README

Stack used:

For the Front end:

- CSS
- HTML
- JavaScript
- JQuery

For the Back end:

- PHP
- MySQL

Design Pattern/Code architecture:

To help keep everything modular I used the MVC design pattern to separate the business logic, database and UI from each other. The controllers consist of the controller.php file and the script.js file. The model consists of the model.php file and the views are all located in the views folder.

Test driven development:

For this project I decided to approach the problem by writing unit tests first. I wrote the tests first to help identify possible errors and inputs that could cause the program and or any functional code to fail, moreover, this would give me a good idea of what the desired behaviour of the program/webapp should be.

Testing Framework:

For this project I used the Qunit testing framework to run my automated tests. To view these tests, you can view their results online by clicking the “view unit tests” button or by opening the test.js file to view the code.

Responsiveness and mobile first:

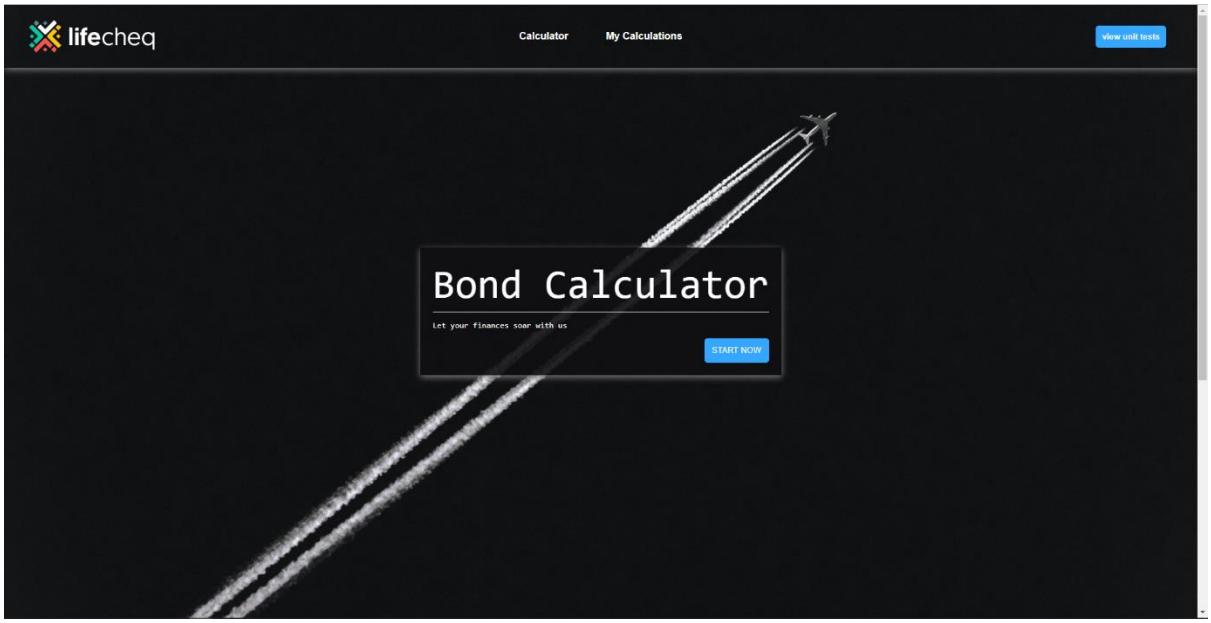
To achieve the desired responsiveness, I used css media queries, however, due to time constraints breakpoints were kept to a minimum. When doing the styling and design I kept to the principle of “designing for mobile first” so that media queries would be fairly lightweight and easy to implement.

Data safety, Hashing and SQL injection attacks:

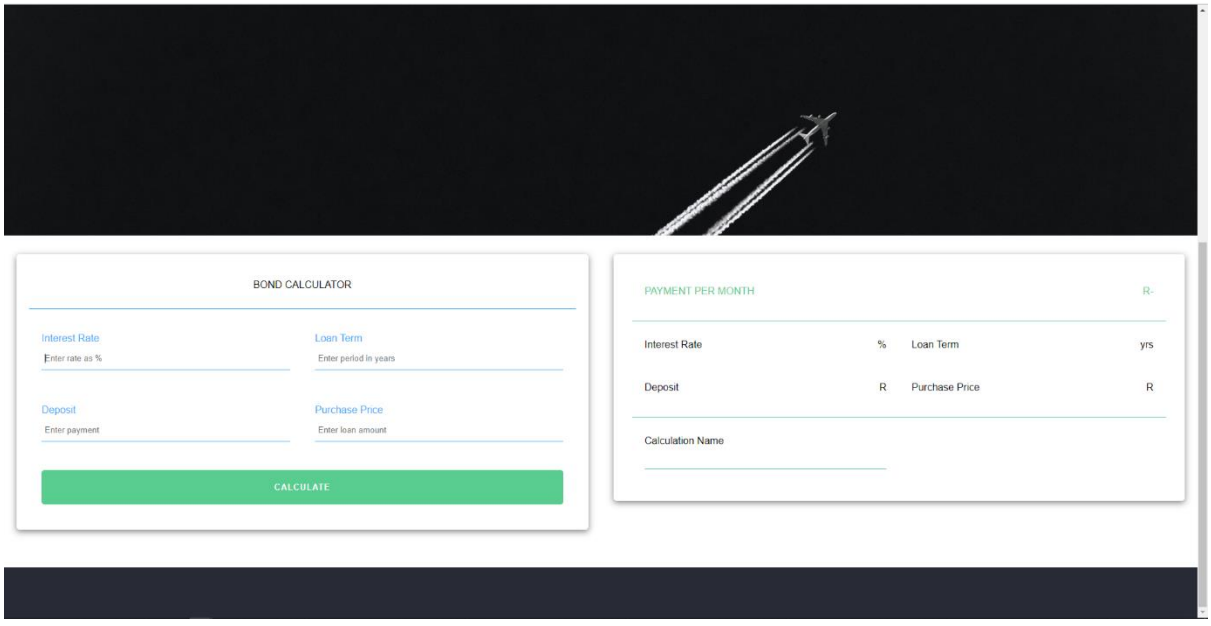
Data security is in my opinion, extremely important online when creating a webapp. If this webapp were to be extended in functionality where authentication or a log in system were to be implemented the users login details such as the password would need to be hashed in such a manner so that decrypting it would be very difficult to do. This is especially important as users tend to use the same passwords repeatedly so exposing a password on one site could lead to it being exposed on many other sites.Keeping up to date with the latest encryption methods and straying away from deprecated technologies e.g. md5 in PHP is very important. Moreover, to avoid SQL injection attacks, malicious characters need to be escaped. To achieve this, I used the PHP function mysqli_real_escape_string() to escape any such characters.

Time Constraints:

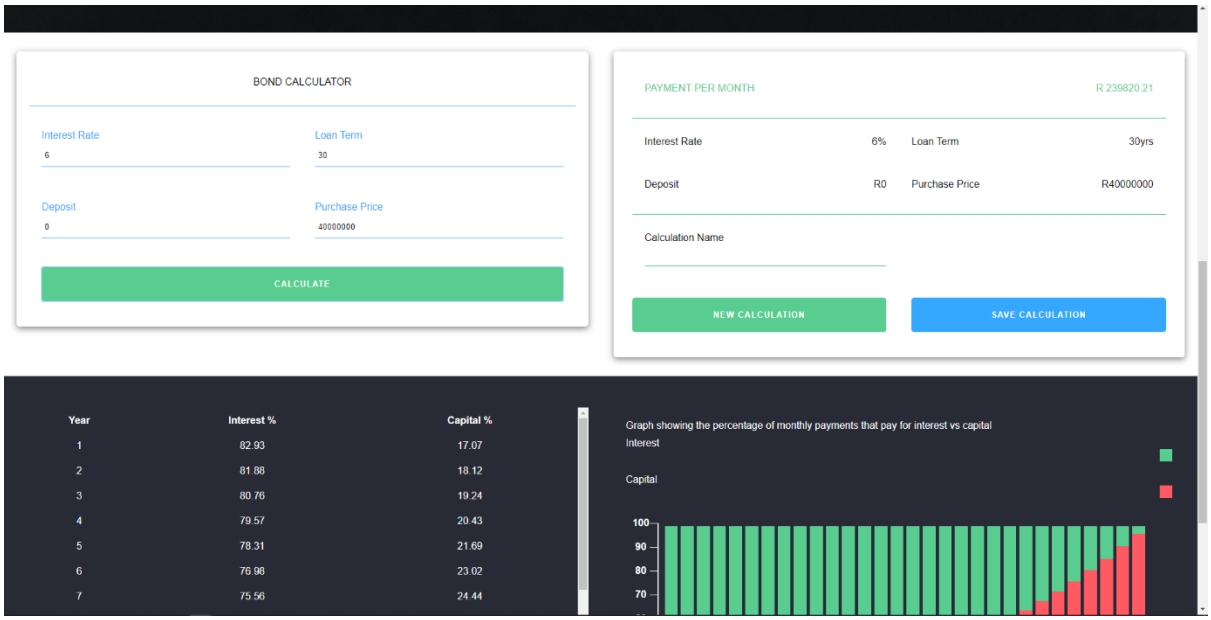
Due to time constraints some breakpoints were not accounted for. The webpage should resemble the images below

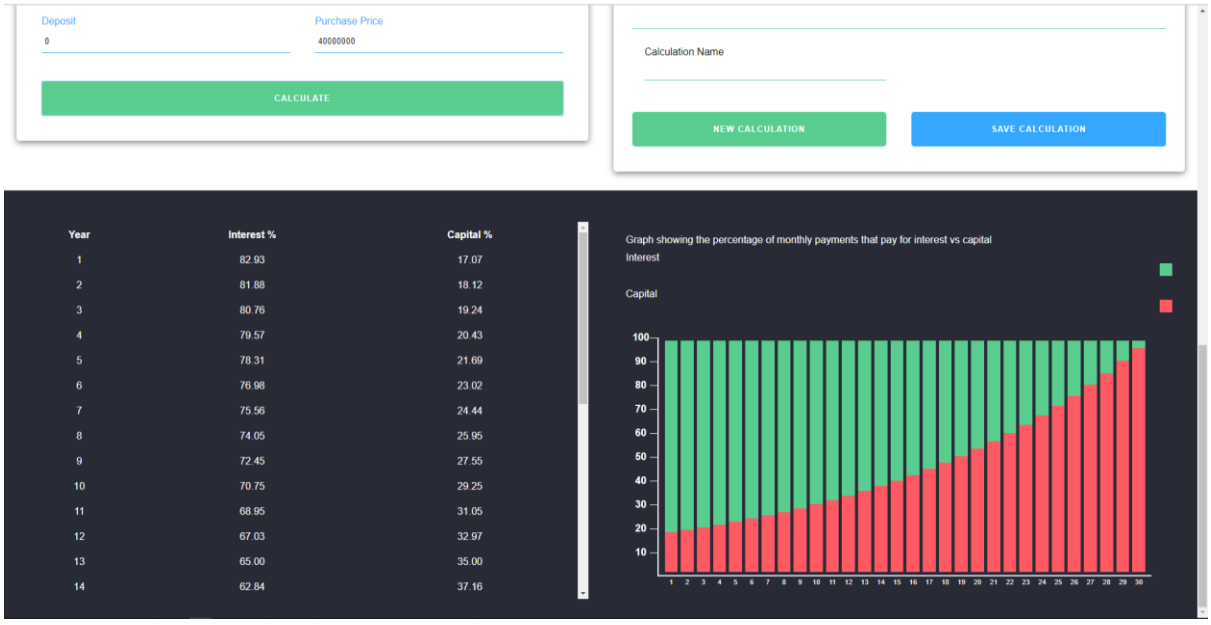


(pre calculation)

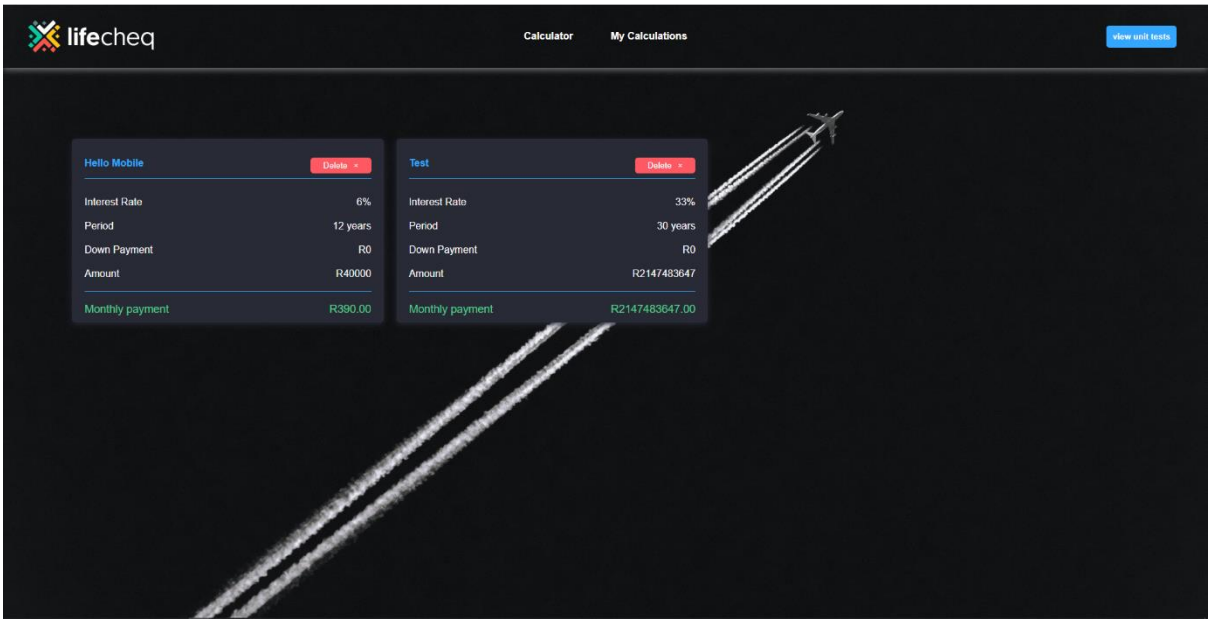


(post calculation 1 of 2)





(post calculation 2 of 2)



My calculations page with 2 calculations shown

SETTING UP THE ENVIRONMENT TO RUN THE CODE LOCALLY:

To run this code locally you must have the following

- XAMPP

Run Instructions:

Once XAMPP is installed:

- Paste code folder in the htdocs folder located inside the main XAMPP folder
- Open XAMPP control panel
- Start Apache web server
- Paste the users folder in the data folder inside the mysql folder (C:\xampp\mysql\data)
- Start MySQL from control panel
- Open browser and type in url http://localhost/BondApp/Bond-Calculator/?page=calculator

VISIT THE WEBSITE:

http://bondcalculator-com.stackstaging.com/?page=calculator