**MedFind Final Project Documentation**

MedFind is a mobile medical price finder app. The purpose of this app is to inform consumers of the Government regulated price for medication. These prices are specified by legal price requirements specified government legislations.

The mobile app has 2 components that make the app work.

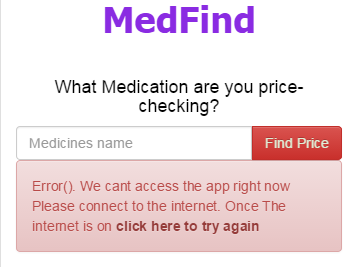
Frontend: Ionic Framework HTML based mobile app. This Application uses the AngularJS JavaScript library.

Back-end: JavaScript library based API. The API was built using JavaScript libraries such as:

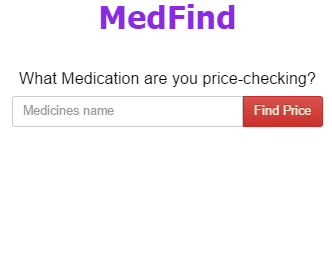
* Csv-parser: Used to convert the csv excel sheet into a JSON file, which is used to populate the database.
* Express: Used as a web service.
* Moment: Date Library.
* Sequelize: Simplifies SQL queries and creates the database in SQLite3.

The API is built in Node.js and can be found in the server.js.

When the app is loaded one of 2 things will happen. If the mobile data connection is switched off, the app will load with the MedFindError elements. As you can see with the div element in red it tells the user to connect to the internet.

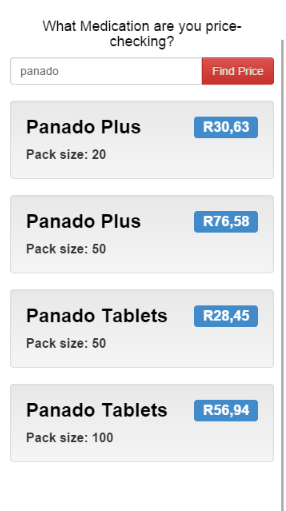


As soon as the user is connected to the internet and clicks the try again link. The application page will load without the error message. This page asks the user to input the medication that they want to price-check.

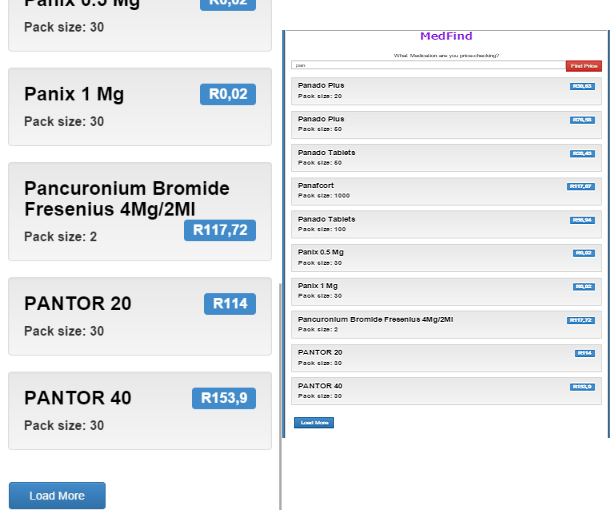


Once the user has typed the name of the medication they want to price-check, the app will spit out a repeater view of “div-elements” that match the name of the requested medication.

Inside the div elements you will see a Name Element, Pack Size element which indicates how many units of medication is in the medicine pack, and a Price indication in blue telling the user how much the pharmacy can charge for the medication.



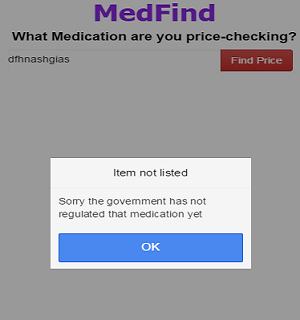
I have set the page to only display 10 items per page in the server.js file,this is important because a user could accidently input “a” into the searchbox and press enter. This is a problem (that the page filter fixes) because then the app would generate 685 records which would consume too much of the users mobile data. The page filter is also important if the user will query a generic name (like:pan) that has more then 35 medications which have a name that starts with the name being queried in the search box. If this is the case the user will be able to load more medications, by clicking the load more button, untill there is no more medications in the database that start with the queried name.



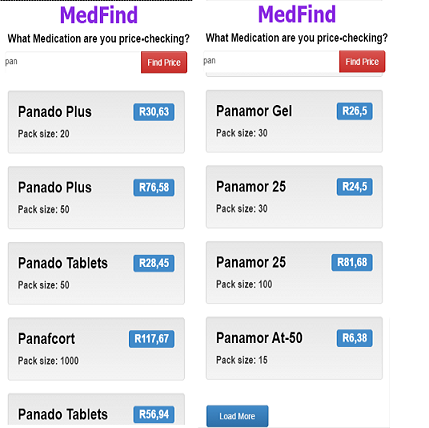
**Current system flow**

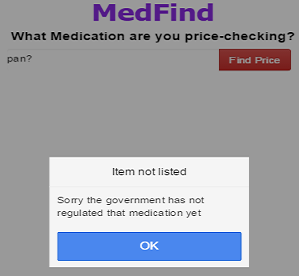
**Final Distribution changes**

After some testing Geoffery and I came up with 5 bugs that needed fixing.

1. An alert message telling the user that there aren’t any records in the database when the user inputs a medication that is not listed by the government. This message is also important for when the user mistakenly inputs rubbish data like, “dfhnashgias”. I achieved the required result by
   1. using the “$ionicPopup.alert” method found in the MedFind.js file
2. Keeping the search elements at a static position of the page, making it easier for the user to make new searches without having to scroll all the way to the top of the page again. I achieved this outcome by:
   1. Modifying the html and css of the index.html.
   2. Creating a new css class called “fixedElement1”.
   3. Setting the scroll attribute of the ion-content to true.
   4. Removing the “priceFinderDiv” element out of the ion-content and placing it in the ion-pane element.
   5. Setting the backround colour of the element to “ActiveBorder” this change gives it an ‘i-frame’ like effect.

As you can see in the image below the search bars position stays fixed no matter how many records are populated



1. When a user inputs non-alphanumeric values like: “?, <, >, !’ etc.. the app showed the MedFindError element which is hidden unless the app encounters an error, which stays there until you refresh the page. To remove this problem I did:
   1. I utilised the AngularJS “ng-pattern” directive and filtered it with an alphanumeric filter: “ng-pattern="/^[a-zA-Z0-9 ]\*$/" required="true"”. Now if the user inputs non-alphanumeric characters the user will just be shown the alert message used for rubbish data, and the MedFindError element stays hidden.
2. Geoffrey mentioned that when the user is searching for Medications he/she might want to activate the search on a keyboard stroke using the “Enter” key. To achieve this I had to:
   1. Add a custom directive called “ngEnter” to the Medfind.js Controller.
   2. Bind the “keydown keypress” functionality in a function that relies on the variable called “event”.
   3. Bind the function to the HTML page with the “$scope.$apply” variable.
   4. Adding the ng-enter directive to the text input in the HTML page.
3. Disabling the “Load More” button when there are no more records to load. I wanted to do this because when the user runs out of records to load and presses the “Load More” button the alert message telling the user that the medication doesn’t exist. To achieve this I did:
   1. Added a conditional if statement in the MedFind.js controller’s “LoadMore” function saying if there is no more records left the “Load More” button must be disabled.