HR Tracker — Technical Documentation

1. Overview

HR Tracker is a mini user-role-based HR onboarding application designed to allow easy access and management of employee data and tasks. Built with

- **Ruby on Rails 8**
- **MySQL**
- **TailwindCSS** (for layouts and styling)
- **Bootstrap Modals** (for task management UI)

2. Feature Summary

- Role-based access control(Employee and Admin)
- Onboarding task assignment + tracking
- Admin dashboards with filters
- Responsive, styled with Bootstrap and TailwindCSS
- Bootstrap Modals for quick task completion

3. Prerequisites

These are the system requirements for local testing:

- Ruby 3.x: Download the latest version from the official Ruby website.
- Rails 8: Set up your local development environment by following the Rails guide.
- MySQL 8+: The Rails guide will walk you through the environment setup for MySQL
- Node.js: Install Node to manage the assets pipeline and run JS code.
- Yarn

- Text Editor like Visual Studio Code

4. Setup & Installation

1. Clone the Repository:
    - `git clone https://github.com/Temmarie/HR-Dashboard.git`
    - `cd HR-Dashboard`
2. Set up the environment
    - Run `bundle install` to install the Rails gems and dependencies.

3. Configure your database
    - Navigate to `database.yml` to configure your database.
        - Update `config/database.yml` with your MySQL credentials(`username` and `password`)
    - Create, migrate and seed your database:
      `rails db:create db:migrate db:seed`
4. TailwindCSS Build Note: If styling does not appear: run `rails tailwindcss:build`. If the issue persists, run: `rake assets:precompile`
5. Start the server with `bin/rails s`.
6. View the app on `localhost:3000`

5. Project Architecture

The application follows the standard **Model-View-Controller (MVC)** pattern:

- **Models:** Handle database interactions (`User`, `Employee`, `OnboardingTask`, `Department`)

- **Controllers:** Coordinate logic between models and views (`SessionsController`, `OnboardingTasksController`, `DepartmentsController`,`DashboardController`, `RegistrationsController`, `Admin::DashboardControler`)

- **Views:** Render UI templates using ERB and TailwindCSS

User requests → Routes → Controller → Model → View → Response

MVC Breakdown

Here's your content formatted into a clean, readable **table format** for documentation, separated into logical sections: Public Routes, Authentication, Employee Management, Onboarding Tasks, Admin Area, and Model Associations.

Public Routes

Path	Description	Controller#Action
`/`	Homepage	`dashboard#home`
`/dashboard`	User dashboard	`dashboard#index`

Authentication & Sessions

`SessionsController`

Method & Path	Description	Action
`GET /session/new`	Login form	`new`
`POST /session`	Create session (login)	`create`
`GET /session`	View current session	`show`
`PUT /session`	Update current session	`update`
`PATCH /session`	Update current session	`update`
`DELETE /session`	Logout	`destroy`
`GET /session/edit`	Session update form	`edit`

`RegistrationsController`

Method & Path	Description	Action
`GET /registrations/new`	Sign-up form	`new`

`POST /registrations`	Create new user	`create`

`PasswordsController`

Method & Path	Description	Action
`GET /passwords/new`	Request password reset form	`new`
`POST /passwords`	Submit password reset request	`create`
`GET /passwords/:token/edit`	Edit form via reset token	`edit`
`PATCH /passwords/:token`	Submit new password	`update`

Employees

`EmployeesController`

Method & Path	Description	Action
`GET /employees`	List employees (admin only)	`index`

Onboarding Tasks

`OnboardingTasksController`

Method & Path	Description	Action
`GET /onboarding_tasks`	List onboarding tasks (current user)	`index`
`PATCH /onboarding_tasks/:id`	Update a task (mark complete, etc.)	`update`
`PUT /onboarding_tasks/:id`	Update a task	`update`

Admin Area

`Admin::DashboardController`

Method & Path	Description	Action
`GET /admin/dashboard`	Admin dashboard	`index`

`GET /admin/assign_task`	Task assignment form	`new`
`POST /admin/assign_task`	Assign a task to an employee	`create`

Model Associations Summary

Model	Associations
`User`	`has_one :employee` `has_many :sessions`
`Employee`	`belongs_to :user` `has_many :onboarding_tasks`
`OnboardingTask`	`belongs_to :employee`
`Session`	`belongs_to :user`

---

6. SQL Operations

SQL Operations Table

Feature	Action	SQL Operation
Authentication	Login (`sessions#create`)	`SELECT * FROM users WHERE email_address = ? LIMIT 1;`
	Create Session	`INSERT INTO sessions (user_id, ip_address, user_agent, created_at, updated_at) VALUES (?, ?, ?, ?, ?);`
	Logout (`sessions#destroy`)	`DELETE FROM sessions WHERE id = ?;`

User Registration	Sign Up (`registrations#create`)	`INSERT INTO users (email_address, password_digest, role, created_at, updated_at) VALUES (?, ?, ?, ?, ?);`
Employees	View All Employees	`SELECT * FROM employees;`
	Add New Employee	`INSERT INTO employees (name, department, user_id, created_at, updated_at) VALUES (?, ?, ?, ?, ?);`
Departments	List All Departments	`SELECT * FROM departments;`
Onboarding Tasks	Assign Task	`INSERT INTO onboarding_tasks (employee_id, task_name, status, created_at, updated_at) VALUES (?, ?, 'pending', ?, ?);`
	Update Task Status	`UPDATE onboarding_tasks SET status = ?, updated_at = ? WHERE id = ?;`
	View Tasks (Employee Dashboard)	`SELECT onboarding_tasks.* FROM onboarding_tasks INNER JOIN employees ON employees.id = onboarding_tasks.employee_id WHERE employees.user_id = ?;`
Admin Dashboard	View Employees and Tasks	`SELECT * FROM employees; SELECT * FROM onboarding_tasks WHERE employee_id IN (?);`
Dashboard	View My Profile and Tasks	`SELECT * FROM employees WHERE user_id = ? LIMIT 1; SELECT * FROM onboarding_tasks WHERE employee_id = ?;`
Active Storage	Upload File (Blob + Attachment)	`INSERT INTO active_storage_blobs (...) VALUES (...); INSERT INTO active_storage_attachments (...) VALUES (...);`

Database Integrity	Enforce Foreign Key Constraints	`ALTER TABLE employees ADD CONSTRAINT fk_employees_users FOREIGN KEY (user_id) REFERENCES users(id);`

---

7. Authentication

Employee authentication has been added as a bonus feature to improve user security

- Built using Rails' built-in authentication generator
- Uses `has_secure_password` for hashing
- User roles: `"admin"` and `"employee"`
- Admin account is seeded for initial access.
- Admin cannot create an account and will be given direct access.

8. Suggested Improvements

Here are some suggested improvements or upgrades to make this app production-grade:

- Use **ActionCable** or **Hotwire** for live status changes
- Use Email invites for employees to create their accounts instead of url
- Tailwind CSS has issues with Rails integration, so consider using styled components to avoid library crashes.
- Use **ActiveStorage** to allow HR to attach documents to employee profiles. Store the documents in a cloud service like Cloudinary or Firebase for production.

- Admins can define reusable task templates per department

- Use **Sidekiq** for sending notifications or reminders

- Optimize database queries to reduce load times and improve performance.

9. Support and Contributions

For any questions or issues, please refer to the troubleshooting section or contact the development team.

Author

Grace Tamara Ekunola

- Github: @Temmarie
- Twitter: @TemmarieW
- Linkedin: Grace Tife Ekunola

Contributing

Feel free to fork the repo, create a new branch, and submit a pull request or check the [issues page](issues/)

License