

Here are some figures to consider while answering the following questions.

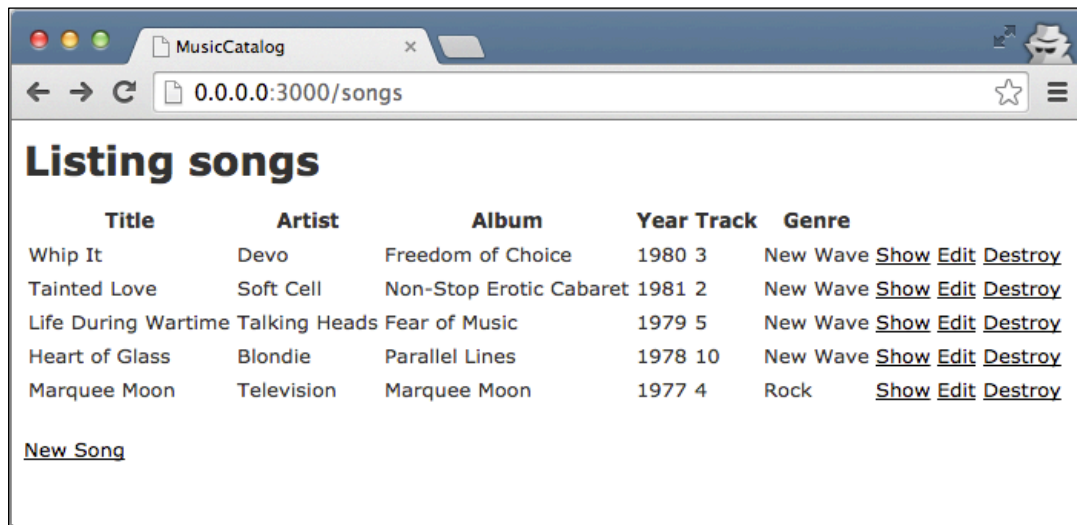


Figure 1. Example page from Music Catalog web app.

```
1 MusicCatalog::Application.routes.draw do
2   resources :songs
3 end
```

Figure 2. config/routes.rb

```
$ rake routes
Prefix Verb  URI Pattern          Controller#Action
songs  GET   /songs(.:format)    songs#index
       POST  /songs(.:format)    songs#create
new_song GET   /songs/new(.:format) songs#new
edit_song GET   /songs/:id/edit(.:format) songs#edit
song  GET   /songs/:id(.:format) songs#show
       PATCH /songs/:id(.:format) songs#update
       PUT   /songs/:id(.:format) songs#update
       DELETE /songs/:id(.:format) songs#destroy
```

Figure 3. Output of rake routes command.

```
1 # == Schema Information
2 #
3 # Table name: songs
4 #
5 # id          :integer          not null, primary key
6 # title       :string(255)
7 # artist      :string(255)
8 # album       :string(255)
9 # year        :string(255)
10 # track       :integer
11 # genre       :string(255)
12 # created_at  :datetime
13 # updated_at  :datetime
14 #
15
16 class Song < ActiveRecord::Base
17 end
```

Figure 4. app/models/song.rb

```
1 class CreateSongs < ActiveRecord::Migration
2   def change
3     create_table :songs do |t|
4       t.string :title
5       t.string :artist
6       t.string :album
7       t.string :year
8       t.integer :track
9       t.string :genre
10
11       t.timestamps
12     end
13   end
14 end
```

Figure 5. db/migrate/20140930033607_create_songs.rb

```

1  class SongsController < ApplicationController
2    def index
3      @songs = Song.all
4    end
5
6    def show
7      @song = Song.find(params[:id])
8    end
9
10   def new
11     @song = Song.new
12   end
13
14   def edit
15     @song = Song.find(params[:id])
16   end
17
18   def create
19     @song = Song.new(song_params)
20     respond_to do |format|
21       if @song.save
22         format.html { redirect_to @song, notice: 'Song was successfully created.' }
23         format.json { render action: 'show', status: :created, location: @song }
24       else
25         format.html { render action: 'new' }
26         format.json { render json: @song.errors, status: :unprocessable_entity }
27       end
28     end
29   end
30
31   def update
32     @song = Song.find(params[:id])
33     respond_to do |format|
34       if @song.update(song_params)
35         format.html { redirect_to @song, notice: 'Song was successfully updated.' }
36         format.json { head :no_content }
37       else
38         format.html { render action: 'edit' }
39         format.json { render json: @song.errors, status: :unprocessable_entity }
40       end
41     end
42   end
43
44   def destroy
45     @song = Song.find(params[:id])
46     @song.destroy
47     respond_to do |format|
48       format.html { redirect_to songs_url }
49       format.json { head :no_content }
50     end
51   end
52
53   private
54   # Never trust parameters from the scary internet, only allow the white list through.
55   def song_params
56     params.require(:song).permit(:title, :artist, :album, :year, :track, :genre)
57   end
58 end

```

Figure 6. app/controllers/songs_controller.rb

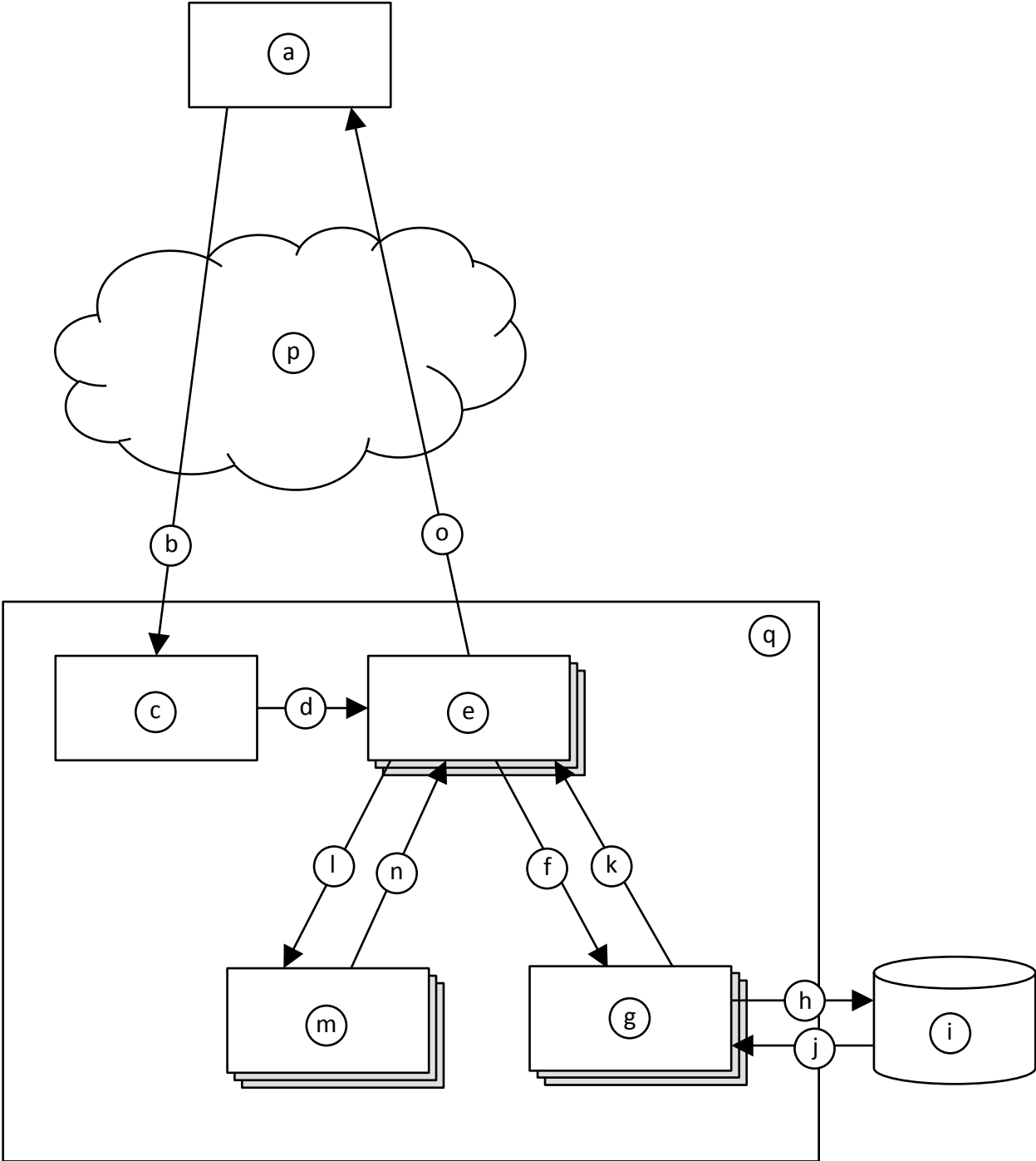
```

1 <h1>Listing songs</h1>
2
3 <table>
4   <thead>
5     <tr>
6       <th>Title</th>
7       <th>Artist</th>
8       <th>Album</th>
9       <th>Year</th>
10      <th>Track</th>
11      <th>Genre</th>
12      <th></th>
13      <th></th>
14      <th></th>
15    </tr>
16  </thead>
17
18  <tbody>
19    <% @songs.each do |song| %>
20      <tr>
21        <td><%= song.title %></td>
22        <td><%= song.artist %></td>
23        <td><%= song.album %></td>
24        <td><%= song.year %></td>
25        <td><%= song.track %></td>
26        <td><%= song.genre %></td>
27        <td><%= link_to 'Show', song %></td>
28        <td><%= link_to 'Edit', edit_song_path(song) %></td>
29        <td><%= link_to 'Destroy', song, method: :delete, data: { confirm: 'Are you sure?' } %></td>
30      </tr>
31    <% end %>
32  </tbody>
33 </table>
34
35 <br>
36
37 <%= link_to 'New Song', new_song_path %>

```

Figure 7. app/views/songs/index.html.erb

Problem: First consider this figure depicting the Rails MVC architecture.



Now, given the architectural diagram, think about how the web page in Figure 1 would have come to be displayed. Fill in each lettered item from the figure (blanks at left) the most appropriate label number (at right). Note that you will not use all of the label numbers.

- | | |
|----------|--|
| a. _____ | 1) routes.rb (Figure 2) |
| b. _____ | 2) song.rb (Figure 4) |
| c. _____ | 3) 20140930033607_create_songs.rb (Figure 5) |
| d. _____ | 4) songs_controller.rb (Figure 6) |
| e. _____ | 5) index.html.erb (Figure 7) |
| f. _____ | 6) Ye Olde Internet |
| g. _____ | 7) Rails server |
| h. _____ | 8) Web browser |
| i. _____ | 9) Call to SongsController#index |
| j. _____ | 10) Call to SongsController#show |
| k. _____ | 11) Call to Song::all |
| l. _____ | 12) Data returned by Song::all |
| m. _____ | 13) Call to Song::find |
| n. _____ | 14) Data returned by Song::find |
| o. _____ | 15) Call to CreateSongs#change |
| p. _____ | 16) Data returned from CreateSongs#change |
| q. _____ | 17) Call to index.html.erb (whatever that means) |
| | 18) Data returned from index.html.erb |
| | 19) Invocation of SQL query |
| | 20) Data returned form SQL query |
| | 21) HTTP GET request |
| | 22) HTTP response |
| | 23) Database |

Solution:

a. 8

b. 21

c. 1

d. 9

e. 4

f. 11

g. 2

h. 19

i. 23

j. 20

k. 12

l. 17

m. 5

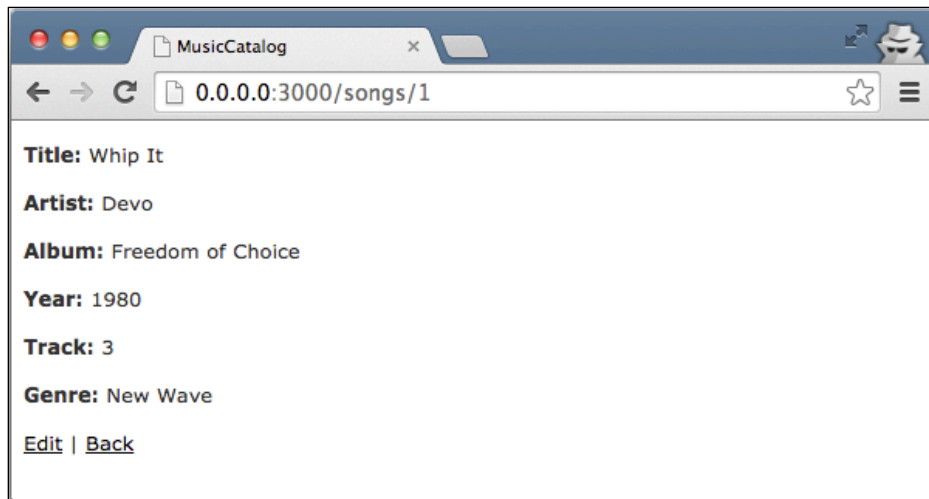
n. 18

o. 22

p. 6

q. 7

Problem: In Figure 1, if you were to click the “Show” link for “Whip It”, this page would display.



Write the ERB file for this page. Assume that a layout, `application.html.erb`, already exists, so your ERB need only include the main content being displayed. Your ERB must include the following types of HTML elements: **p** and **strong**.

Solution:

It's OK to omit line 1.

```
1 <p id="notice"><%= notice %></p>
2
3 <p>
4   <strong>Title:</strong>
5   <%= @song.title %>
6 </p>
7
8 <p>
9   <strong>Artist:</strong>
10  <%= @song.artist %>
11 </p>
12
13 <p>
14   <strong>Album:</strong>
15   <%= @song.album %>
16 </p>
17
18 <p>
19   <strong>Year:</strong>
20   <%= @song.year %>
21 </p>
22
23 <p>
24   <strong>Track:</strong>
25   <%= @song.track %>
26 </p>
27
28 <p>
29   <strong>Genre:</strong>
30   <%= @song.genre %>
31 </p>
32
33 <%= link_to 'Edit', edit_song_path(@song) %> |
34 <%= link_to 'Back', songs_path %>
```

Problem: Modify the web app such that the page from Figure 1 includes only songs from 1980 or later.
Here are a few hints:

- To create a new array:
 - `my_array = Array.new`
- To add an item to the end of an array:
 - `my_array.push(my_item)`
- To convert a string to an integer:
 - `my_int = my_string.to_i`

Solution:

Here's one straightforward way to solve the problem by changing SongsController#index (in songs_controller.rb):

```
1 class SongsController < ApplicationController
2   def index
3     # BEFORE:
4     #@songs = Song.all
5     #
6     # AFTER:
7     @songs = Array.new
8     Song.all.each do |song|
9       if song.year.to_i >= 1980 then
10        @songs.push(song)
11      end
12    end
13  end
end
```

(The rest of the file remains unchanged.)

Problem: Imagine that you wanted to change the web app such that it now stores the name of the songwriter with each song. Answer the following in plain English.

- a. How would you go about updating the web app's "M" (as in MVC) component?
- b. How would you change the "V" files in the above figures?
- c. How would you change the "C" files in the above figures?

Solution:

- a. To update the model (“M”) component, you would need to create a new migration (similar to Figure 5). A common way to do this would be with this Rails command:

```
$ rails generate migration AddSongwriterToSongs songwriter:string
```

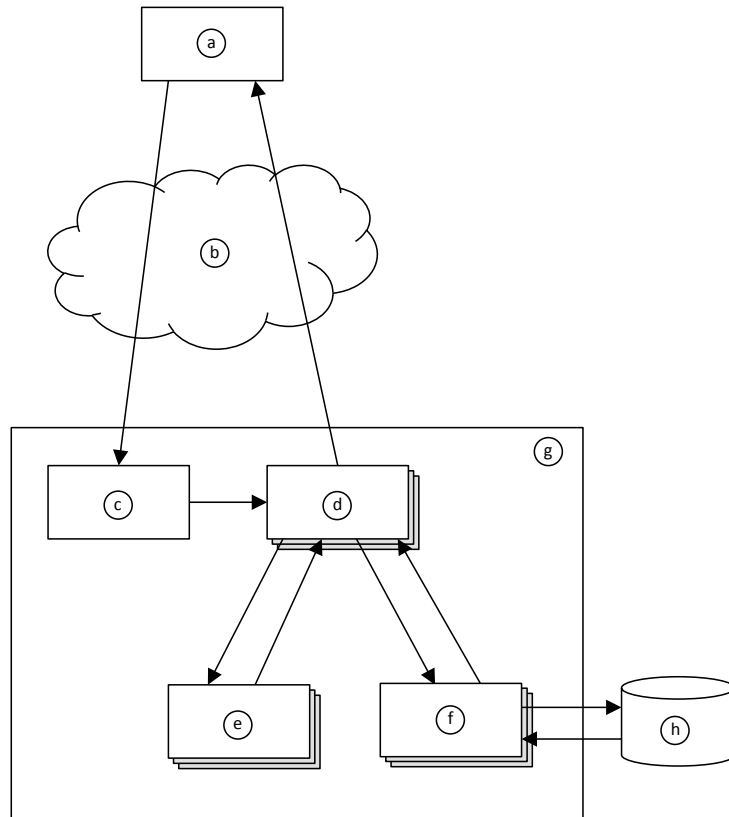
This command generates an appropriate migration file. Note that the class name after migration must be of the form `AddXxxToYyy`.

- b. The view (“V”) files above (i.e., the ERBs) would need to also display the songwriter values by adding appropriate HTML and calls to `song.songwriter`.
- c. In the controller (“C”) file above (`song_controller.rb`), the `song_params` method would need to be updated to account for the `:songwriter` parameter.

Problem:

Given the Rails MVC architectural diagram below, label each component.

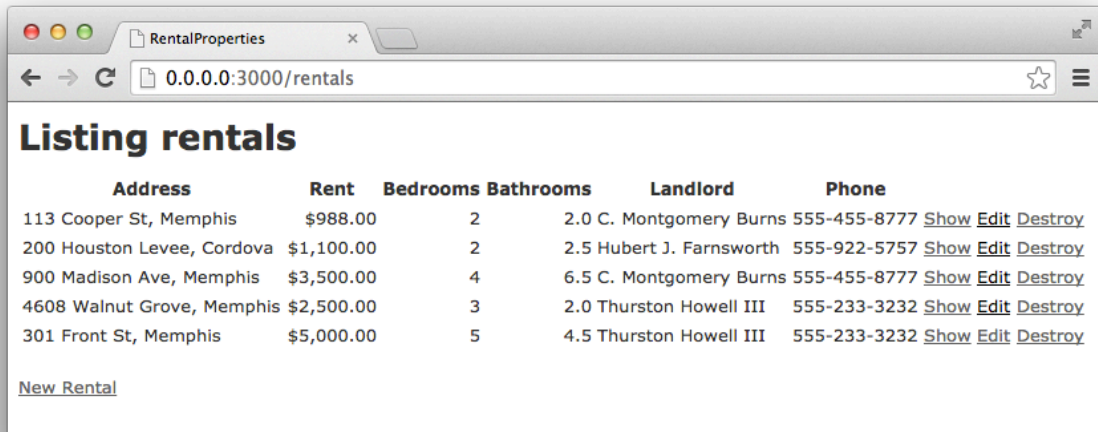
1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____



Solution:

1. Web Browser
2. Ye Olde Internet
3. Rails Router
4. Controller
5. View
6. Model
7. Rails Server
8. Database

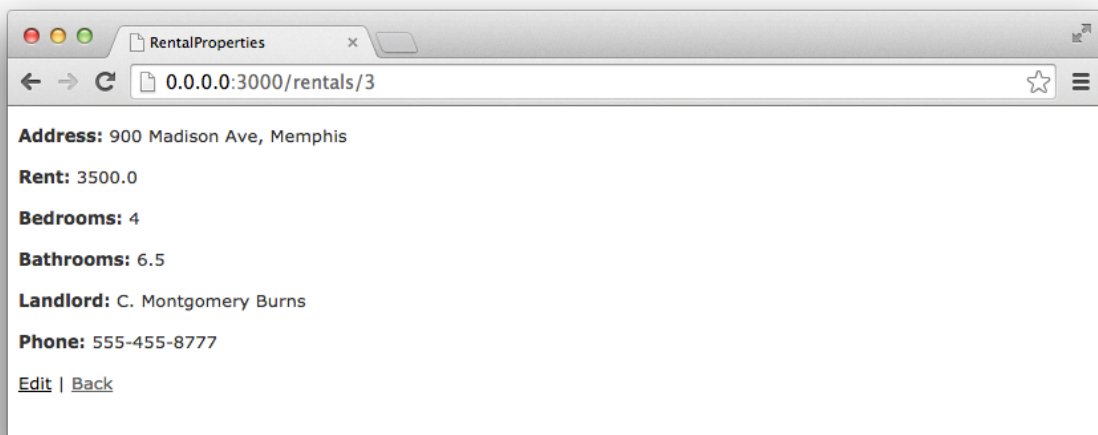
Here are some figures to consider while answering the following questions.



Address	Rent	Bedrooms	Bathrooms	Landlord	Phone	
113 Cooper St, Memphis	\$988.00	2	2.0	C. Montgomery Burns	555-455-8777	Show Edit Destroy
200 Houston Levee, Cordova	\$1,100.00	2	2.5	Hubert J. Farnsworth	555-922-5757	Show Edit Destroy
900 Madison Ave, Memphis	\$3,500.00	4	6.5	C. Montgomery Burns	555-455-8777	Show Edit Destroy
4608 Walnut Grove, Memphis	\$2,500.00	3	2.0	Thurston Howell III	555-233-3232	Show Edit Destroy
301 Front St, Memphis	\$5,000.00	5	4.5	Thurston Howell III	555-233-3232	Show Edit Destroy

[New Rental](#)

Figure 8. Index page for rental-property web app.



Address: 900 Madison Ave, Memphis

Rent: 3500.0

Bedrooms: 4

Bathrooms: 6.5

Landlord: C. Montgomery Burns

Phone: 555-455-8777

[Edit](#) | [Back](#)

Figure 9. Show-rental page for rental-property web app.


```

$ rake routes
  Prefix Verb  URI Pattern          Controller#Action
  rentals GET   /rentals(.:format)  rentals#index
                   POST  /rentals(.:format)  rentals#create
  new_rental GET  /rentals/new(.:format) rentals#new
  edit_rental GET  /rentals/:id/edit(.:format) rentals#edit
  rental GET   /rentals/:id(.:format) rentals#show
                   PATCH /rentals/:id(.:format) rentals#update
                   PUT   /rentals/:id(.:format) rentals#update
                   DELETE /rentals/:id(.:format) rentals#destroy

```

Figure 10. Result of "rake routes" command for rental-property web app.

```

1  # == Schema Information
2  #
3  # Table name: rentals
4  #
5  # id          :integer          not null, primary key
6  # address     :string(255)
7  # rent        :decimal(, )
8  # bedrooms   :integer
9  # bathrooms   :float
10 # landlord    :string(255)
11 # phone       :string(255)
12 # created_at  :datetime
13 # updated_at  :datetime
14 #
15
16 class Rental < ActiveRecord::Base
17 end

```

Figure 11. Rental-property web app file: app/models/rental.rb

```

1 ▼ class RentalsController < ApplicationController
2   def index
3     @rentals = Rental.all
4   end
5
6   def show
7     # YOUR ANSWER HERE
8   end
9
10  def new
11    @rental = Rental.new
12  end
13
14  def edit
15    @rental = Rental.find(params[:id])
16  end
17
18 ▼ def create
19   @rental = Rental.new(rental_params)
20 ▼   respond_to do |format|
21 ▼     if @rental.save
22       format.html { redirect_to @rental, notice: 'Rental was successfully created.' }
23       format.json { render action: 'show', status: :created, location: @rental }
24 ▼     else
25       format.html { render action: 'new' }
26       format.json { render json: @rental.errors, status: :unprocessable_entity }
27     end
28   end
29 end
... and so on ...

```

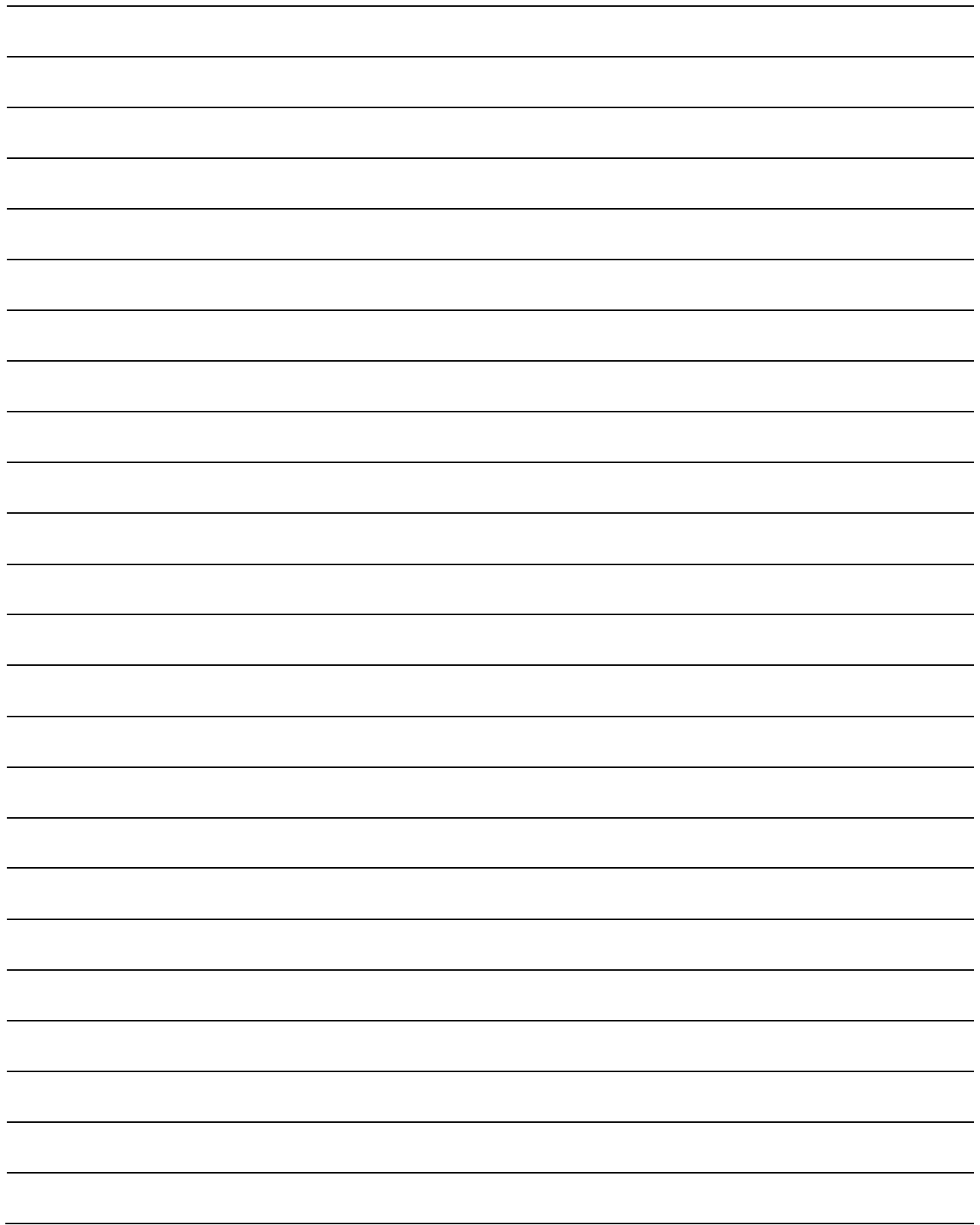
Figure 12. Rental-property web app file: app/controllers/rentals_controller.rb

```

1 <h1>Listing rentals</h1>
2
3 <table>
4   <thead>
5     <tr>
6       <th>Address</th>
7       <th>Rent</th>
8       <th>Bedrooms</th>
9       <th>Bathrooms</th>
10      <th>Landlord</th>
11      <th>Phone</th>
12      <th></th>
13      <th></th>
14      <th></th>
15    </tr>
16  </thead>
17
18  <tbody>
19    <%= @rentals.each do |rental| %>
20      <tr>
21        <td><%= rental.address %></td>
22        <td style="text-align: right;"><%= number_to_currency(rental.rent) %></td>
23        <td style="text-align: right;"><%= rental.bedrooms %></td>
24        <td style="text-align: right;"><%= rental.bathrooms %></td>
25        <td><%= rental.landlord %></td>
26        <td><%= rental.phone %></td>
27        <td><%= link_to 'Show', rental %></td>
28        <td><%= link_to 'Edit', edit_rental_path(rental) %></td>
29        <td><%= link_to 'Destroy', rental, method: :delete, data: { confirm: 'Are you sure?' } %>
30      </td>
31    </tr>
32    <%= end %>
33  </tbody>
34 </table>
35
36 <br>
37 <%= link_to 'New Rental', new_rental_path %>

```

Figure 13. Rental-property web app file: app/views/index.html.erb



Solution:

```
def show
  @rental = Rental.find(params[:id])
end
```

<p>

Address: <%= @rental.address %>

</p>

<p>

Rent: <%= @rental.rent %>

</p>

<p>

Bedrooms: <%= @rental.bedrooms %>

</p>

<p>

Bathrooms: <%= @rental.bathrooms %>

</p>

<p>

Landlord: <%= @rental.landlord %>

</p>

<p>

Phone: <%= @rental.phone %>

</p>

cont'd next page

<%= link_to 'Edit', edit_rental_path(@rental) %>

<%= link_to 'Back', rentals_path %>

Solution:

It would violate the single-responsibility principle (SRP) because a controller is responsible for translating between UI actions and operations on the model, whereas a view is responsible for UI presentation. Line 3 is an operation on the model — a controller responsibility. Moving this line into the view would mean that the view now has both view and controller responsibilities.

Here is a figure to consider while answering the following questions.

```
1 # id      :integer      not null, primary key
2 # name    :string
3 # email   :string
4 class User < ActiveRecord::Base
5   has_many :sales
6 end
```

```
1 # id      :integer      not null, primary key
2 class Sale < ActiveRecord::Base
3   belongs_to :user
4   has_many :line_items
5 end
```

```
1 # id      :integer      not null, primary key
2 # quantity :integer
3 class LineItem < ActiveRecord::Base
4   belongs_to :sale
5   belongs_to :item_description
6 end
```

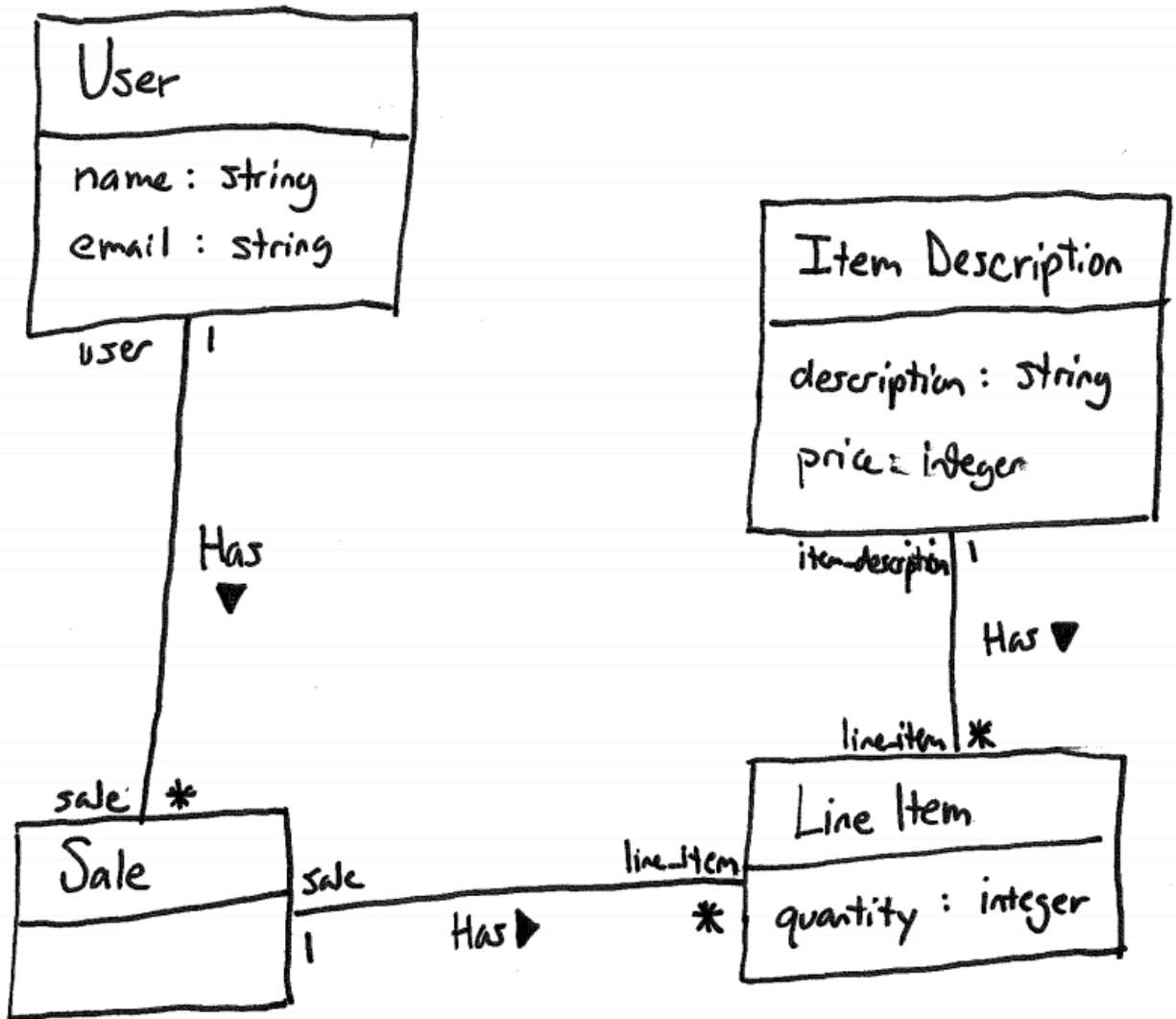
```
1 # id      :integer      not null, primary key
2 # description :string
3 # price    :integer
4 class ItemDescription < ActiveRecord::Base
5   has_many :line_items
6 end
```

Figure 14. Model classes for a point-of-sale system.

Problem:

Create a UML class diagram representing the Figure 14 point-of-sale model classes. Be sure to label all associations and association ends, and include all multiplicities. Don't include "id" attributes (objects have identity by default).

Solution:



Problem:

Consider the following execution of a point-of-sale system with the model in Figure 14. Two users register: Alice Zed (azed@memphis.edu) and Bob Young (byoung@memphis.edu). Alice purchases the following things: 2 Bug Zappers (\$20 each) and 1 Garden Hose (\$12 each). Bob purchases the following things: 3 Bug Zappers and 1 Spider Spray (\$4 each). Later, Alice makes another purchase: 1 Spider Spray. Create an object diagram that depicts the model objects after this execution.

Solution:

