

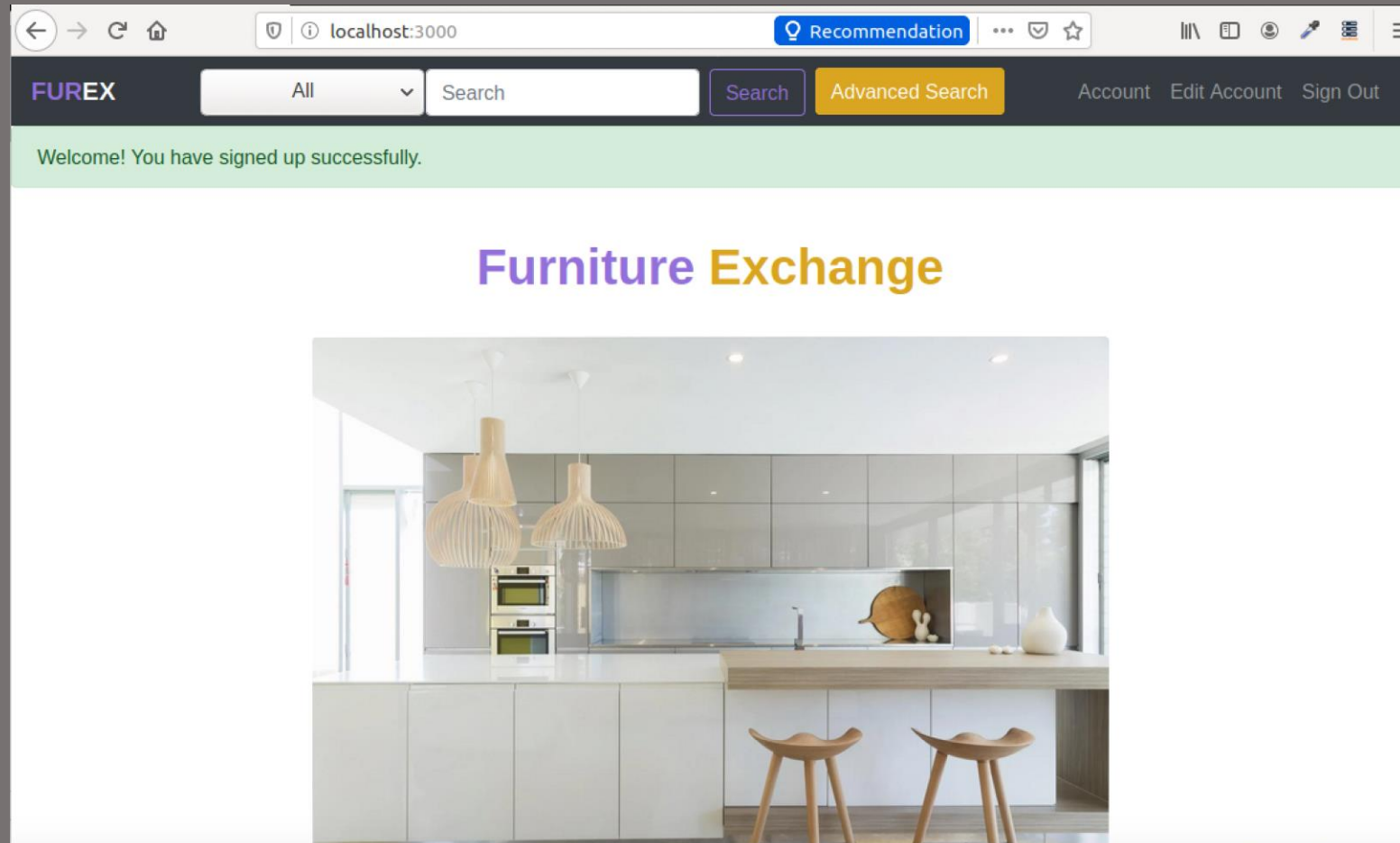
FUREX

Team 0011 1001 0000 0001

Amanda Cheng, Kevin Dong, Prachiti Garge, Troy Stein, Yifan Yao

FUREX – Furniture Exchange

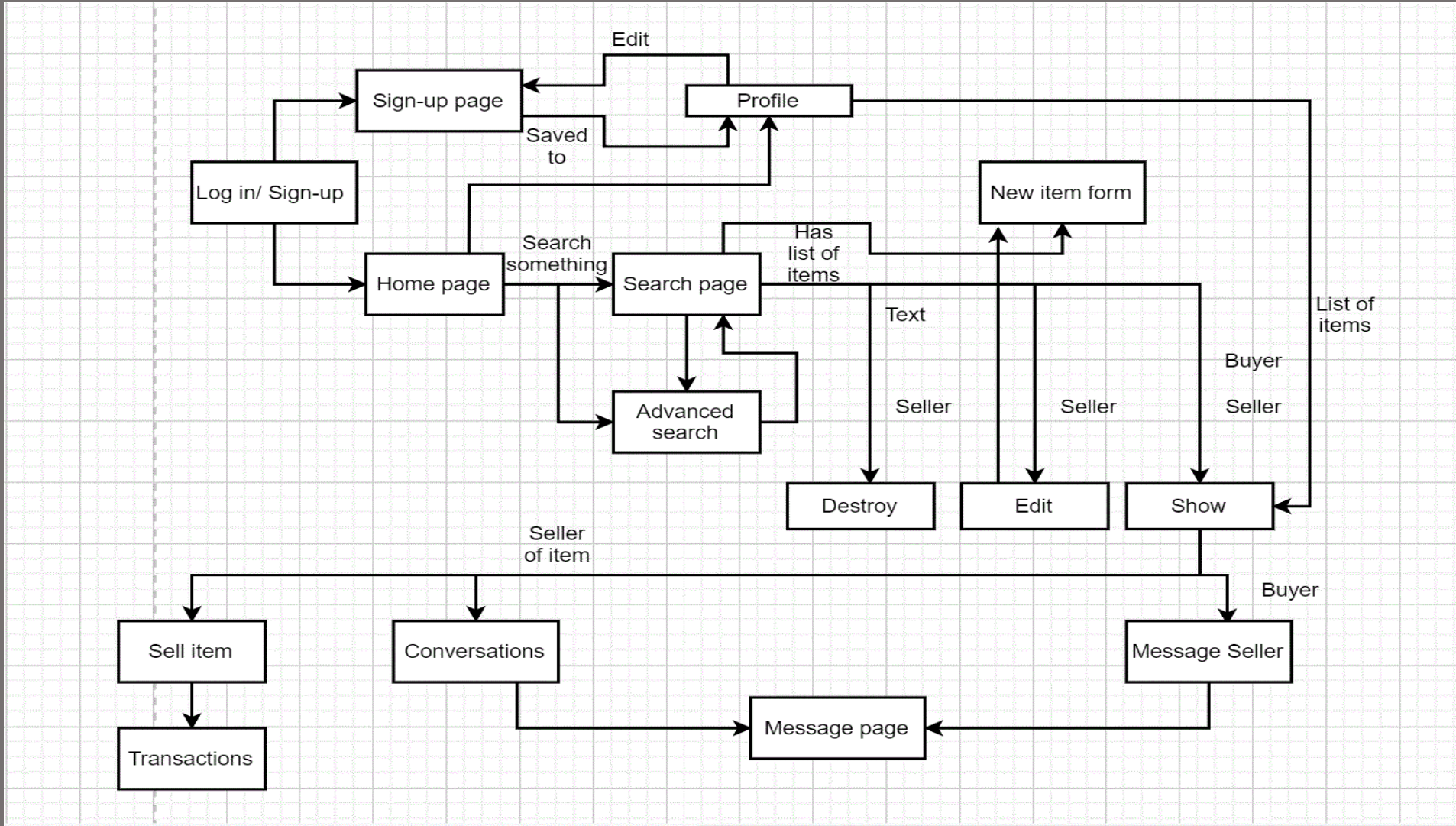
- E-commerce for used and non-used furniture around OSU campus
- Platform for mainly OSU students looking for furniture

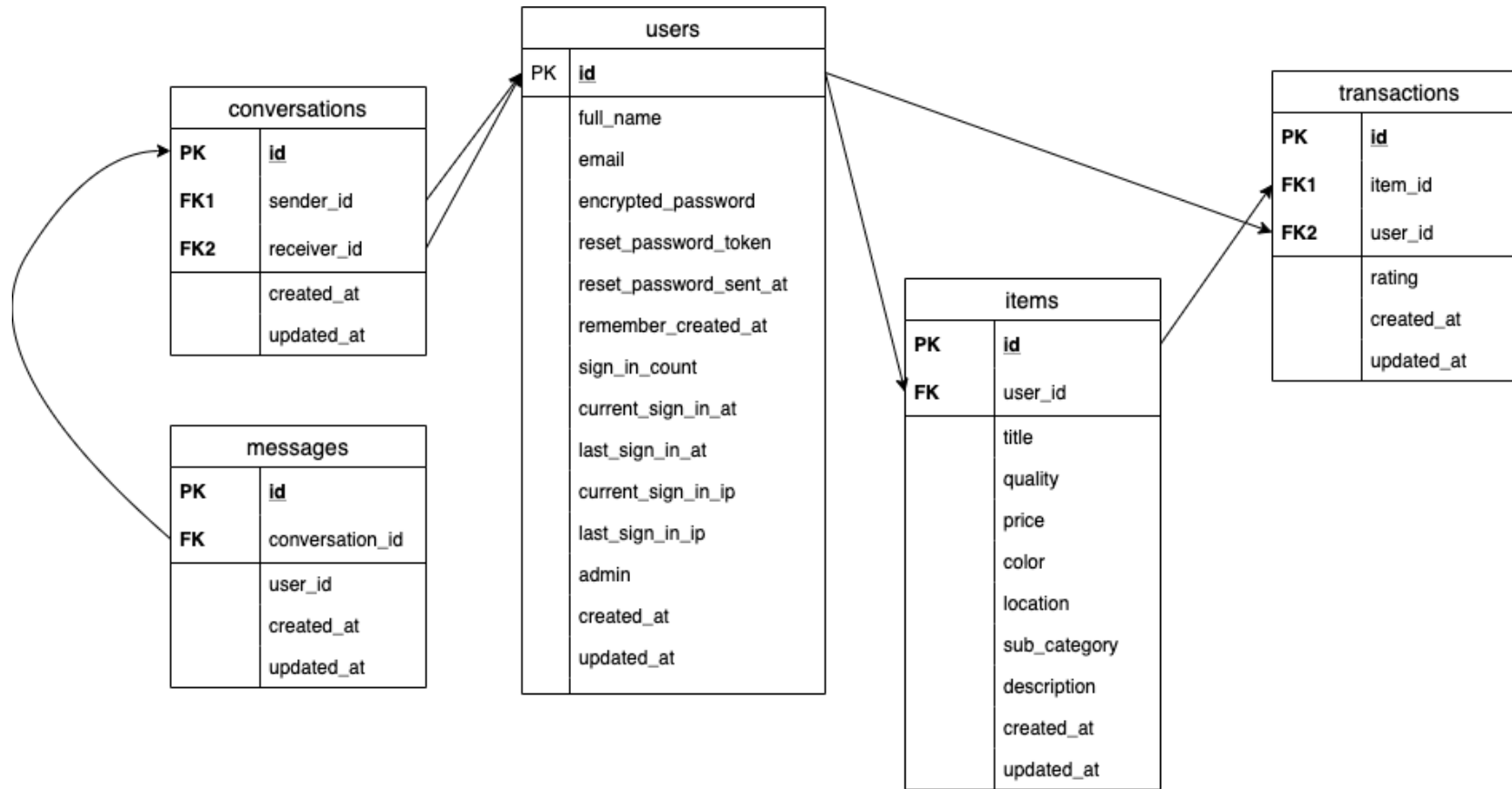


Why FUREX

- Finding furniture at a reasonable price and quality can be challenging for students
- FUREX helps local OSU people find furniture
- Cutting deals and exchanging addresses and information is much safer and convenient within a closed community.

State Diagram





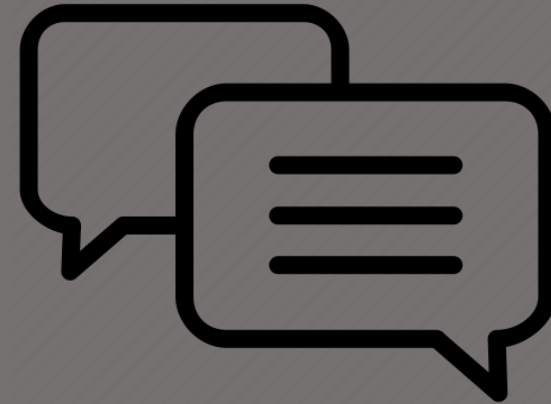
ER Diagram

FUREX Schema (Snippet)

```
create_table "items", force: :cascade do |t|
  t.integer "user_id"
  t.string "title", limit: 100, null: false
  t.string "quality", null: false
  t.decimal "price", null: false
  t.string "color", null: false
  t.string "location", null: false
  t.string "sub_category", null: false
  t.string "description"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["user_id"], name: "index_items_on_user_id"
end

create_table "searches", force: :cascade do |t|
  t.string "keywords"
  t.string "sub_category"
  t.string "color"
  t.string "quality"
  t.string "price"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
end
```

Messaging



Concept

- Feature to enable communication between potential buyers and sellers.
- Communication is initiated by the potential buyer and is item-specific.

On the Item description page

Seller:

g.2

[Conversations](#)

View for this section



```
<% if user_signed_in? && current_user.id != @item.user_id %>
  <%= link_to "Message #{@item.user.full_name}", conversations_path(sender_id:
current_user.id, receiver_id: @item.user.id), method: 'post', class: "button is-link" %>
<% else user_signed_in? && current_user.id == @item.user_id %>
  <%= link_to "Conversations", conversations_path %>
```

Message

Message Seller

g.3 07/30/20 at 2:04 AM

What is this?

Inquire about a item...

Send message

Controller for messages

```
def index
  @messages = @conversation.messages
  # If more than 10 messages, show only latest ten
  if @messages.length > 10
    @over_ten = true
    @messages = @messages[-10..-1]
  end
  if params[:m]
    @over_ten = false
    @messages = @conversation.messages
  end
  @message = @conversation.messages.new
end

# Post
def create
  @message = @conversation.messages.new(message_params)
  # If conversation already present between two users, redirect to it.
  if @message.save
    redirect_to conversation_messages_path(@conversation)
  end
end

# Get
def new
  @message = @conversation.messages.new
end
```

Controller for messages



```
# Define methods
private

# Message has user id and content
def message_params
  params.require(:message).permit(:body, :user_id)
end

def find_conversation
  @conversation =
Conversation.find(params[:conversation_id])
end
```

View for messages

```


<h1>Message Seller</h1>

  <% if @over_ten %>
    <%= link_to "Show previous", '?m=all', class: 'btn btn-primary' %>
  <% end %>

  <div class="col">
    <% @messages.each do |message| %>
      <% if message.body %>
        <% user = User.find(message.user_id) %>
        <em><%= user.full_name %></em>
        <%= message.message_time %>
        <%= sanitize markdown_to_html(message.body) %>
      <% end %>
    <% end %>
  </div>
  <div class="col-lg-3">
    <%= form_for [@conversation, @message] do |f| %>
      <%= f.text_area :body, placeholder: "Inquire about a item...", class: 'form-control' %>
      <%= f.text_field :user_id, value: current_user.id, type: "hidden" %>
      <div>
        <%= f.submit "Send message", class: "btn btn-success mt-2" %>
      </div>
    <% end %>
  </div>
</div>


```

Conversation

Conversations

g.3

Controller for conversations

```
def index
  @users = User.all
  @conversations = Conversation.all
end

# Post new conversation
def create
  # If conversation already present, assign that to @conversation
  if Conversation.between(params[:sender_id], params[:receiver_id]).present?
    @conversation = Conversation.between(params[:sender_id],
    params[:receiver_id]).first
  # If conversation not present, make a new conversation
    @conversation = Conversation.create!(conversation_params)
  end
  # Redirect user to that conversation
  redirect_to conversation_messages_path(@conversation)
end

# Define method

private

# Conversation is between 2 people
def conversation_params
  params.permit(:sender_id, :receiver_id)
end
```


View for conversations

```


<h1>Conversations</h1>
    <% @conversations.each do |conversation| %>
      <% if conversation.sender_id == current_user.id || conversation.receiver_id ==
current_user.id %>
        <% if conversation.sender_id == current_user.id %>
          <% receiver = User.find(conversation.receiver_id) %>
        <% else %>
          <% receiver = User.find(conversation.sender_id) %>
        <% end %>
        <% unless current_user.id == receiver %>
          <div class="mc-auto py-1">
            <%= link_to receiver.full_name, conversation_messages_path(conversation), class:
"btn btn-success" %>
          </div>
        <% end %>
      <% end %>
    </div>
  </div>
</div>


```

Inside routes.rb



```
resources :conversations do  
  resources :messages  
end
```

User Authentication



Devise

Built-in Features:

- Authenticate user
- Remembering a user from saved cookie
- Securely store user's Pa\$\$w0rd
- Track details on each authentication
- Email Confirmation/Account Recovery (did not implement)

Added by Our Team:

- Integrated with Bootstrap
- Allow user to add/edit their full name (field `full_name`)
- Only OSU email will be accepted
- List of Users
- Admin Role & Admin Panel (field `admin`)

Log in

Email

Password

☐ Remember me

Log in

[Sign up](#)

[Forgot your password?](#)

Login Page

Sign up

Only OSU Email(name.#@osu.edu) will be accepted

Full Name

Email

Password **(6 characters minimum)**

Password confirmation

Sign up

[Log in](#)

Signup Page

Create the First Admin Account



```
# ./Rakefile

# rake promote_admin
task :promote_admin => :environment do
  User.first.update_attribute('admin', true)
end
```

- We are born equal (every registered user will not have admin privilege)
- Promote first_user -> admin:
`$ rake promote_admin`
- Admins will have privileges to all resources

Sample List of Users

Users

http://localhost:3000/users

ID	Email	Name	Last Sign-in IP	Last Sign-in At	Membership Since	Admin
1	admin@furex.org	Example	10.0.2.2	2020-07-29 16:53:52 -0500	2020-07-29 16:53:52 -0500	true

Sample Admin Panel

Current Admins

http://localhost:3000/users/add_admin

ID	Email	Name	Last Sign-in IP	Last Sign-in At	Membership Since	Admin
1	admin@furex.org	Example	10.0.2.2	2020-07-29 16:53:52 -0500	2020-07-29 16:53:52 -0500	true

Email:

Add Admin

Security Matters

- Compile with Regulations: GDPR (Expensive \$\$\$), FCAA, HIPPA
 - GDPR: €20 million / 4% of annual global turnover
 - Notify affected user and provide proper solutions
- Hide button/contents from views is not a valid solution
 - Contents can still be accessed via controller
- Solutions:
 - Consider security issues before writing codes
 - Ensure only authorized person will get the corresponded resources

```
# ./app/controllers/users/users_controller.rb

class UsersController < ApplicationController
  # GET /users
  def index
    @title = 'Users'
    if current_user.admin
      @users = User.all
    else
      respond_to do |format|
        format.html { redirect_to home_index_url, alert: 'You are not permitted
to access this page.' }
      end
    end
  end

  # GET /users/add_admin
  def add_admin
    # Trimmed...
  end

  # PUT /users/set_admin
  def set_admin
    # Trimmed...
  end
end
```

Only Admins
Can Access
List of Users

User Account



FUREX

All ▾

Search

Search

Advanced Search

Account

Edit Account

Sign Out

User Profile for Amanda Cheng


Email: cheng.1319@osu.edu

Number of Listings: 5

Add New Item!


Your Items

Window Pane




Show Item | Sell Item | Edit Item | Delete Item

Minimal Bed Frame




Show Item | Sell Item | Edit Item | Delete Item

Clothing Rack



Show Item | Sell Item | Edit Item | Delete Item

Table



User Profile Controller

- Devise `current_user` helper method
- Find all rows in which the `current_user`'s id is the same as the seller's id
- Query methods
 - `.where` finds all instances that satisfy the condition
 - `.count` counts the number of instances (rows) of a table

```
def user_profile
  @title = 'User Profile'
  @full_name = current_user.full_name
  @email = current_user.email

  @user_items = []
  @item = Item.where(:user_id =>
    current_user.id)
  @num_listings = @item.count
end
```

User Profile View

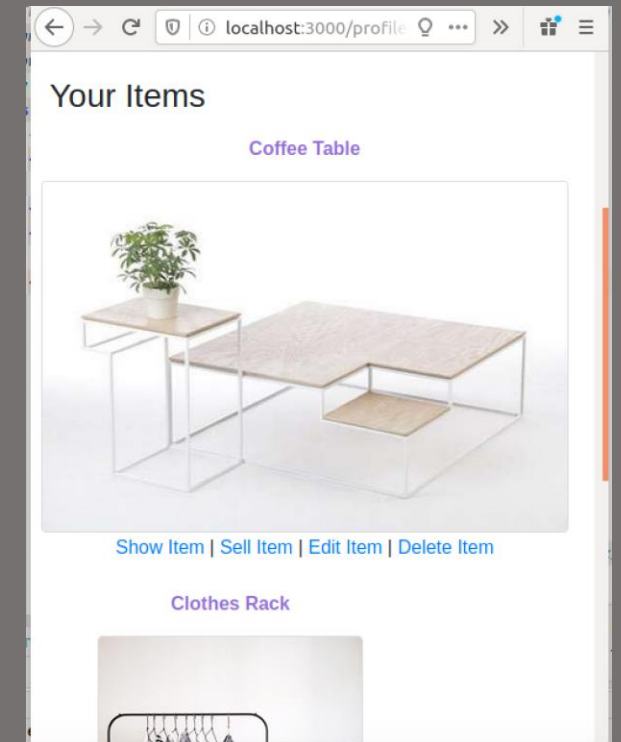
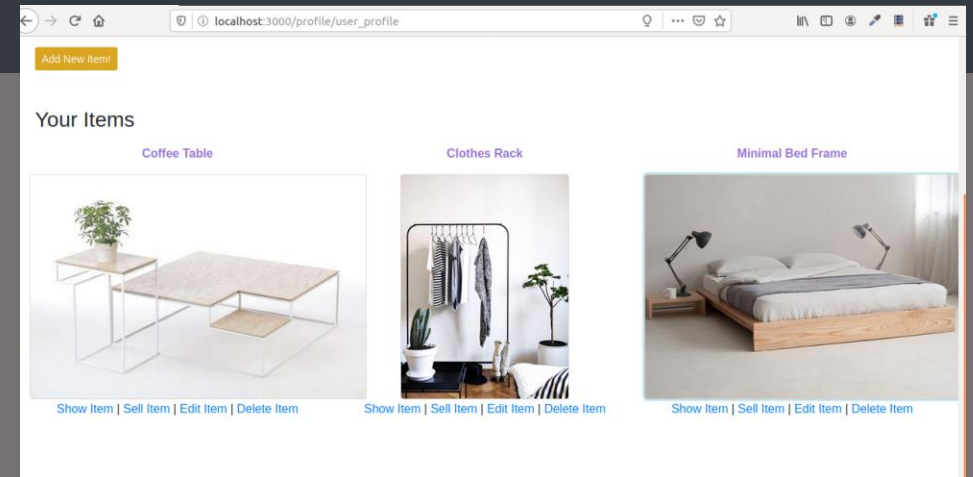
- Bootstrap for responsive seller listings

```

<div class="center_jumbotron bg">
  <h2 id="title"><%= @title + " for " + @full_name %></h2>
  <div class="userAttr">
    <div class="d-flex flex-row">
      <p class="text-primary p-2">Email: </p>
      <p class="p-2"><%= current_user.email %> </p>
    </div>
    <div class="d-flex flex-row">
      <p class="text-primary p-2">Number of Listings: </p>
      <p class="p-2"><%= @num_listings %></p>
    </div>
  </div>

  <div class="newItem">
    <%= link_to "Add New Item!", new_item_path, class: "btn btn-sm btn-primary" %>
  </div>

```

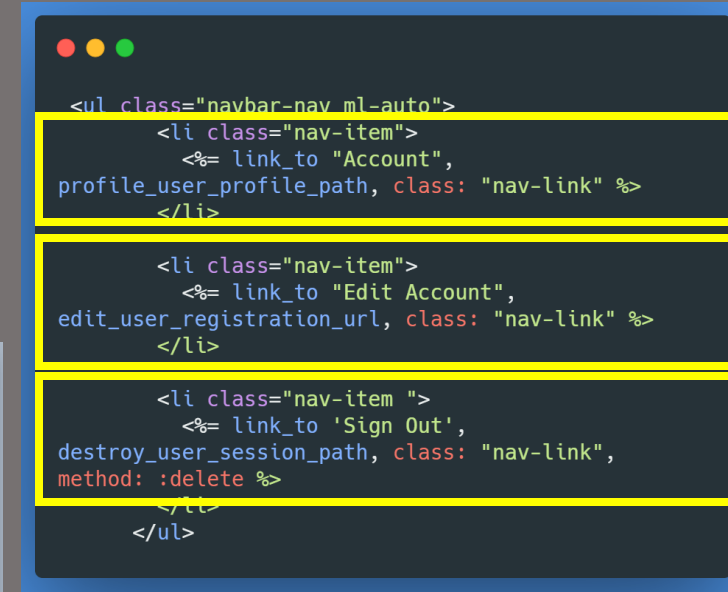
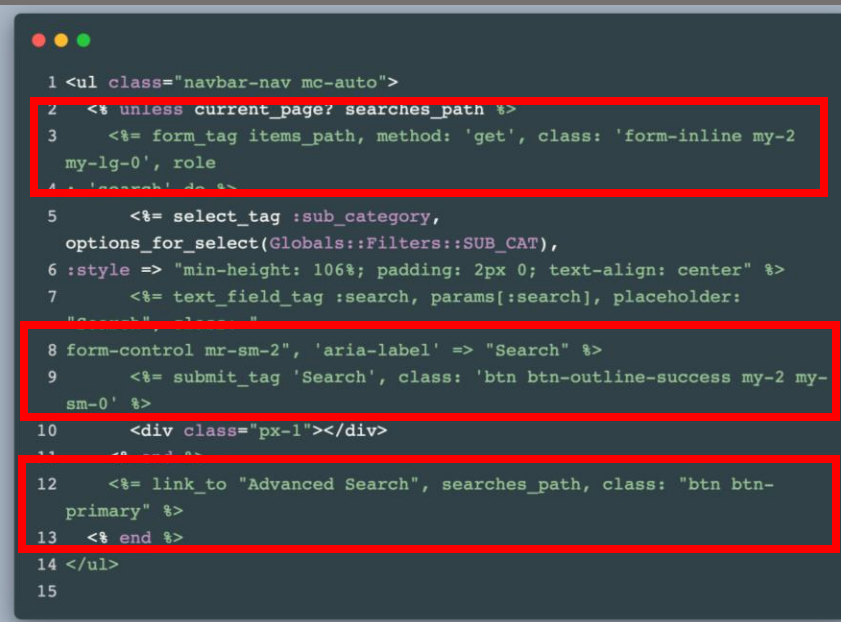
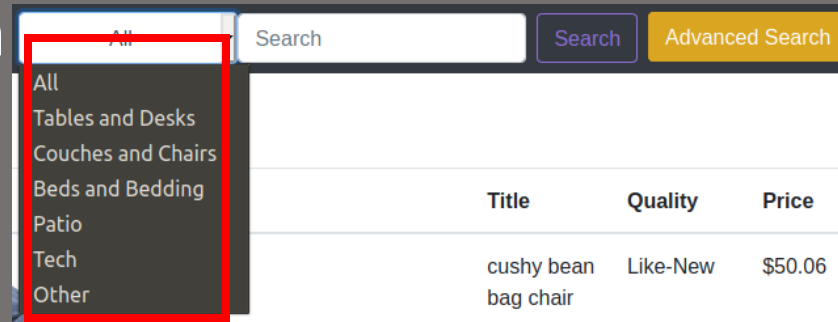
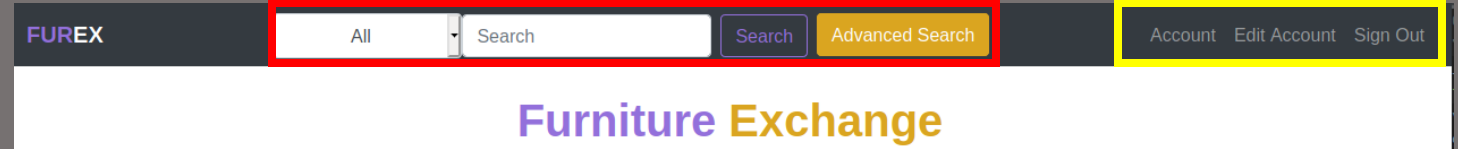


Navigation



View

- Partial implemented in application layout
- Notable Features
 - Basic & Advanced Search
 - Account Operations
 - 100% Bootstrap
- No Model
- No Controller



Items (Listings)



New Item

Title*

Quality*

Price*

Color*

Location*

Sub category*

Description:

Picture* No file chosen

[Back](#)

Add Items

List of Items (Creator/Admin View)

Items

Picture	Title	Quality	Price	Category	Operations
<div><div>Making a meme</div><div>Making a meme that only some people will get</div><div>Making a meme that only meme creators get</div><div>Making a meme that only moderators get</div></div>	asdfs	New	\$32.00	Tables and Desks	<div>Show</div> <div>Edit</div> <div>Destroy</div>

New Item

List of Items (User View)

Items

Picture	Title	Quality	Price	Category	Operations
<div><div>Making a meme</div><div>Making a meme that only some people will get</div><div>Making a meme that only meme creators get</div><div>Making a meme that only moderators get</div></div>	asdfs	New	\$32.00	Tables and Desks	<div>Show</div>

[New Item](#)

Ratings



Creating a Transaction

- Transaction can be added by clicking on sell item
- Upon clicking you can sell to a user by ID
- Enter the user id into User ID and press submit (if you don't know user ID checkout the messages for the item)

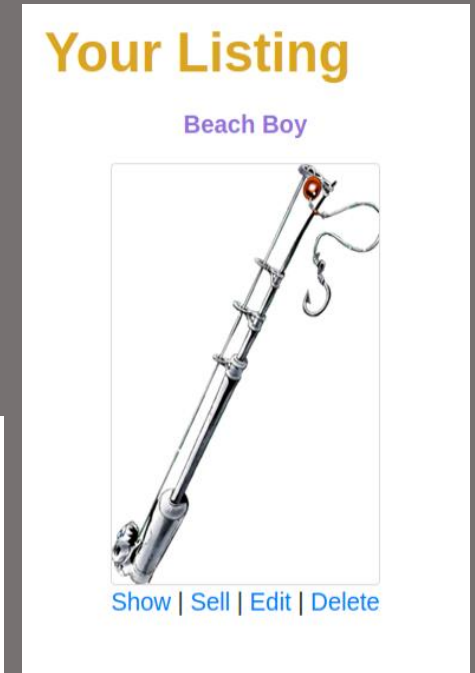
New Transaction

You can only create transaction once.

You may find User ID through conversation.

User ID

Submit



Beach Boy

Item ID: 19

Purchaser ID: 5

Purchaser Name: b5

Rating:

Rating an item

- The rating page contains all transactions and the ratings
- By clicking on Rate it takes us to the transaction
- Clicking on rate allows us to rate the transaction

Order History

Title	Seller	Bought On	Your Rates
Long fishing rod	s2	2020-07-31 13:44:26 -0500	3
Wooden study table	s1	2020-07-31 13:47:50 -0500	4
fhgfh	b4	2020-07-31 19:09:10 -0500	Rate Now
Beach Boy	Troy Stein	2020-07-31 22:00:45 -0500	Rate Now

Rate a Transaction

Please provide a value between 1 to 5

Rating

Submit

Beach Boy

Item ID: 19

Purchaser ID: 5

Purchaser Name: b5

Rating:

[Rate](#)

```
<% if @transaction.user_id != current_user.id %>
  <!-- Code for seller -->
<% end %>

<% if @transaction.user_id == current_user.id %>
  <!-- Code for purchaser-->
<% end %>
<!-- ... -->
```

Transaction (Model)

```
create_table "transactions", force: :cascade do |t|
  t.integer "item_id", null: false
  t.integer "user_id", null: false
  t.integer "rating"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["item_id"], name:
"index_transactions_on_item_id"
  t.index ["user_id"], name:
"index_transactions_on_user_id"
end
```

- item_id: Item for transaction
- user_id: User who purchased item
- rating: Rating for seller

Get Rating for Each User (Controller)

- Helper function called `avg_rating`
- Getting value directly from SQL



```
# ./app/helpers/users_helper.rb

def avg_rating(user_id)
  get_avg_rating = "SELECT AVG(rating) FROM transactions
                    JOIN items ON transactions.item_id = items.id
                    WHERE items.user_id = #{user_id};"
  ActiveRecord::Base.connection.execute(get_avg_rating)[0]["AVG(rating)"]
end
```

Show Rating (View/Controller)

User Profile for Troy

Email: stein.463@osu.edu

Number of Listings: 5

My Rating: 2.67 out of 5

Add New Item!

Calling Helper Method to Get Rating via SQL

```

# ./app/controllers/items_controller.rb

class ItemsController < ApplicationController
  include UsersHelper

  def user_profile
    # Trimmed...
    @rating = avg_rating current_user.id
  end

end
```

Different Content for Different Users (View)

Rate a Transaction

Please provide a value between 1 to 5

Rating

Submit

New Transaction

You can only create transaction once.

You may find User ID through conversation.

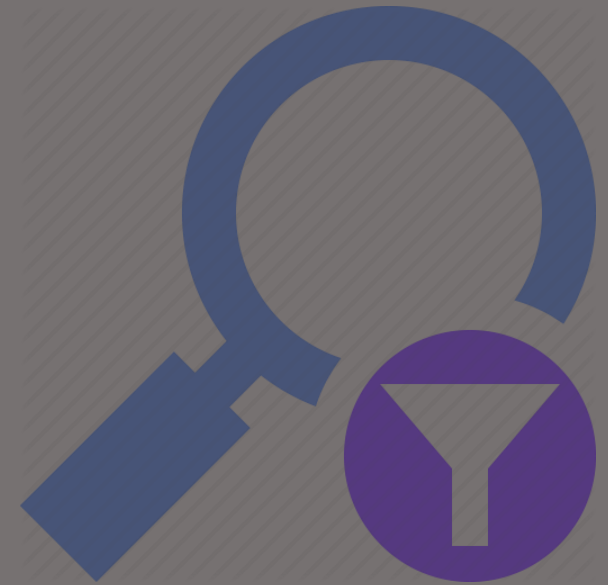
User ID

Submit

```
<% if @transaction.user_id != current_user.id %>
  <!-- Code for seller -->
<% end %>





<% if @transaction.user_id == current_user.id %>
  <!-- Code for purchaser-->
<% end %>
<!-- ... -->
```

Search and Filter



View

- Basic Search
 - Category
 - Search Terms
- Advanced Search
 - Keywords
 - Filters

FUREX						Account Edit Account Sign Out	
Items							
Picture	Title	Quality	Price	Category	Operations		
	cushy bean bag chair	Like-New	\$50.06	Couches and Chairs	Show		
	wooden study table	Used	\$23.56	Tables and Desks	Show		
	long fishing rod	Acceptable	\$4.00	Other	Show	Edit	Destroy
	queen sized mattress	Used	\$45.30	Beds and Bedding	Show	Edit	Destroy

Controller

```
def index
  @title = 'Items'
  if params[:search]
    @items = Item.where('title LIKE ?', "%#{params[:search]}%")
    @items = Item.where('sub_category LIKE ?', params[:sub_category]) if params[:sub_category] != 'All'
  else
    @items = Item.all
  end
end
```

```
module Globals
  # Application wide variables
  class Filters
    SUB_CAT = ['All', 'Tables and Desks', 'Couches and Chairs', 'Beds and Bedding', 'Patio', 'Tech', 'Other'].freeze
    COLORS = %w[All Red Blue Yellow Green Orange Purple Brown Pink White Black].freeze
    QUALITY = %w[All New Like-New Used Acceptable Poor].freeze
    LOCATION = %w[All North East South West Other].freeze
  end

  class Items
    ...
  end
end
```

- Basic Search

- Category

- Application Wide Variables

- Search Terms

- Matching text

The screenshot shows a web application interface. At the top, there is a navigation bar with a dropdown menu currently set to 'All'. To the right of the dropdown is a search input field with the placeholder text 'Search', a 'Search' button, and an 'Advanced Search' button. Below the navigation bar, a dropdown menu is open, displaying a list of categories: 'All', 'Tables and Desks', 'Couches and Chairs', 'Beds and Bedding', 'Patio', 'Tech', and 'Other'. An orange arrow points from the 'Matching text' bullet point in the list to this dropdown menu. Below the dropdown, a table displays search results. The table has three columns: 'Title', 'Quality', and 'Price'. The first row shows 'cushy bean bag chair' with a quality of 'Like-New' and a price of '\$50.06'.

Title	Quality	Price
cushy bean bag chair	Like-New	\$50.06

View

- Basic Search
 - Category
 - Search Terms
- Advanced Search
 - Keywords
 - Filters

Filters

Keywords:

Sub category:
All

Color:
All




Quality:
All

Max price:

Location:
All

[Search](#) [Return](#)

Advanced Search

Picture	Title	Quality	Price	Color	Location	Category	Operations
	cushy bean bag chair	Like-New	\$50.06	Green	East	Couches and Chairs	Show
	wooden study table	Used	\$23.56	Brown	North	Tables and Desks	Show
	long fishing rod	Acceptable	\$4.00	Black	South	Other	Show Edit Destroy

```
<%= form_tag searches_path, method: 'get' do %>
  <div class="field pt-2">
    <%= label_tag :keywords %>:
    <%= text_field_tag :keywords %>
  </div>

  <div class="field pt-2">
    <%= label_tag :sub_category %>:
    <%= select_tag :sub_category,
options_for_select(Globals::Filters::SUB_CAT) %>
  </div>
  ...
end %>
```

Advanced Search

Controller

- Advanced Search

- Keywords
- Filters
 - Sub Category
 - Color
 - Quality
 - Max Price
 - Location

```
class SearchesController < ApplicationController
  def index
    items = Item.all
    items = items.where('title LIKE ?', "%#{
{params[:keywords]}%") if params[:keywords] != ''
    items = items.where('sub_category LIKE ?',
params[:sub_category]) if params[:sub_category] != 'All'
    items = items.where('color LIKE ?', params[:color]) if
params[:color] != 'All'
    items = items.where('quality LIKE ?', params[:quality]) if
params[:quality] != 'All'
    items = items.where('price <= ?', params[:price]) if
params[:price] != ''
    items = items.where('location LIKE ?', params[:location])
if params[:location] != 'All'
    @items = items
  end
  ...
end
```

```
module Globals
  # Application wide variables
  class Filters
    SUB_CAT = ['All', 'Tables and Desks', 'Couches
and Chairs', 'Beds and Bedding', 'Patio', 'Tech',
'Other'].freeze
    COLORS = %w[All Red Blue Yellow Green Orange
Purple Brown Pink White Black].freeze
    QUALITY = %w[All New Like-New Used Acceptable
Poor].freeze
    LOCATION = %w[All North East South West
Other].freeze
  end

  class Items
    ...
  end
end
```

Filters

Keywords:

Sub category:

All ▼

Color:

All ▼

Quality:

All ▼

Max price:

Location:

All ▼

Search

Return

LIVE CODE PRESENTATION