

Exploration: Research and Implementation Learnings

Final Project Sprint 02
Presentation

Description of Exploration


For this sprint, I focused on exploring the implementation of role-based access control and security clearance features in my Rails application. My objective was to enhance user experience for Shipping Agents and Logistics Managers by ensuring efficient workflows through dynamic role management.

Brief Overview of What You Learned

1. Role-based access control (RBAC): Dynamically managing permissions based on user roles.



2. Real-time feedback mechanisms: Alerts for restricted access areas.



3. Rails Active Record: Handling data relationships and validations efficiently.

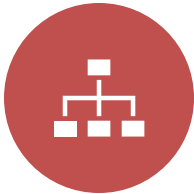


4. CSS media queries and JavaScript toggles: UI enhancements like dark mode.



5. Database schema management: Designing for efficient permission storage.

Summary of Learnings and Implemented Features



1. ROLE-BASED
ACCESS CONTROL
IMPLEMENTATION



- DYNAMIC
PERMISSIONS FOR
SHIPPING AGENTS
AND LOGISTICS
MANAGERS.



2. SECURITY
CLEARANCE
FEEDBACK



- REAL-TIME
NOTIFICATIONS FOR
ACCESS CLEARANCE
ISSUES.



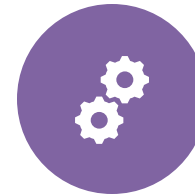
3. DARK MODE
INTEGRATION



- TOGGLE WITH
PERSISTENT
SETTINGS USING
TAILWIND CSS AND
LOCAL STORAGE.



4. DATABASE
ENHANCEMENTS

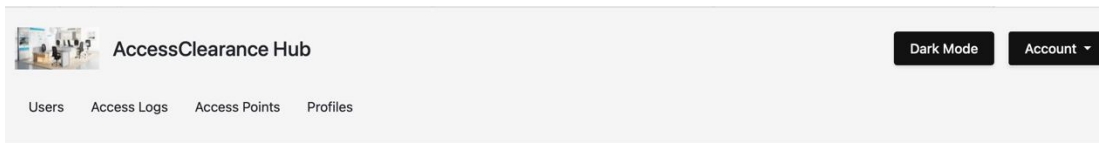


- OPTIMIZED
SCHEMA DESIGN
AND ACTIVE
RECORD
VALIDATIONS.

Screenshots of Key Implementations



1. ROLE-BASED ACCESS CONTROL IMPLEMENTATION



Create New User

Please fill in the details below to create a new user.

Username

Password

Leave blank if you do not want to change the password. A default password will be set if none is provided.

Full name

Email

Select a role

- ✓ Admin
- Editor
- Viewer
- Shipping agent
- Logistics manager

Last login



- DYNAMIC PERMISSIONS FOR SHIPPING AGENTS AND LOGISTICS MANAGERS.

APP/MODELS/USER.RB

```
# Permission methods
± willmaddocke
def can_create_items?
  admin? || logistics_manager?
end

± willmaddocke
def can_view_items?
  viewer? || shipping_agent? || logistics_manager?
end

± willmaddocke
def can_edit_items?
  admin? || editor? || logistics_manager?
end

± willmaddocke
def can_delete_items?
  admin? || logistics_manager?
end

± willmaddocke
def can_request_access?
  shipping_agent?
end

± willmaddocke
def can_audit_logs?
  logistics_manager?
end
end
```

Continued...



2. SECURITY CLEARANCE FEEDBACK



- REAL-TIME
NOTIFICATIONS FOR
ACCESS CLEARANCE
ISSUES.

APP/CONTROLLERS/ELE
VATED_ACCESS_REQUE
STS_CONTROLLER.RB

AccessClearance Hub

Dark Mode Account

Users Access Logs Access Points Profiles

Elevated Access Requests

New Elevated Access Request

Select User:	Select Access Point:	Records per page:
✓ Select User	Select Access Point	10
Verlene Lynch		
Zoila Kilback		
Ahmad Gerhold		
Fredric Funk		
Cesar Nikolaus		
Luis Schaefer DO		
Ned Abernathy		
Ashleigh Kerluke VM		
Val Wyman		
Devora Kilback		
Rory Douglas		
Jerry Tillman JD		
Ashleigh Kerluke VM	85209 Arica Avenue, North Merlinoport, KS 77046	
Val Wyman	8718 Arla Keys, East Isela, CT 95921-9662	
Val Wyman	55226 Emalie Spring Lake, Dando, MT 06955-2601	

```
private
# Use callbacks to share common setup or constraints between actions.
# willmaddocke
def set_elevated_access_request
  @elevated_access_request = ElevatedAccessRequest.find(params[:id])
end

# Only allow a list of trusted parameters through.
# willmaddocke
def elevated_access_request_params
  params.require(:elevated_access_request).permit(:user_id, :access_point_id, :reason, :status)
end


# Ensure that either Shipping Agents or Logistics Managers can access the request form.
# willmaddocke
def ensure_request_access
  unless current_user.role == 'shipping_agent' || current_user.role == 'logistics_manager'
    redirect_to root_path, alert: 'Access denied.'
  end
end

# Ensure only Logistics Managers can approve or deny requests.
# willmaddocke
def ensure_logistics_manager
  redirect_to root_path, alert: 'Access denied.' unless current_user.role == 'logistics_manager'
end
end
```

Continued...



3. DARK MODE INTEGRATION

 AccessClearance Hub

Light Mode

Account ▾

Users

Access Logs

Access Points

Profiles

Access Logs

New Access Log

Provides comprehensive tracking of all access events, aiding in security management.

Select Username

Select Address

Success

Select Date

Records per page:

All

All

All

All

10

User ID	Access Point ID	Timestamp	Successful	Actions
reyes	4324 Leif Stravenue, Port Howard, WI 46774	11/22/2024 10:07	Yes	<div>ShowEditDelete</div>
reyes	Suite 289 61985 Jerde Crescent, Lake Guyburgh, MN 81393	11/22/2024 10:07	No	<div>ShowEditDelete</div>
reyes	Suite 289 61985 Jerde Crescent, Lake Guyburgh, MN 81393	11/22/2024 10:07	No	<div>ShowEditDelete</div>
reyes	Suite 726 63381 Mayert Roads, West Noah, WI 06852-2649	11/22/2024 10:07	Yes	<div>ShowEditDelete</div>
carter	60224 MacGyver Trace, Johnsontown, GA 85163	11/22/2024 10:07	No	<div>ShowEditDelete</div>
amie	60224 MacGyver Trace, Johnsontown, GA 85163	11/22/2024 10:07	No	<div>ShowEditDelete</div>
amie	Suite 939 692 Cinda Pine, Yostburgh, NV 45930	11/22/2024 10:07	No	<div>ShowEditDelete</div>



- TOGGLE WITH
PERSISTENT SETTINGS
USING TAILWIND CSS
AND LOCAL STORAGE.

APP/JAVASCRIPT/DAR
K_MODE_TOGGLE.JS

```
document.addEventListener("turbo:load", function() {
  const toggleButton : HTMLInputElement = document.getElementById("dark-mode-toggle");
  const body : HTMLDocument = document.body;

  if (!toggleButton) {
    console.error("Dark mode toggle button not found!");
    return;
  }

  // Check for stored preference in localStorage
  const storedTheme : string = localStorage.getItem("theme") || "light";

  // Apply the stored theme on page load
  if (storedTheme === "dark") {
    body.classList.add("dark-mode");
    toggleButton.innerText = "Light Mode";
    toggleButton.classList.remove("btn-light");
    toggleButton.classList.add("btn-dark");
  } else {
    body.classList.remove("dark-mode");
    toggleButton.innerText = "Dark Mode";
    toggleButton.classList.remove("btn-dark");
    toggleButton.classList.add("btn-light");
  }

  // Prevent duplicate event listeners
  if (toggleButton.dataset.initialized === "true") return;
  toggleButton.dataset.initialized = "true";

  // Toggle dark mode on button click
  toggleButton.addEventListener("click", function() {
    const isDarkMode : boolean = body.classList.toggle("dark-mode");

    // Update button text and style
    toggleButton.innerText = isDarkMode ? "Light Mode" : "Dark Mode";
    toggleButton.classList.toggle("btn-dark", isDarkMode);
    toggleButton.classList.toggle("btn-light", !isDarkMode);

    // Store the new theme in localStorage
    localStorage.setItem("theme", isDarkMode ? "dark" : "light");
  });
});
```

Continued...



4. DATABASE ENHANCEMENTS



- OPTIMIZED
SCHEMA DESIGN
AND ACTIVE
RECORD
VALIDATIONS.
APP/MODELS/USER
.RB

```
class AddIndexesToAccessPoints < ActiveRecord::Migration[7.1]
  ⤵ willmaddocke
  def change
    # Adding indexes for performance
    add_index :access_points, :location
    add_index :access_points, :access_level
    add_index :access_points, :description

    # Uncomment the lines below to add unique indexes if applicable
    # add_index :access_points, :location, unique: true
    # add_index :access_points, :access_level, unique: true
    # add_index :access_points, :description, unique: true
  end
end
```

```
class User < ApplicationRecord
  # Include default Devise modules
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable

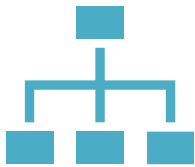
  enum role: { admin: 0, editor: 1, viewer: 2, shipping_agent: 3, logistics_manager: 4 }

  after_initialize do
    self.role ||= :viewer if new_record? && role.nil?
  end

  # Relationships
  has_one :profile, dependent: :destroy
  has_many :access_logs, dependent: :destroy

  # Validations
  validates :email, presence: true, uniqueness: true
  validates :username, presence: true, uniqueness: true
  validates :role, inclusion: { in: roles.keys }
```


Issues Encountered



1. Role-Based Complexity:
Avoiding conflicts in
overlapping permissions.



2. Real-Time Feedback
Integration: Challenges with
asynchronous notifications.



3. Styling Consistency:
Ensuring contrast and
readability in dark mode.

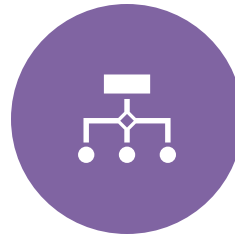
Successes



1. SUCCESSFULLY
IMPLEMENTED ROLE-BASED
ACCESS CONTROL AND
SECURITY CLEARANCE CHECKS.



2. DELIVERED FUNCTIONAL
DARK MODE TOGGLE WITH
PERSISTENT SETTINGS.



3. CREATED A SCALABLE AND
MAINTAINABLE SCHEMA FOR
FUTURE EXTENSIONS.



4. ENHANCED USER
EXPERIENCE WITH INTUITIVE
AND RESPONSIVE
NOTIFICATIONS.

Resources Used



[1. MDN Web Docs: Prefers Color Scheme Media Query](#)



[2. Rails Guides: Active Record Basics](#)



[3. CSS Tricks: Dark Mode in CSS](#)

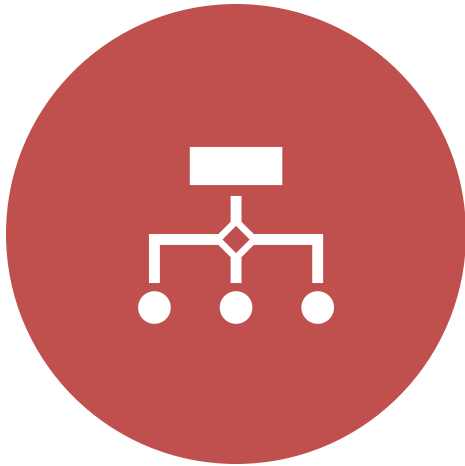


[4. Tailwind CSS Dark Mode](#)



[5. Stimulus Reflex Dark Mode Tutorial](#)

GitHub Repository



EXPLORE THE CODE AND PROGRESS FOR THIS
SPRINT IN THE GITHUB REPOSITORY:



[HTTPS://GITHUB.COM/WILLMADDOCK/FINAL-
PROJECT/TREE/SPRINT02](https://github.com/willmaddock/final-project/tree/sprint02)