize
  + Pg
  + Pg-hstore
  + Dotenv

**Database Layout:**

* Clients
  + Client\_id
  + First name
  + Last name
  + Phone
  + Email
* Bikes
  + Bike\_id
  + Brand
  + Model
  + Color
  + Size
  + Client\_id
* Tickets
  + Ticket\_id
  + Client\_id
  + Bike\_id
  + Status\_id
* Ticket-Status
  + Status\_id
  + status
* Tickets-items
  + Ticket\_item\_id
  + Ticket\_id
  + item\_id
* Items
  + Item\_id
  + name
  + Price

Fallback Plan:

Combine home and ticket view page

This will allow for all data to be accessessed

Have it do a class toggle?

New plan:

* Use similar home page .get and display the tickets in a side bar, also include search function

* Functions
  + GET – search for items
    - Will technically create a new ticket-item reference
  + POST – new ticket items
    - New ticket items can be created if not available through search
  + DELETE – ticket Items
    - Pressing X on ticket item will delete it (will not delete item, just reference)
  + DELETE – ticket
    - Pressing delete button at top can delete entire ticket and related ticketitems
  + PUT – ticket
    - Pressing edit button at top will allow for editing of client and bike info (updating the selected client or bike row)
* Need to create tables that have column names
* Need to provide space for search results
* Moved displayTicket into ticketView.js
* Changed add ticket function call SQL
  + Inserts ticked and just returns status of 200
* Front end receives status 200 and calls newticket callback
  + Newticketcallback just opens the viewTicket window
* On load viewTicket runs display ticket which queries the DB for the most recent ticket information
  + Attaches that information to web page for display