

# Code Challenge

---


A Small Browser-based app for pre-qualifying a potential user for a loan.

---

---

---

---



## Overview

Implement a small browser-based application for pre-qualifying a potential user for a loan.

## Must haves

1. Build instructions - Include a simple README or other file giving clear, concise instructions on how to build/run your code
2. Repeatable builds - All of engineers should be able to take your submission and instructions and run them without any major, obvious issues
3. Version history - Version control and history is an important part of our process and we expect it to be a part of yours too.  
— use GIT

Nice to haves— recommended if you have time:

1. Design docs - Explain your thinking clearly in an overview document helps us and future readers (including yourself) better understand your thinking and your structure.
2. Comprehensive Testing - We don't expect complete test coverage, but you should provide some tests for key, critical areas. This helps us see your testing strategies and what you think is critical.
3. Comprehensive Error Handling - Understanding the places where errors can be thrown and how to deal with them is an important part of this task, but we again understand that doing this completely could be time consuming. Feel free to detail errors through comments or other similar means to demonstrate your understanding.

## Gotchas

To be clear, many of the following issues will either reflect poorly or result in a no-pass from the interview team:

1. Unstructured code/Single commit - We expect any results to be reflective of production quality, maintainable code. Writing the entirety of the challenge in a single file, or committing it once is not reflective of that quality standard.

2. Scope Creep - An important part of professional engineering is producing a product that matches requirements. Submissions that go far beyond what is asked will reduce our ability to engage with your submission and is not a good use of your time.

3. Re-implementing standard libraries - Spending your limited time writing lists, http servers, security algorithms, etc. provided by the standard library is in general a bad idea and specifically not a good use of your time. It's okay to stick to the proven software

## Requirements

-----  
Your application should lead a user through the initial portion of qualifying for an auto loan. It should consist of a landing page to collect basic info about both the car they are interested in buying and their own financial situation. Based on this, you should be able to make a simulated network call that will redirect the user to either a new account creation page or a disqualification notice.

The styling should be simple, plain but deliberate.

### \* Landing Page

The initial page should show a simple form with inputs for the following field:

Auto Purchase Price (Currency)

Auto Make (Text)

Auto Model (Text)

User Estimated Yearly Income (Currency)

User Estimated Credit Score (Number from 300-850)

These fields are all required and should validate to type. Provide feedback to the user when they wrong or missing. There should be space for marketing copy (Lorum Ipsum) and controls for moving forward.

### \* New Account Page

If the api call does not return a disqualification message(see below), this page should have a simple account creation form including:

Username (Text)

Password (Text)

The username should be validated as an email and password should require more than 8 characters and a number or special character. Ensure the user types their password twice to validate their intent.

### \* Disqualification Page

Display a simple page with the disqualification message that comes from the api call as well as fake information to get in contact with a customer service. There should be no further way to get off this page or re-enter the information.

### \* API call

You should implement a mock fetch call for your backend communication. This call should have the same interface as the real fetch and return a promise wrapped response object. The response should return disqualify message (Lorum Ipsum is fine) if the purchase price is more than 1/5th of the income or their estimated credit is below 600. Otherwise it should return a positive qualification flag. A 'Bad Request' response should be returned for any auto purchase price above \$1,000,000.

## LANDING PAGE

- Show Inputs for:
  - Auto Purchase Price (Currency) ✓
  - Auto Make (Text) ✓
  - Auto Model (Text) ✓
  - User Estimated Yearly Income (Currency) ✓
  - User Estimated Credit Score (Number form (300-850) ✓
- Fields must be required
- Fields should validate type ✓
- Space for marketing (Lorem Ipsum)
- Control for moving forward ✓

## NEW ACCOUNT PAGE

- Username Input (Text) ✓
- Password Input (text) should be 2 ✓
- Username should validate an email ✓
- Password should require more than 8 characters and a number/special character
- ensure user types password in twice

## DISQUALIFICATION PAGE

- display a message that comes from the api call ✓
- display fake info to get in contact with customer service
- should be no way to get off this page & re-enter info ✓

## API CALL

- implement mock fetch call for backend communication ✓
- response should return disqualify message (Lorum Ipsum) ✓
- show disqualify message if purchase price is more than 1/5th of income or credit is below 600 ✓
- A `Bad Request` response should be returned for any auto purchase about \$1,000,000

User Story

# Landing Page

The landing page will collect basic information about the user by having the user input their information into a form.

Information on Form

- Auto Purchase Price (currency)
- Auto Make (Text)
- Auto Model (Text)
- User Estimated Yearly Income (currency)
- User Estimated Credit Score (Numbers from 300-850)

// These fields are required and must validate type

// If type is wrong or missing provide feedback to the user

// should be space for marketing copy (Lorem Ipsum)

// a button to move to the next page

# New Account Page

If the Api Call does not return a disqualification message, this page should have a simple creation form including:

## Form Info

- Username (Text)
- Password (Text)

// User name should be validated as an Email

// Password should be more than 8 characters long & a number or Special Character

// User must type Password twice - to validate their intent

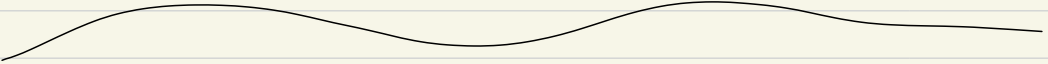
# Disqualification Page

## Page Should Show

- Disqualification message that comes from the api call
- Fake information to call customer Service. → From api
- They should not be able to get off this page to re-enter the information



# API call

- implement a mock fetch call for your backend communication
  - The call should interface as a real fetch & return a promise wrapped response object
  - Response should return DA message - if the purchase price is more than  $\frac{1}{5}$ th of the income or their estimated credit is below 600
  - Otherwise should return a Positive Qualification Flag → Send them to new account page
  - A 'Bad response' should be returned for any auto purchase price above \$1,000,000
- 

## The Api call needs to do 3 things

1. Should Return the disqualify message (Lorum Ipsum)

- we will return this call if: the purchase Price is more than  $\frac{1}{5}$  of the Income or their estimated credit score is below 600

- Otherwise it should return a positive qualification flag

- A 'Bad Request' response should be returned if the auto purchase price is above \$1,000,000

<https://www.youtube.com/watch?v=D9DdY2WmM-s>

# Youtube Video On Testing w/ React

Notes on the Video

Video #1 & 2

- We use Jest for Testing
- Mock is Fake "Something"

Video #3

# List of Todo's

Make My Development Background

Create the Landing Page

- create a form with inputs - being:

- Car Purchase Price (\$)

- The Car Make (Text)

- The Car Model (Text)

- User estimated yearly income (\$)

- user estimated credit score (number: 300-850)

- submit button

: The form should also:

- validate to type

- Provide Feedback if user has not filled in a field

- Provide Feedback if a user has filled in wrong information in a field

- Grab user inputs

- move state to App component

- Find out if the Purchase Price is more than  $\frac{1}{5}$  of the income ↱

- Also find out if their credit score is below 600

- Create a function that checks the income → Purchase Price || their credit score is above/below 600

- create mock API message via JSON

- On Start Up → Display: Form: True

Display: False

- Display The fail message for the loan form API Call → in Disqualify Component

- Create New Account Creation Page - Component

- Upon approval show New account Component & hide Form Page

## New Account Page

- create a user name input (must be email)

- create Password inputs (2)

- Password must be :

- more than 8 characters
    - must include a number || Special Character
    - user must type password in twice to validate their intent