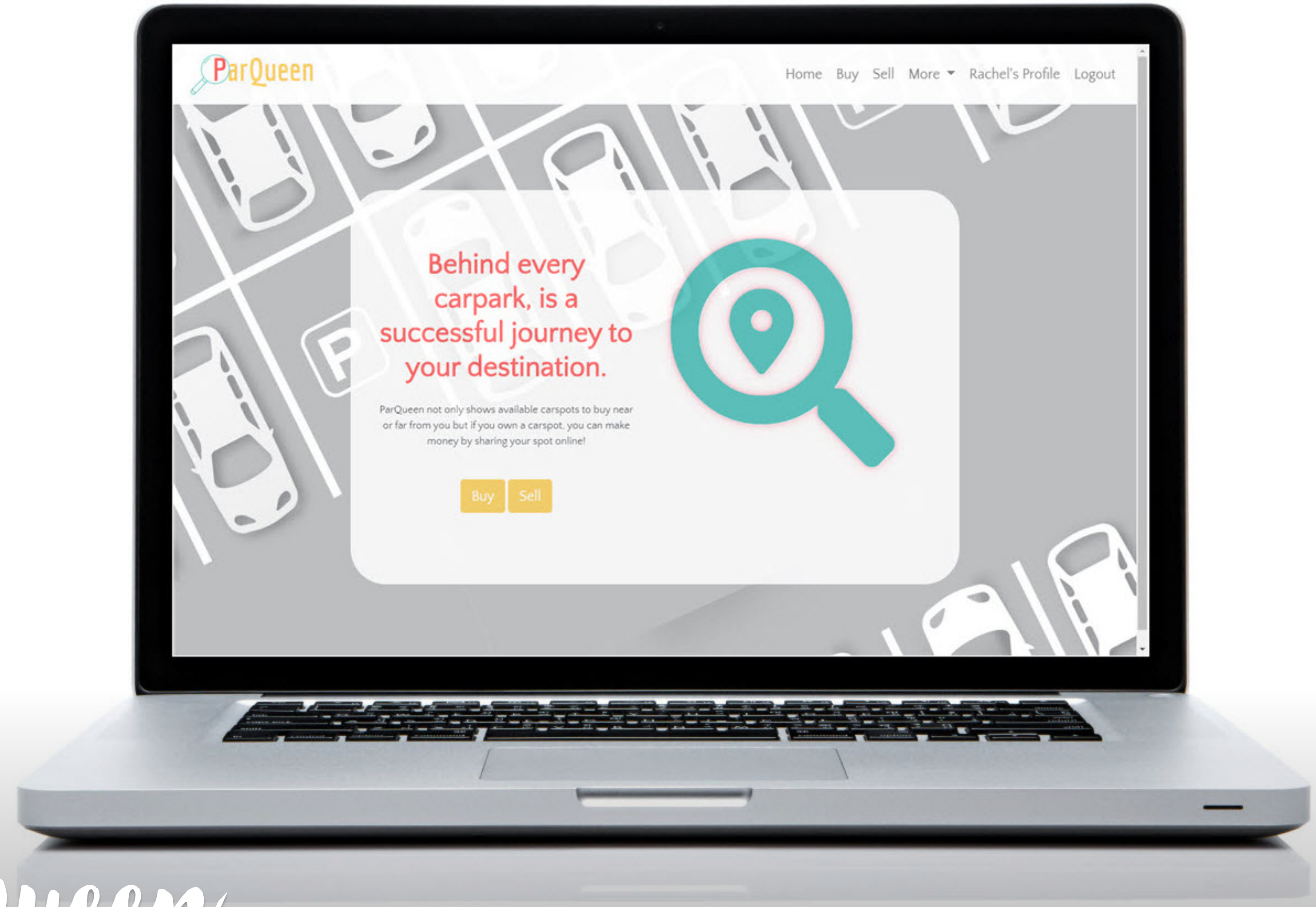


Behind every carpark, is a successful journey to your destination.

ParQueen not only shows available carspots to buy near or far from you but if you own a carspot, you can make money by sharing your spot online!

Parqueen

A MARKET PLACE APPLICATION

Why ParQueen?

The ParQueen Market place application is the best market place application to connect people to easily purchase car spots available in Australia.

This cloud-based market place solves a problem by enabling individuals and businesses to upload their available car spaces anywhere at anytime to sell.

ParQueen is the only application to use when selling a car spot or looking for a new one. No longer do people have to wait for periods of time when looking for a carspot, whether it be a regular visit to the shopping centre, a monday to friday commute to the city for your staff, consistent travelling to the airport or your favourite beach in summer. ParQueen can help you save time and money by providing you with an easy and convenient method for searching or selling your carspot.

As the population grows, there are more cars on the road and a bigger demand on where to park. Join the community and list your carspot today, no matter if it's street parking, residential, Airport or a business there is always a need.

View ParQueen

Target Audience

The are two types of target audience for ParQueen, for people and businesses who:

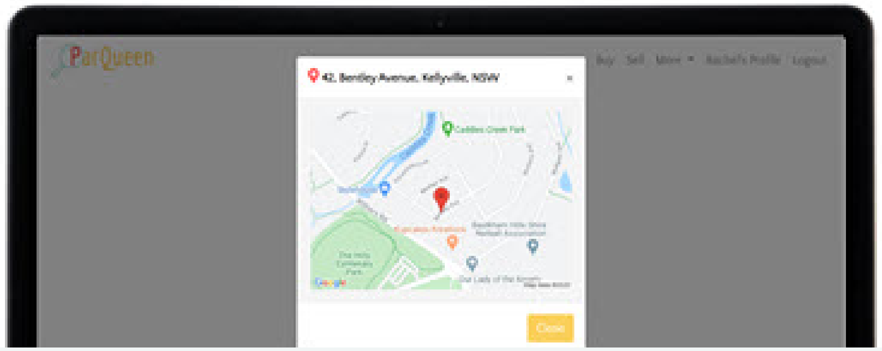
1. Need an online presence to reach a greater audience in order to sell their carspots quickly; and
2. An easy and convenient application to find and purchase a carspot for every need

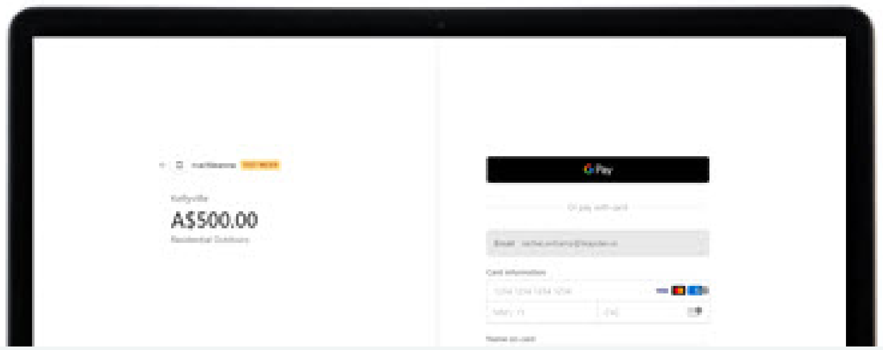
🟥 #ff5959

🟨 #facf5a

🟩 #49beb7

Style





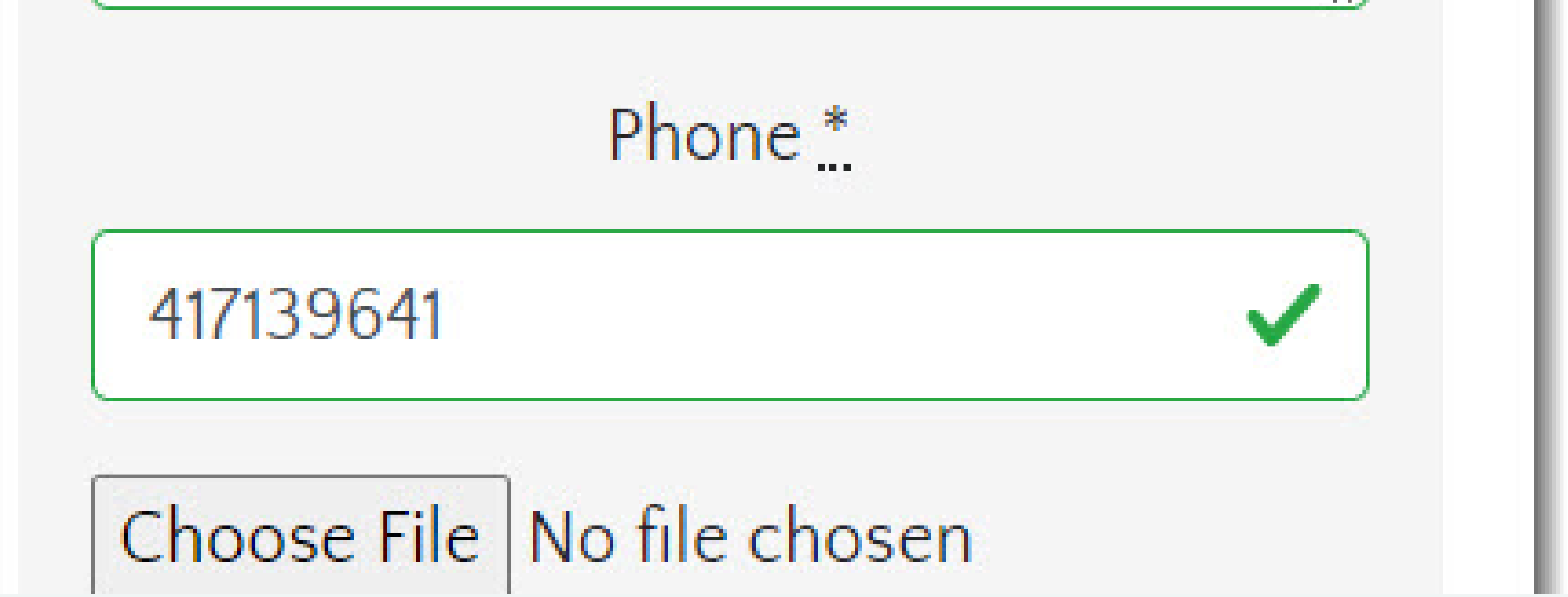
Functionality

With an easy to use interface, users can purchase or sell a carspot in the matter of minutes.

ParQueen has designed the app to be quick and efficient for buying and selling, with fast sign up forms and limited information required for the profile. From there, users can search for a carspot by suburb or view all - making it easier and quicker for them to locate their dream spot.

The application also offers a secure trusted platform for payment processing called Stripe so people and businesses can get paid faster.

Features



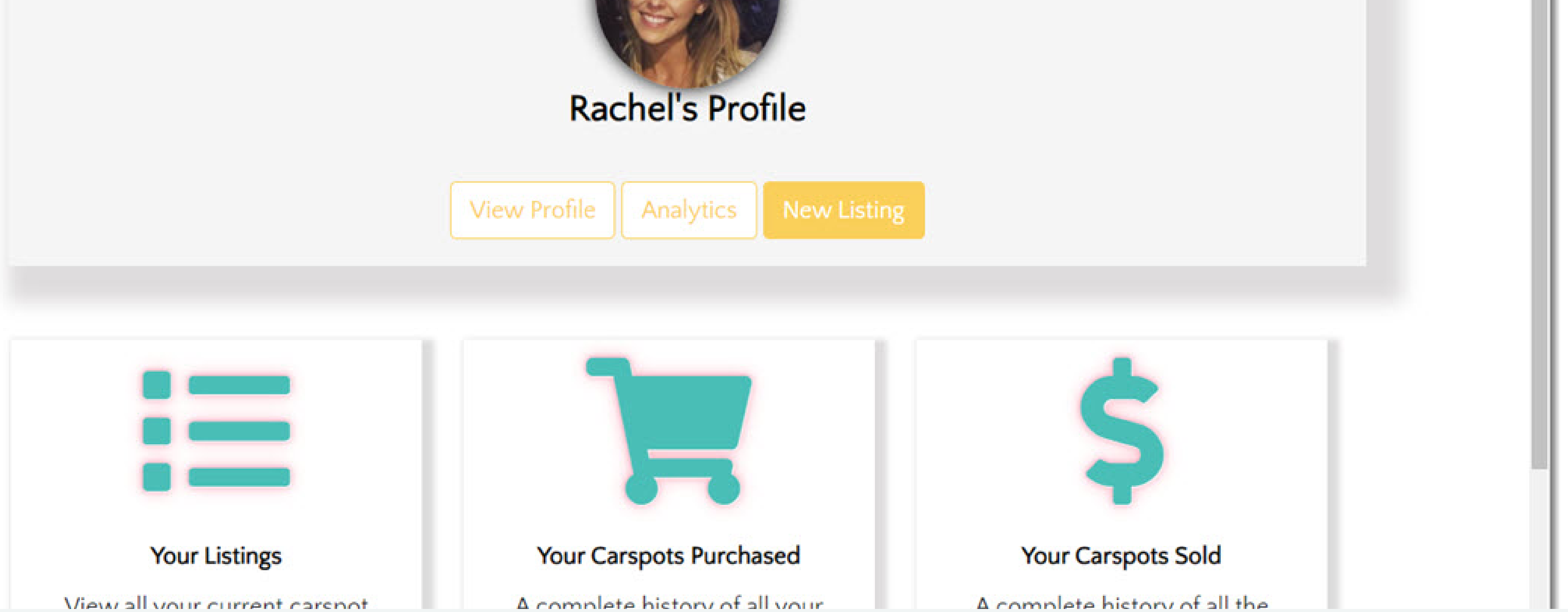
Phone *

417139641

Choose File No file chosen

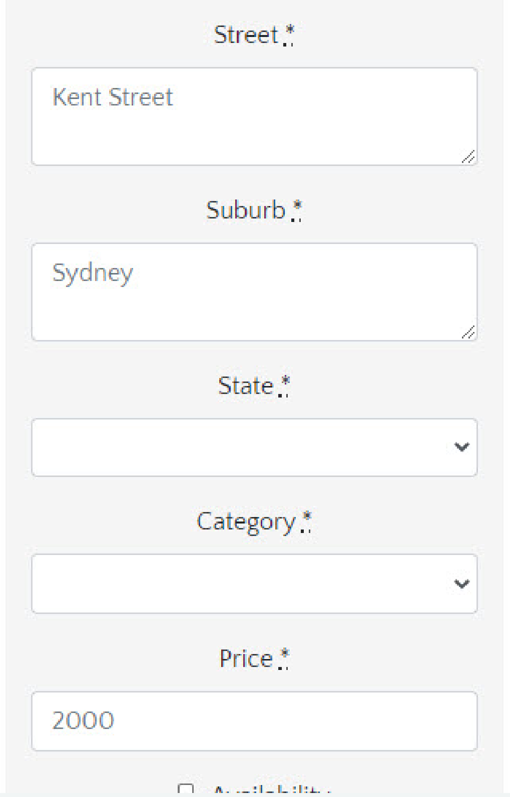
Secure Login

- ParQueen requires users to have an account should they wish to sell or buy a carspot.
- Authorization inhibits other users from being able to access other profiles to modify their listings. This quality feature demonstrates that ParQueen is built to be user-friendly and secure for all users.



Profile

- Users have **access to their profile,** in which they can manage their details, view analytics, their available carspots and create new carspot listings.
- The **dashboard** can help users track and manage all carspot listings from the Profile dashboard. This dashboard allows users to view all their listings and manage them, see what they have purchased from ParQueen and what they have sold from ParQueen.
- **Profile Analytics** allows Users to see a quick overview of of total Listings, Carspots Purchased by them and Carspots Sold by them.
- **Profile Modals** allows Users to never have to leave their profile to view their listings, carspots sold or purchased as all the information is displayed in a pop-up modal on the profile page.
- Users can **Control** a listing if they are not ready to sell their carspot or for some reason they don't want to advertise it anymore, then the user can change the availability of the listing to no longer appear on the listing page.


Street :*

Kent Street

Suburb :*

Sydney

State :*

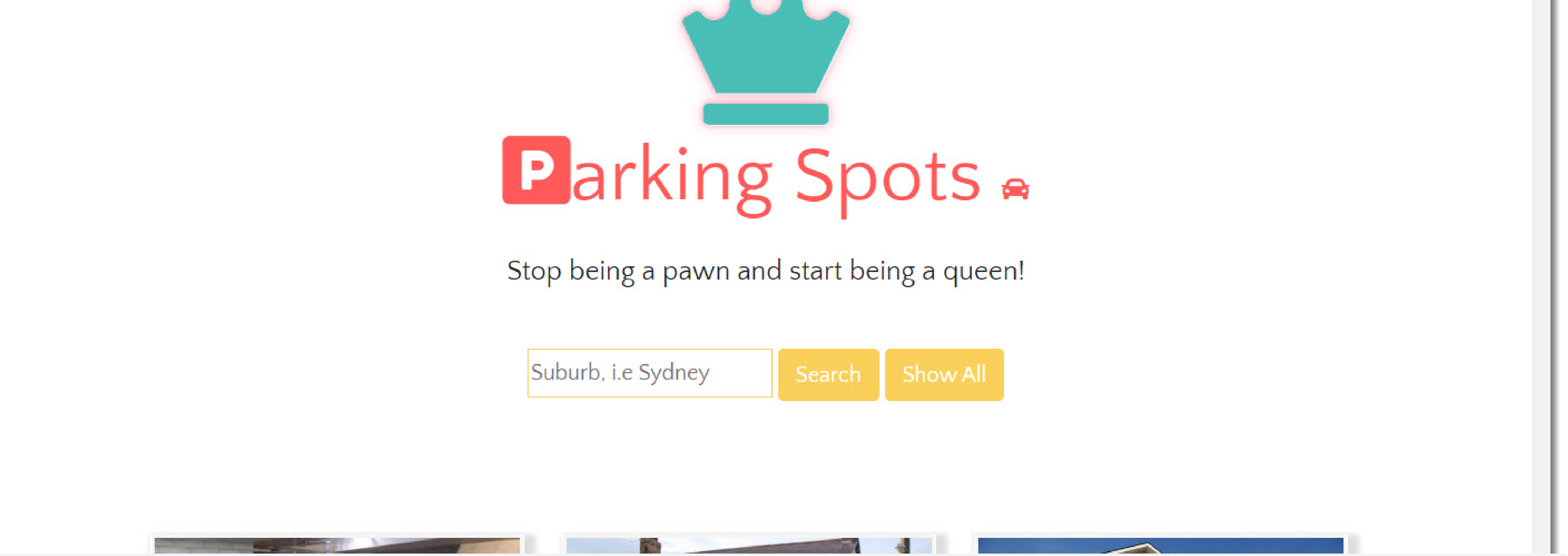
Category :*

Price :*

2000

Product & Profile Forms

- ParQueen offers easy to use profile forms and new product listing forms with minimal information required to get the listing up within minutes. Inbuilt functions such as dropdown options make the selection quicker.
- Users are only able to upload gif, png, jpg and jpeg files

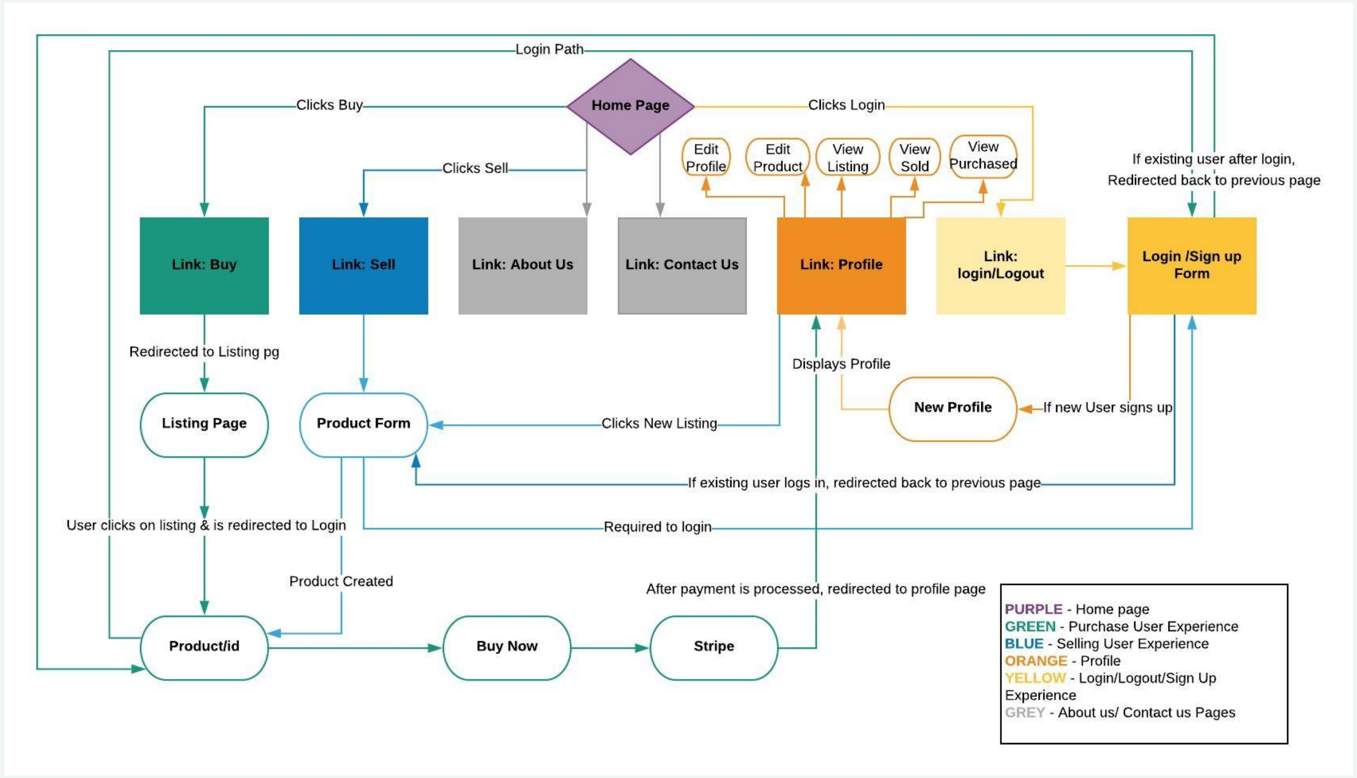

Parking Spots

Stop being a pawn and start being a queen!

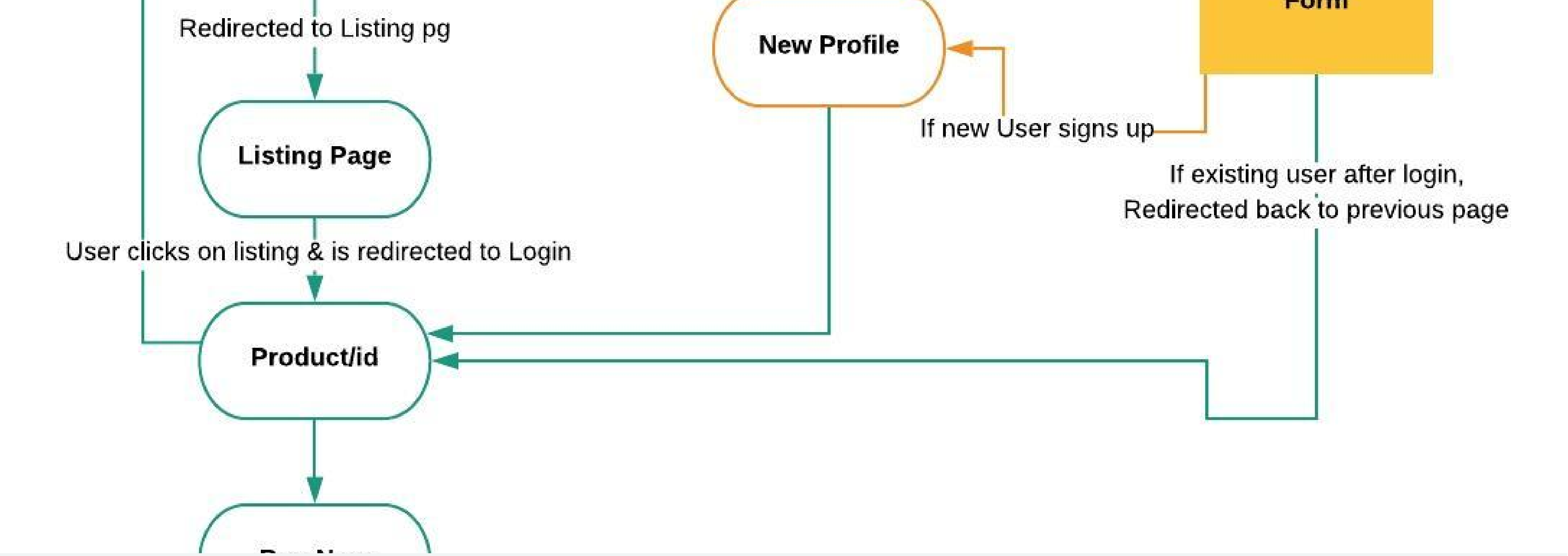
Listings

- Once a carspot is listed, it will display automatically on the listings page.
- With an **inbuilt search method**, people can easily find a car located within or 20km around the desired location by entering in the suburb.
- **Google Maps Integration** allows customers to review where the carspot is exactly situated. This provides a seamless user experience.
- ParQueen securely stores and protects images from profiles and listings on **AWS**
- ParQueen is integrated with **Stripe** allowing users the flexibility and security when making payments online. In return, people selling a car will receive their money faster.

User Experience



There are two types of user experience. The customer and the vendor.


Redirected to Listing pg
Listing Page
User clicks on listing & is redirected to Login
Product/id
New Profile
Form
If new User signs up
If existing user after login,
Redirected back to previous page

Customer Experience

- Customer is on home page
- Customer clicks 'Buy' from Nav bar or button on home page
- User is redirected to the Listing page where they are presented with the carspots available or they can search by suburb
- User clicks on 'Details' of a particular listing and is then re-directed to login page:

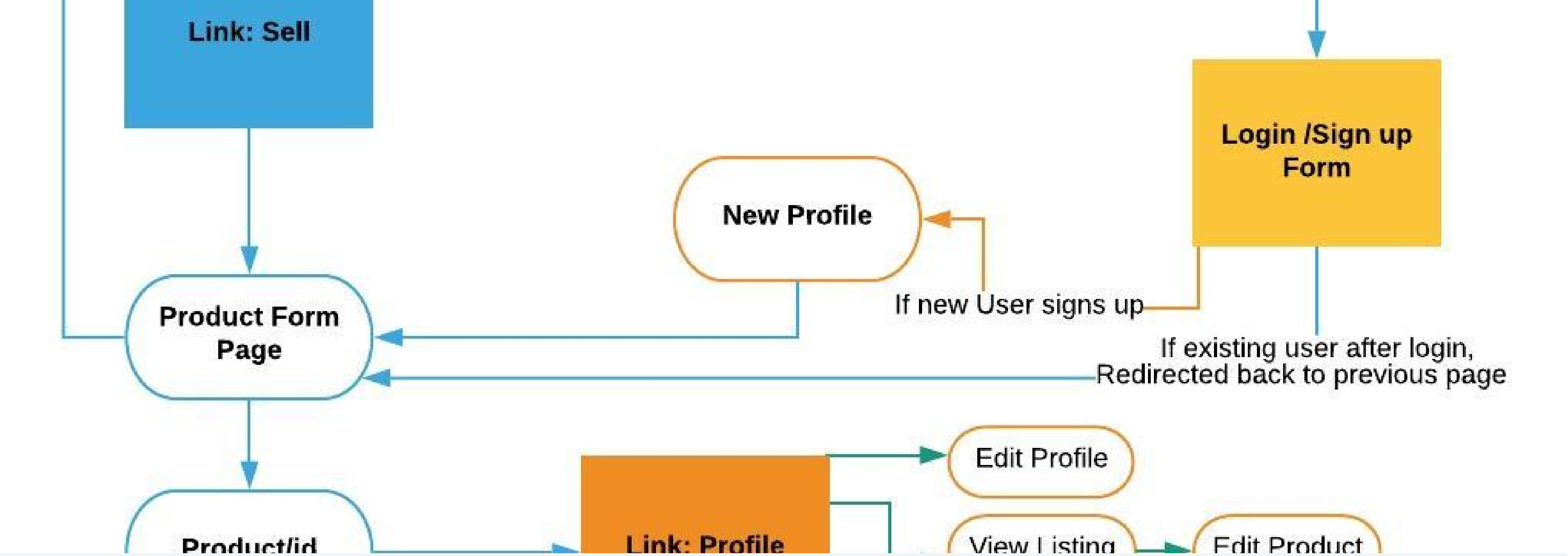
If existing user logs in:

- Redirected to that ProductID to view listing
- User clicks 'Buy'
- Redirected to stripe to pay

- Upon successful payment, user is re-directed back to profile to view their purchased products

If new user logs in :

- Redirected to create a new profile
- Upon profile save, user is re-directed to that previous ProductID
- User clicks 'Buy'
- Redirected to stripe to pay
- Upon successful payment, user is re-directed back to profile to view their purchased products



Vendor Experience

- Vendor is on home page
- Vendor clicks 'Sell' from Nav bar or button on home page
- Vendor is re-directed to the login page:

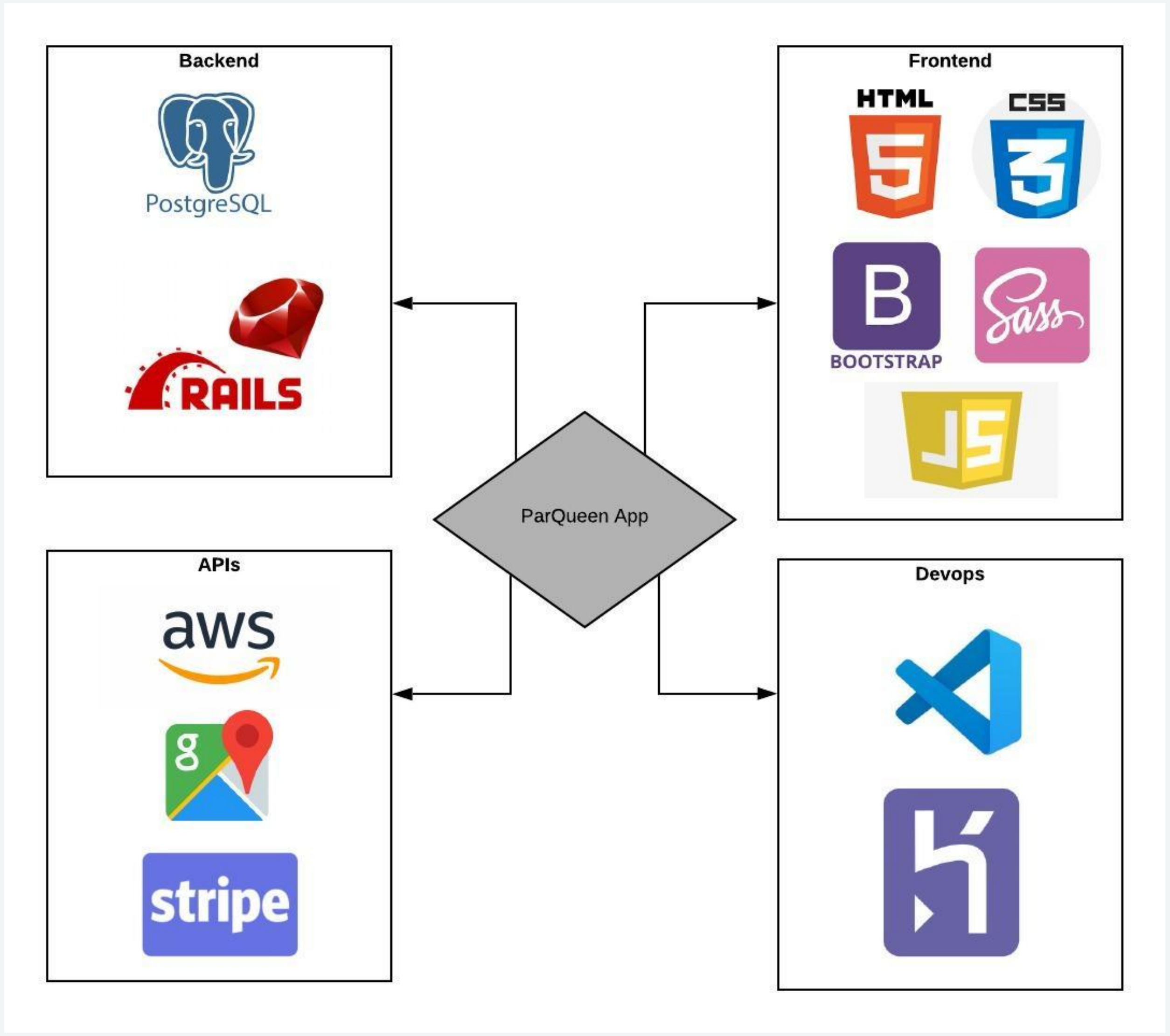
If existing user logs in:

- Redirected to Sell Carspot form
- Upon saving, vendor is redirected to the ProductID to view listing
- Vendor can access profile to view listings, add, edit, delete and see what has been sold

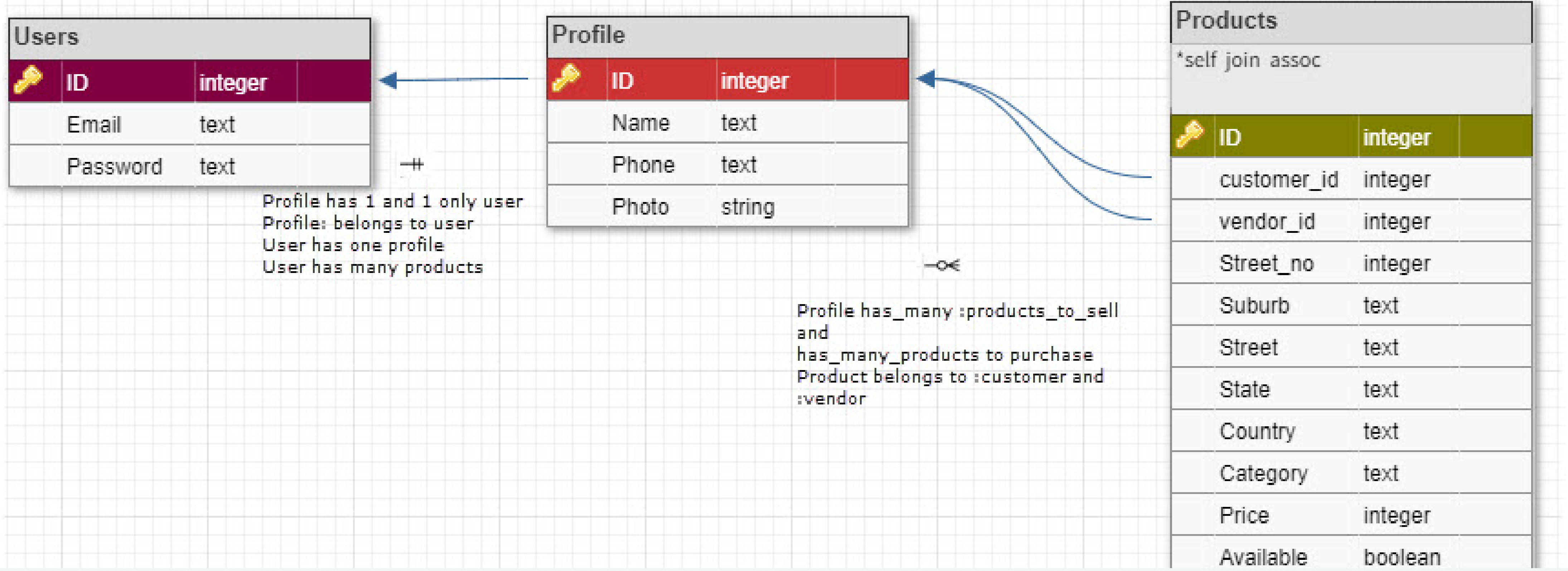
If new user logs in :

- Redirected to create a new profile
- Upon profile save, vendor is redirected to Sell Carspot form
- Upon saving, vendor is redirected to the ProductID to view listing
- Vendor can access profile to view listings, add, edit, delete and see what has been sold

Tech Stack



Front-end: HTML5, CSS3, SCSS, Bootstrap and JavaScript **| Back-end:** PostgreSQL, Ruby 2.7.1 and Ruby on Rails 6.0.3.2. **| DevOps:** Git, Github, Visual Studio Code and Heroku **| APIs:** Google Geocoding API, Stripe and AWS S3



Users

🔑	ID	integer
	Email	text
	Password	text

Profile

🔑	ID	integer
	Name	text
	Phone	text
	Photo	string

Products

*self join assoc

🔑	ID	integer
	customer_id	integer
	vendor_id	integer
	Street_no	integer
	Suburb	text
	Street	text
	State	text
	Country	text
	Category	text
	Price	integer
	Available	boolean

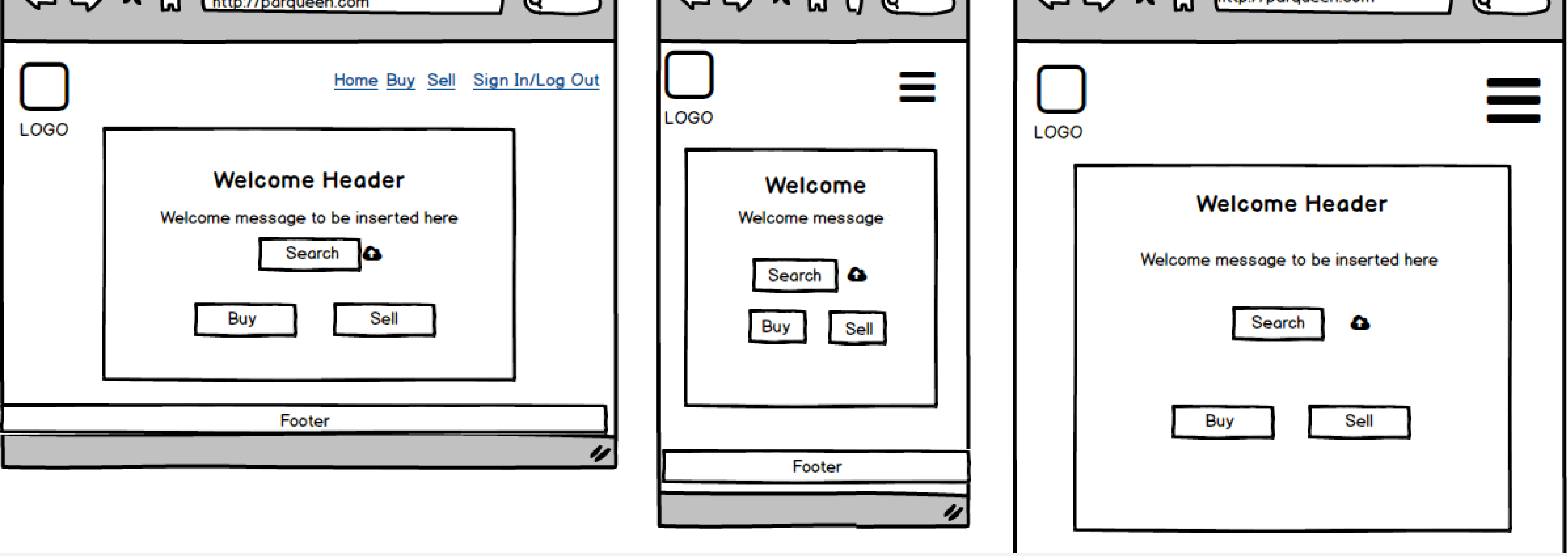
Profile has 1 and 1 only user
Profile: belongs to user
User has one profile
User has many products

Profile has_many :products_to_sell
and
has_many_products to purchase
Product belongs to :customer and
:vendor

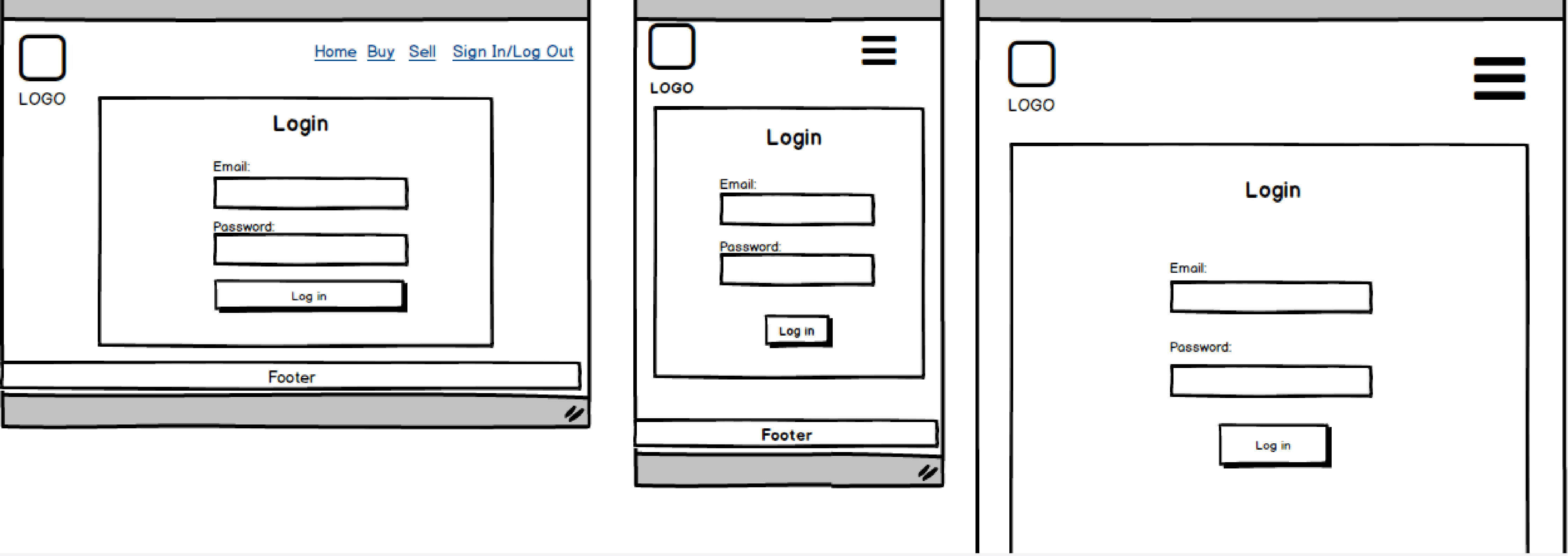
ERD

Simple ERD displaying a User table that has one profile and many products. The Profile table belongs to the user and has many products to sell and products to purchase. The Products table belongs to the customer and the vendor on the Profile table. This is achievable by implementing self-join association.

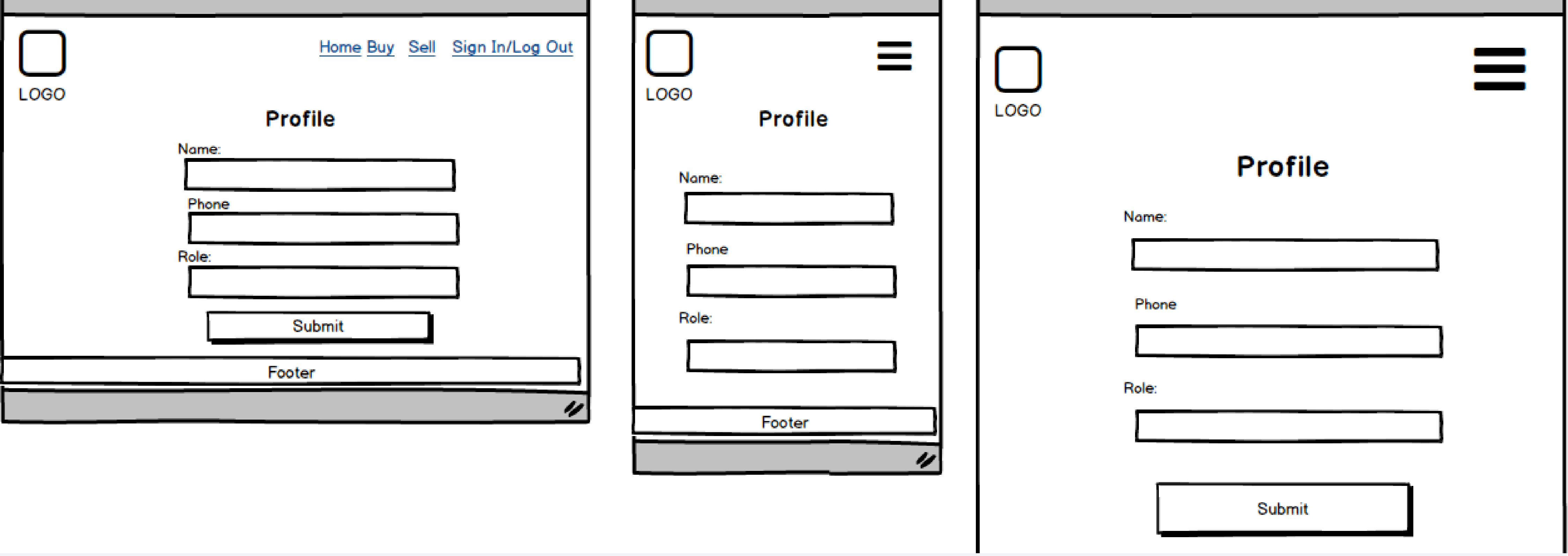
Wireframes


http://parqueen.com
Home
Buy
Sell
Sign In/Log Out
LOGO
Welcome Header
Welcome message to be inserted here
Search
Buy
Sell
Footer
LOGO
Welcome
Welcome message
Search
Buy
Sell
Footer
LOGO
Welcome Header
Welcome message to be inserted here
Search
Buy
Sell

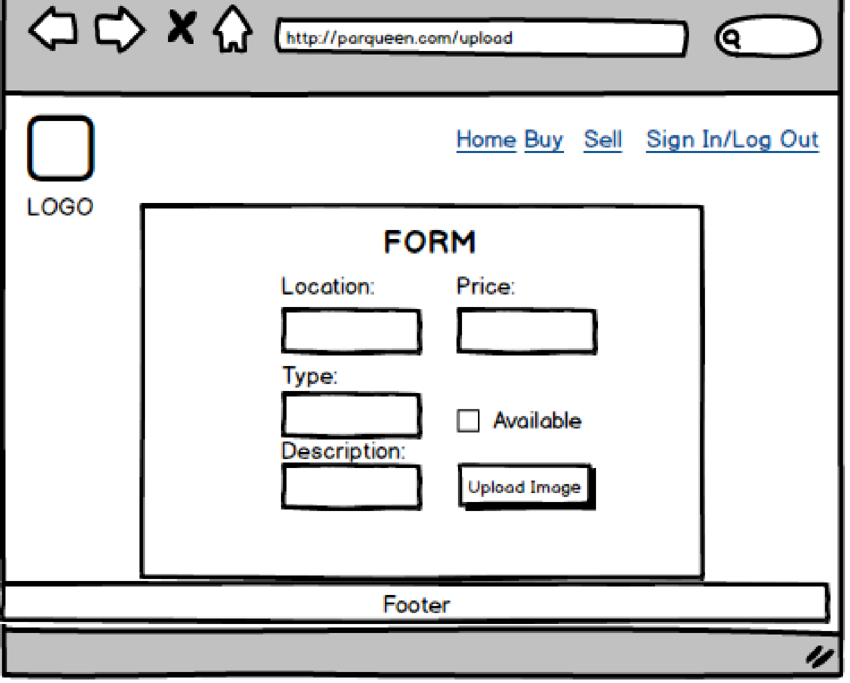
Home Page

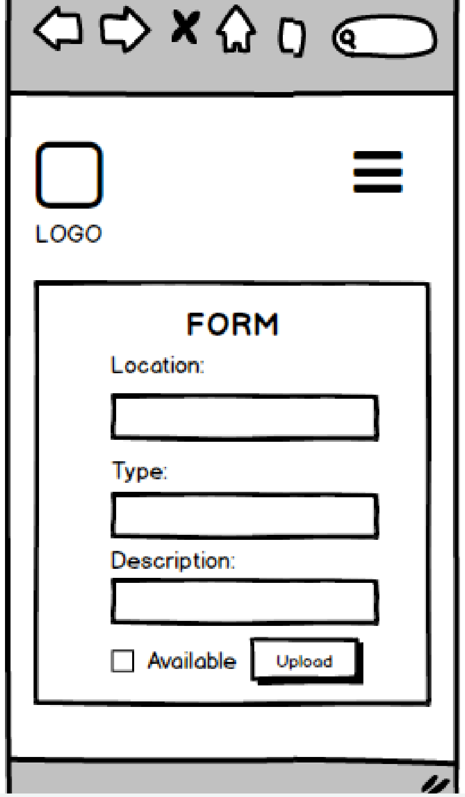

LOGO
Home
Buy
Sell
Sign In/Log Out
Login
Email:
Password:
Log in
Footer
LOGO
Login
Email:
Password:
Log in
Footer
LOGO
Login
Email:
Password:
Log in

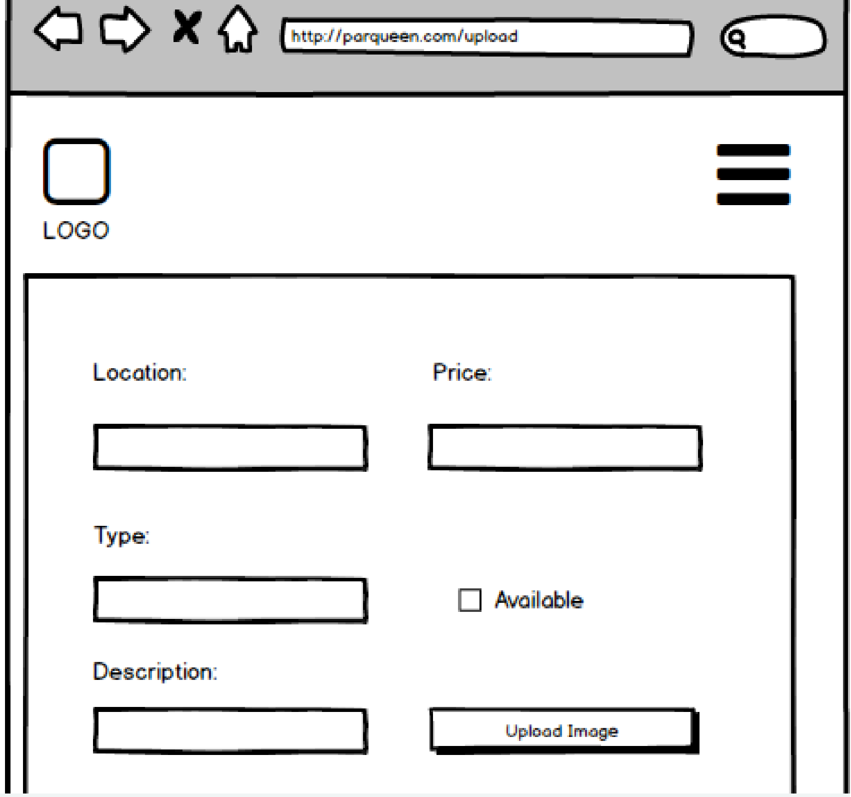
Sign-in



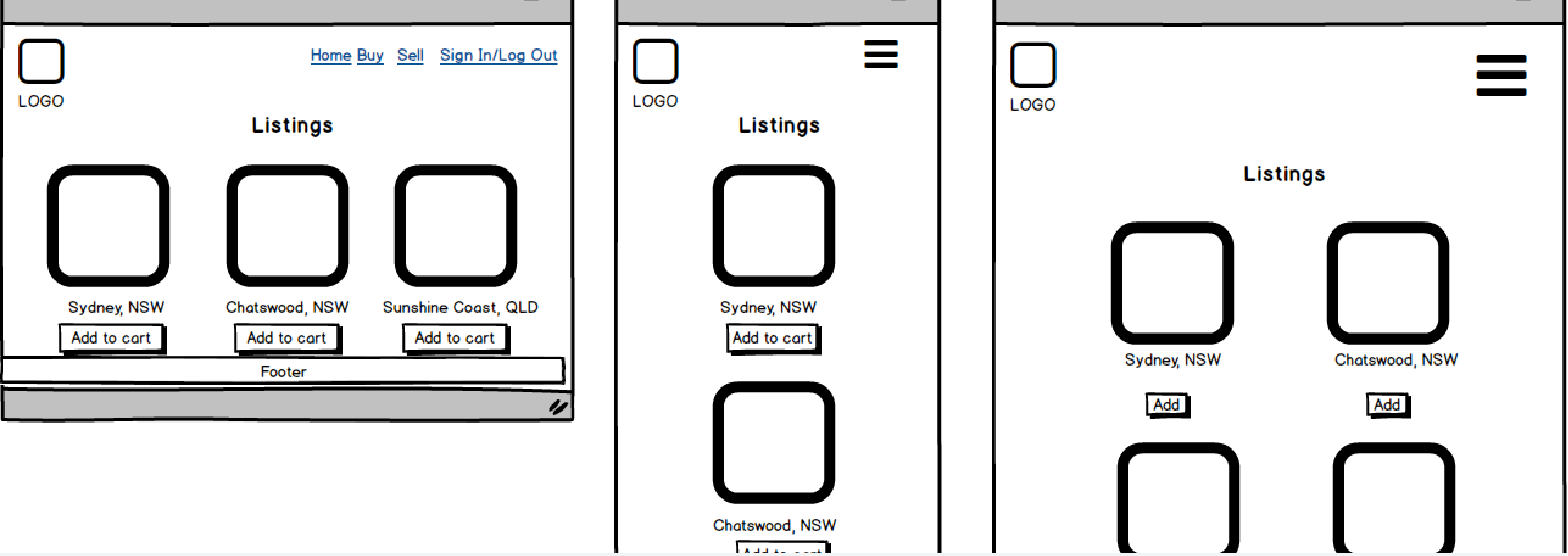
Profile Form



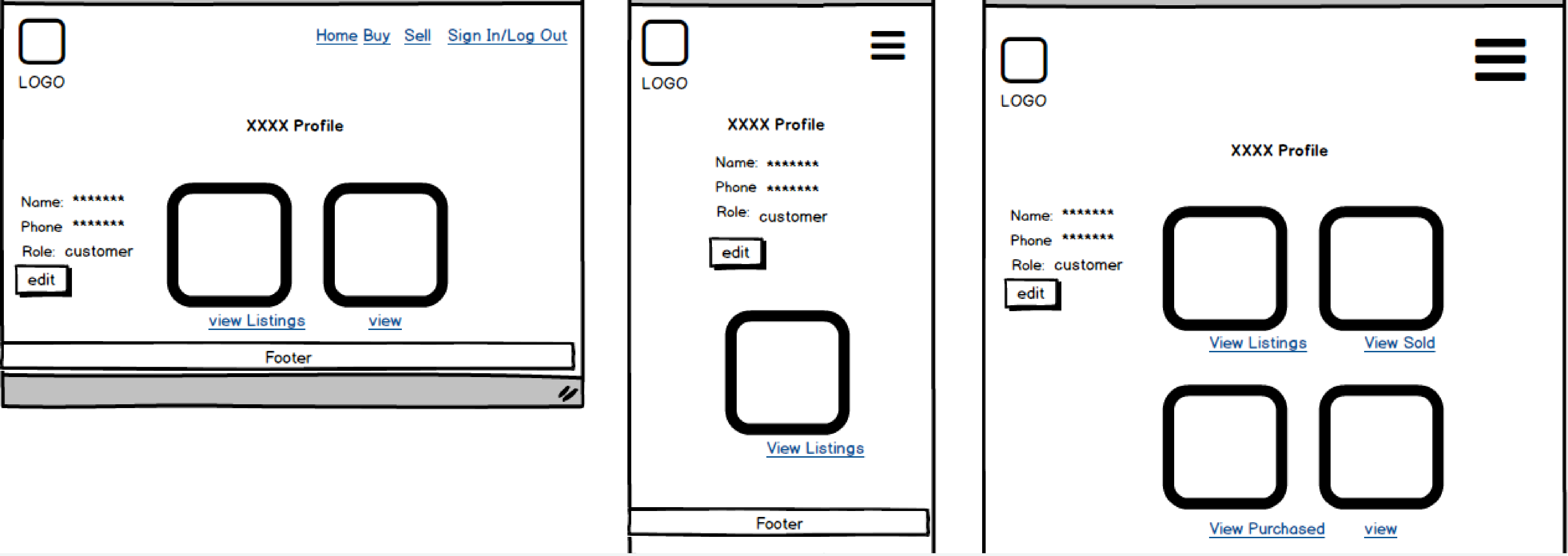




Upload Car Form

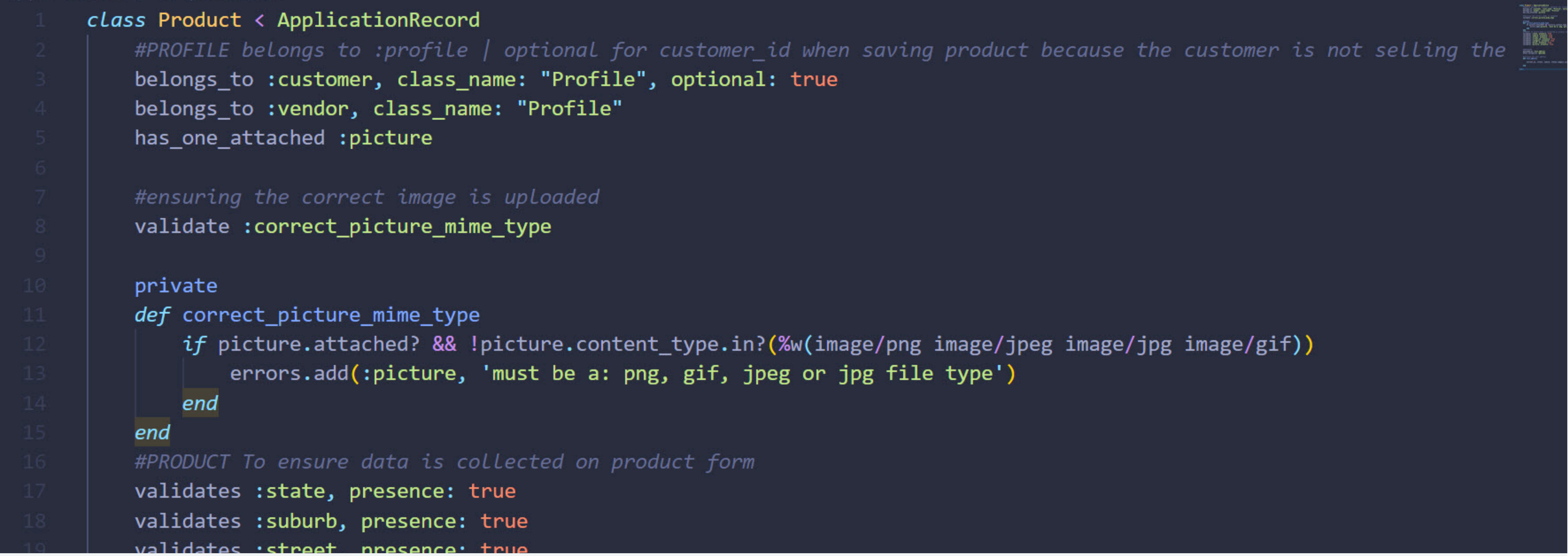


Listing Page



User Profile

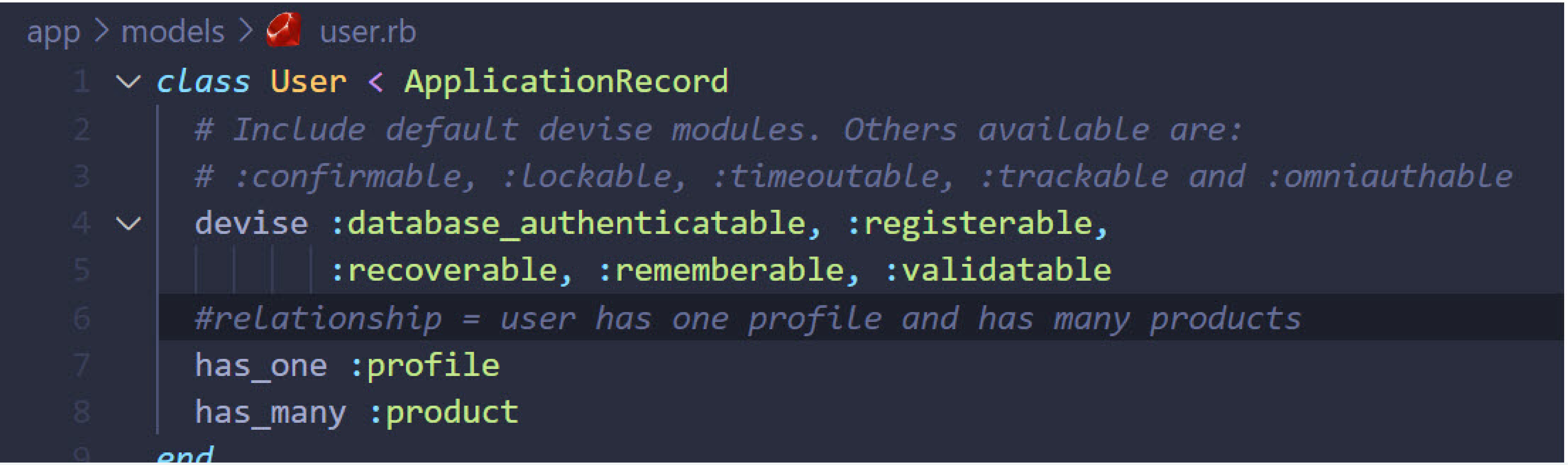
Model Relationships



```ruby
class Product < ApplicationRecord
    #PROFILE belongs to :profile | optional for customer_id when saving product because the customer is not selling the
    belongs_to :customer, class_name: "Profile", optional: true
    belongs_to :vendor, class_name: "Profile"
    has_one_attached :picture

    #ensuring the correct image is uploaded
    validate :correct_picture_mime_type

    private
    def correct_picture_mime_type
        if picture.attached? && !picture.content_type.in?(%w(image/png image/jpeg image/jpg image/gif))
            errors.add(:picture, 'must be a: png, gif, jpeg or jpg file type')
        end
    end
    #PRODUCT To ensure data is collected on product form
    validates :state, presence: true
    validates :suburb, presence: true
    validates :street, presence: true
```

For what may look like a complex application with a many Models, the ParQueen application is designed to not have complex Model relationships. Excluding the Application_record.rb Model, there are a total of three Models, these are:

- User
- Products
- Profile

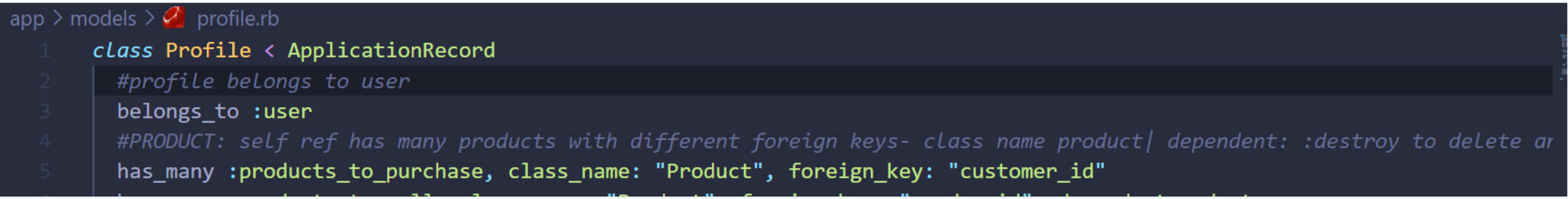


app > models > user.rb

```ruby
class User < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable
  #relationship = user has one profile and has many products
  has_one :profile
  has_many :product
end
```

User Model

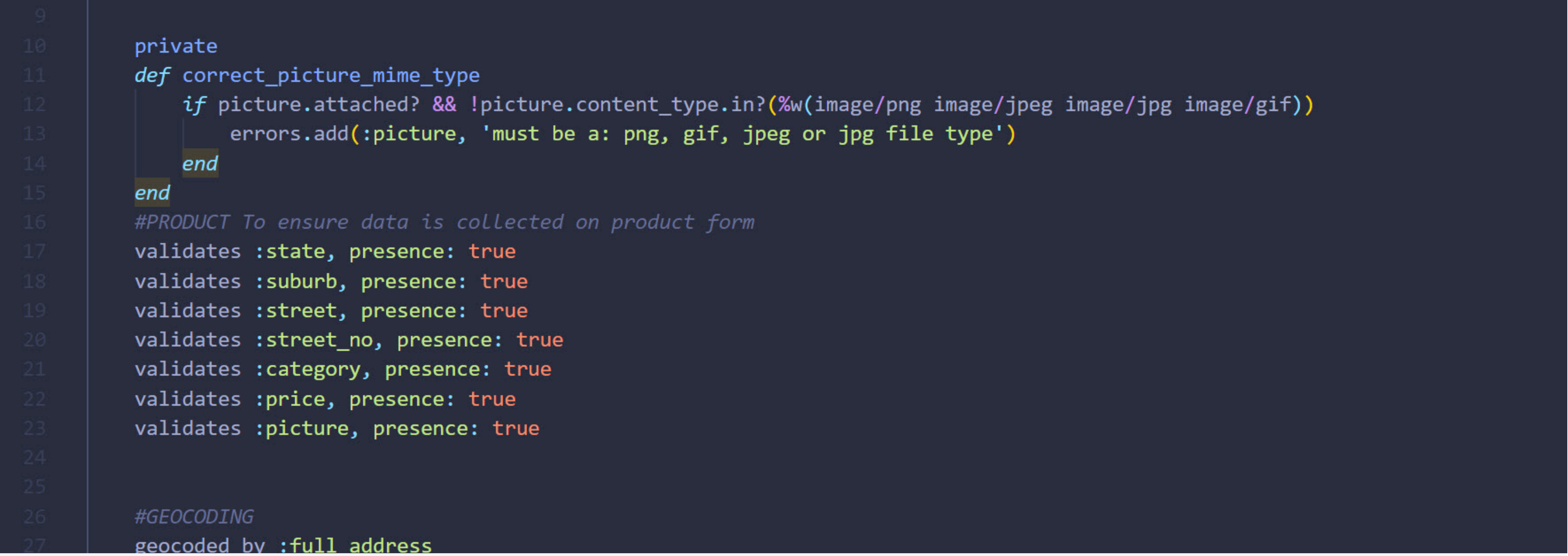
- The User model has one profile because you don't want the user to have many profiles; and
- The User model has many products because a user can have 0 to many products.
- In addition, this model also handles the Devise users gem for authentication when signing in to the application



```ruby
class Profile < ApplicationRecord
  #profile belongs to user
  belongs_to :user
  #PRODUCT: self ref has many products with different foreign keys- class name product| dependent: :destroy to delete ar
  has_many :products_to_purchase, class_name: "Product", foreign_key: "customer_id"
```

Profile Model

- The Profile model belongs to the user as there is only one user and profile shared between them;
- The Profile model has many products to sell via the Product model that corresponds to a vendor_id; and
- The Profile Model has many products to purchase via the Product model that corresponds to the customer_id
- This model also only accepts one attached picture and validates the file type and validates that all the fields are completed by the vendor/customer user



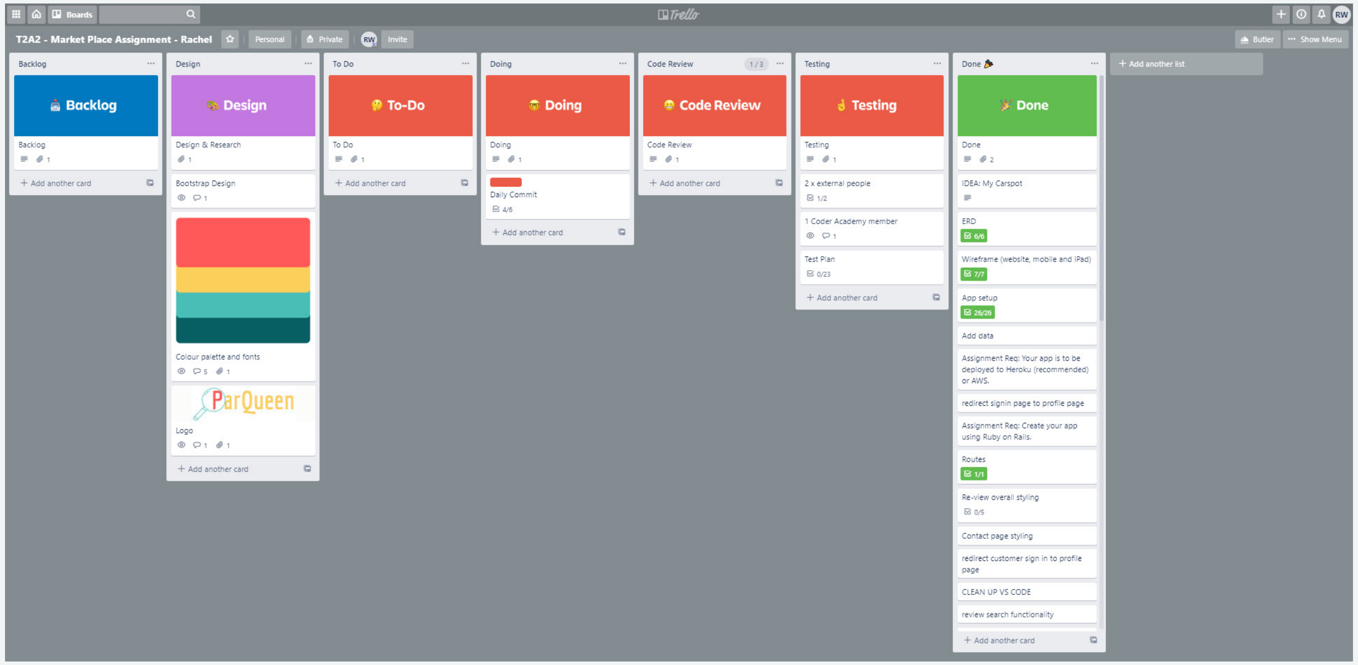
```
    private
    def correct_picture_mime_type
        if picture.attached? && !picture.content_type.in?(%w(image/png image/jpeg image/jpg image/gif))
            errors.add(:picture, 'must be a: png, gif, jpeg or jpg file type')
        end
    end
    #PRODUCT To ensure data is collected on product form
    validates :state, presence: true
    validates :suburb, presence: true
    validates :street, presence: true
    validates :street_no, presence: true
    validates :category, presence: true
    validates :price, presence: true
    validates :picture, presence: true


    #GEOCODING
    geocoded by :full address
```

Product Model

- The Product model belongs to the customer from the Profile model; and
- The Product model belongs to the vendor from the Profile model
- Customer is however optional because when creating a product, the customer does not yet exist until they have ordered a product, but the vendor is required upon selling and item
- This model also only accepts one attached picture and validates the file type and validates that all the fields are completed/filled by the vendor user
- This model also validates the address by using the gem geocoder to enable the feature of GoogleMaps, the address is joined together.

Task Allocation and Tracking

- ERD
- Wireframes
- Trello
- Top 5(or more) tasks on notepad



Backlog
- Backlog
- Backlog 1
- + Add another card

Design
- Design
- Design & Research 1
- Bootstrap Design 1
- Colour palette and fonts 5 1
- Logo 1 1
- + Add another card

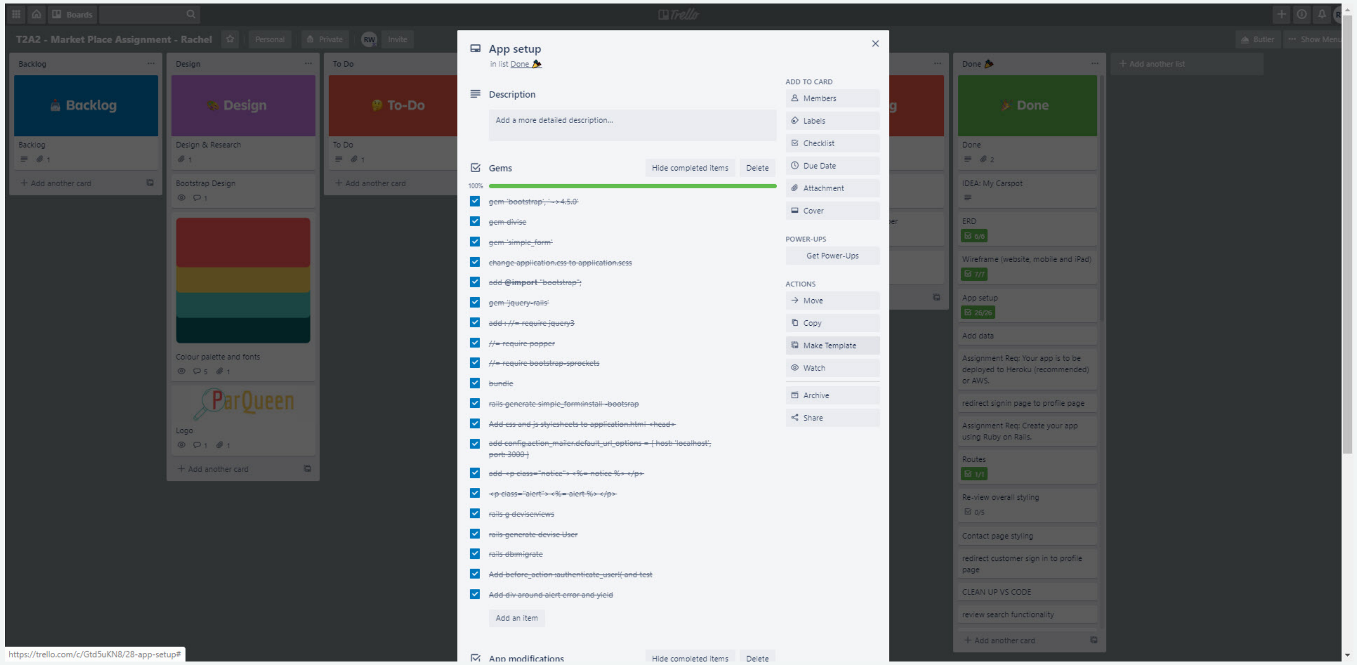
To Do
- To-Do
- To Do 1
- + Add another card

Doing
- Doing
- Doing 1
- Daily Commit 4/6
- + Add another card

Code Review 1 / 3
- Code Review
- Code Review 1
- + Add another card

Testing
- Testing
- Testing 1
- 2 x external people 1/2
- 1 Coder Academy member 1
- Test Plan 0/23
- + Add another card

Done 🎉
- Done
- Done 2
- IDEA: My Carspot
- ERD 6/6
- Wireframe (website, mobile and iPad) 7/7
- App setup 26/26
- Add data
- Assignment Req: Your app is to be deployed to Heroku (recommended) or AWS.
- redirect signin page to profile page
- Assignment Req: Create your app using Ruby on Rails.
- Routes 1/1
- Re-view overall styling 0/5
- Contact page styling
- redirect customer sign in to profile page
- CLEAN UP VS CODE
- review search functionality
- + Add another card



App setup
in list Done

Description

Add a more detailed description...

Gems

100%

- ✓ ~~gem 'bootstrap', '~> 4.5.0'~~
- ✓ ~~gem divise~~
- ✓ ~~gem 'simple_form'~~
- ✓ ~~change application.css to application.scss~~
- ✓ ~~add @import "bootstrap";~~
- ✓ ~~gem 'jquery-rails'~~
- ✓ ~~add : //= require jquery3~~
- ✓ ~~//= require popper~~
- ✓ ~~//= require bootstrap-sprockets~~
- ✓ ~~bundle~~
- ✓ ~~rails generate simple_form:install --bootsrap~~
- ✓ ~~Add css and js stylesheets to application.html <head>~~
- ✓ ~~add config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }~~
- ✓ ~~add <p class="notice"> <%= notice %> </p>~~
- ✓ ~~<p class="alert"> <%= alert %> </p>~~
- ✓ ~~rails g devise:views~~
- ✓ ~~rails generate devise User~~
- ✓ ~~rails db:migrate~~
- ✓ ~~Add before_action :authenticate_user!(and test~~
- ✓ ~~Add div around alert error and yield~~

Add an item

App modifications

Done

- Done
- IDEA: My Carspot
- ERD 6/6
- Wireframe (website, mobile and iPad) 7/7
- App setup 26/26
- Add data
- Assignment Req: Your app is to be deployed to Heroku (recommended) or AWS.
- redirect signin page to profile page
- Assignment Req: Create your app using Ruby on Rails.
- Routes 1/1
- Re-view overall styling 0/5
- Contact page styling
- redirect customer sign in to profile page
- CLEAN UP VS CODE
- review search functionality

https://trello.com/c/Gtd5uKN8/28-app-setup#

Lessons Learned

- Changing my complex ERD to be simplified by having self join association
- Current_user.profile ID when calling it in Controller and Views
- If and else for user_signed_in and current_user.profile
- APIs and credentials for Stripe integration, Google Maps and AWS
- Dropping tables & troubleshooting
- Rolling back & troubleshooting pending migrations
- Search functionality

- Dropdown on forms
- Validating file types
- Validating fields
- Number_to_currency ($), (,) and (.) = $2,000.00
- Re-directions
- After_sign_in_path_for_resource and storing previous_path
- Heroku errors and deployment via terminal

View ParQueen