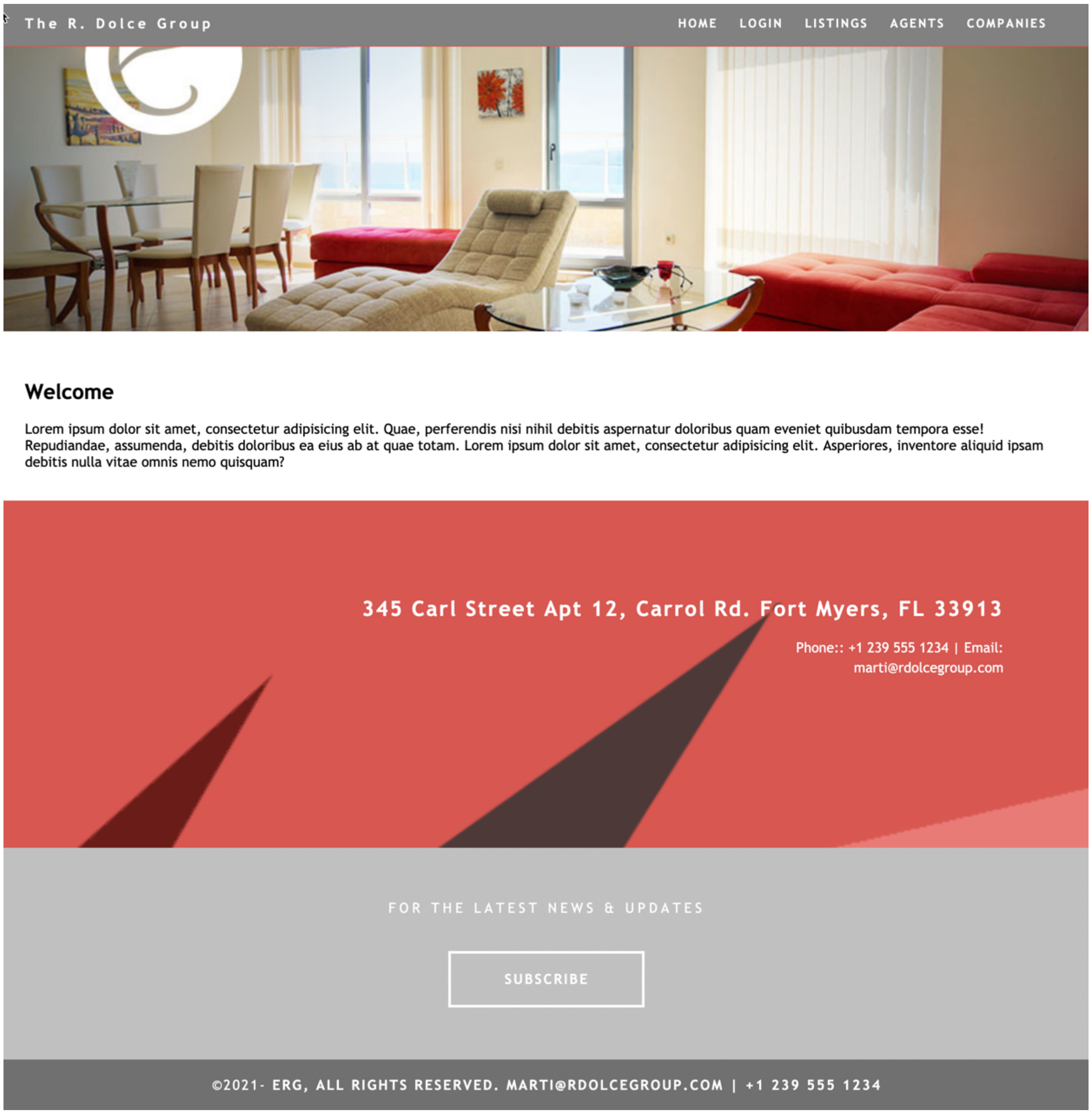


Welcome

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quae, perferendis nisi nihil debitis aspernatur doloribus quam eveniet quibusdam tempora esse! Repudiandae, assumenda, debitis doloribus ea eius ab at quae totam. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Asperiores, inventore aliquid ipsam debitis nulla vitae omnis nemo quisquam?

345 Carl Street Apt 12, Carrol Rd. Fort Myers, FL 33913

Phone:: +1 239 555 1234 | Email: marti@rdolcegroup.com

©2021- ERG, ALL RIGHTS RESERVED. MARTI@RDOLCEGROUP.COM | +1 239 555 1234

Building A Sinatra App With MySQL

Phase 2 Project Development for Flatiron School

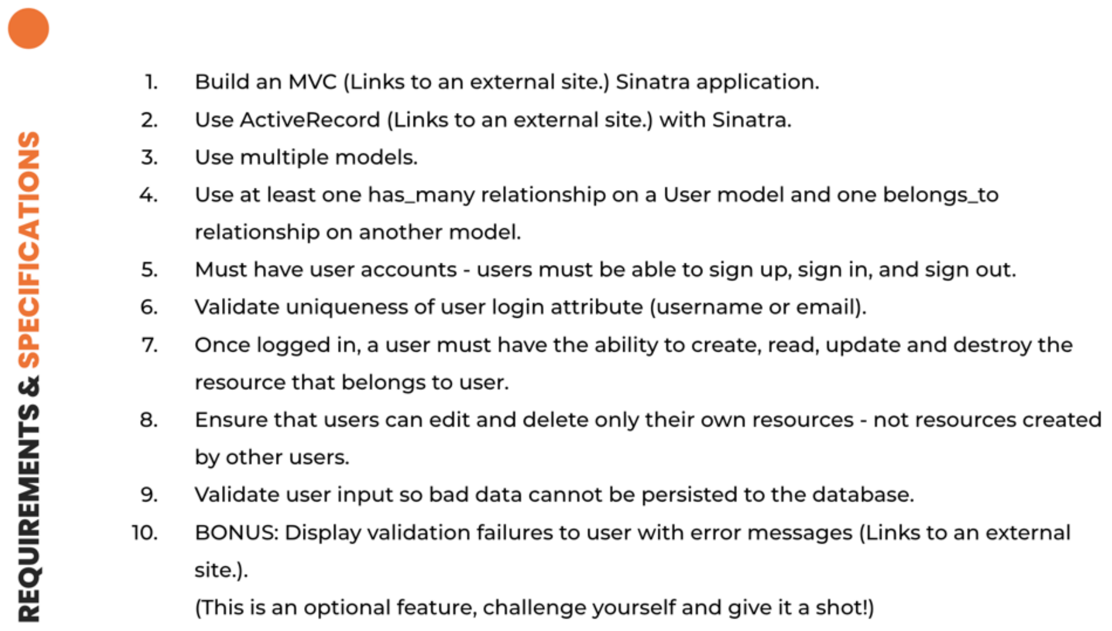
DRAFT 20200117-V022

Getting Started

As a new developer you need to understand how the process works when thinking of moving into a new job or contract work opportunity. You have to start with a plan. Typically the work begins when the stakeholders and the business team has decided on a new technology 'business' requirement which then evolves into a list of specifications outlining the functionality and features of this new business initiative.

This application uses Sinatra with Active Record with a MySQL database.

With this idea in mind, my Phase 2 Requirements from Flatiron School are represented in the chart below.



REQUIREMENTS & SPECIFICATIONS

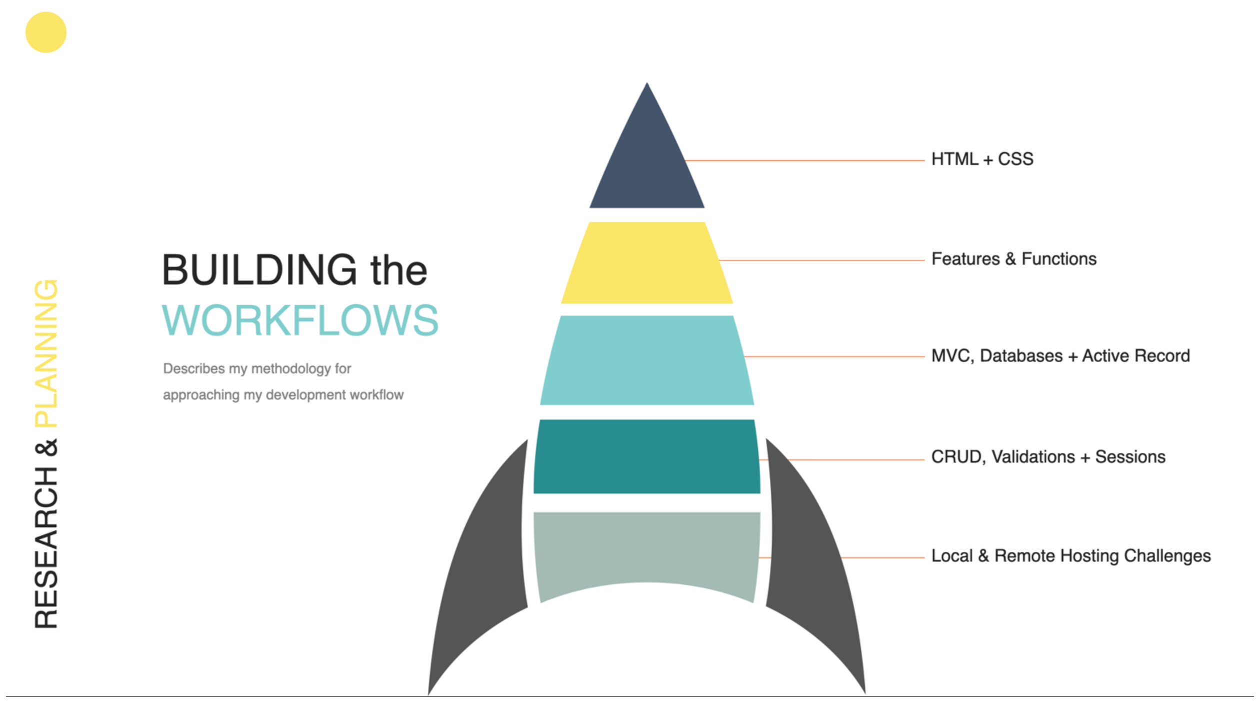
1. Build an MVC (Links to an external site.) Sinatra application.
2. Use ActiveRecord (Links to an external site.) with Sinatra.
3. Use multiple models.
4. Use at least one has_many relationship on a User model and one belongs_to relationship on another model.
5. Must have user accounts - users must be able to sign up, sign in, and sign out.
6. Validate uniqueness of user login attribute (username or email).
7. Once logged in, a user must have the ability to create, read, update and destroy the resource that belongs to user.
8. Ensure that users can edit and delete only their own resources - not resources created by other users.
9. Validate user input so bad data cannot be persisted to the database.
10. BONUS: Display validation failures to user with error messages (Links to an external site.).
    (This is an optional feature, challenge yourself and give it a shot!)

Now the SPEC’s As Defined by the Project

Technical specifications define functional and feature goals for the project. These can be further defined as user stories to ensure you have met each requirement as a developer.

[ ] Use Sinatra to build the app
[ ] Use ActiveRecord for storing information in a database
[ ] Include more than one model class (e.g. User, Post, Category)
[ ] Include at least one has_many relationship on your User model (e.g. User has_many Posts)
[ ] Include at least one belongs_to relationship on another model (e.g. Post belongs_to User)
[ ] Include user accounts with unique login attribute (username or email)
[ ] Ensure that the belongs_to resource has routes for Creating, Reading, Updating and Destroying
[ ] Ensure that users can't modify content created by other users
[ ] Include user input validations
[ ] BONUS - not required - Display validation failures to user with error message (example form URL e.g. /posts/new)
[ ] Your README.md includes a short description, install instructions, a contributors guide and a link to the license for your code

Confirm
[ ] You have a large number of small Git commits
[ ] Your commit messages are meaningful
[ ] You made the changes in a commit that relate to the commit message
[ ] You don't include changes in a commit that aren't related to the commit message


RESEARCH & PLANNING
BUILDING the
WORKFLOWS
Describes my methodology for
approaching my development workflow
HTML + CSS
Features & Functions
MVC, Databases + Active Record
CRUD, Validations + Sessions
Local & Remote Hosting Challenges

Getting started on a project of this size as a newby can be overwhelming for some. I've elected to break the overall specifications into modular topics. That way I can plan my approach.

DEVELOPMENT + GITHub

Discusses how I use RubyMine to speed up the development of Github projects and how I manage merging different branches for each iteration of my project.

HTML + CSS

The User Experience or the Front - End defines the structure of the application and how an end user interacts with the features and functionality of the application. I'll use a layout.erb to define my overall theme and build my pages using CSS.

Features & Functions

My project tracks the status of our guest rentals and payments received. First I'll build the application in Sinatra using basic concepts in Ruby.

MVC, Databases + Active Record

Choosing a database framework is the way to achieve MVC using Sinatra. Based on the requirements for this project we will use Active Record. However I'll use MySQL as the backend for my application.

CRUD, REST, USERS, Validations + Sessions

I'll need to determine how to use routes to get and post data from the back-end. In addition, I'll also need to add in features for making changes and/or deleting that data. Using a combination of HTML/ERB pages will achieve this. Thinking of it as a series of actions: list all, creating, saving, editing, searching, filtering and deleting data.

Users Roles & Logins
Understanding roles or user permission should validate the current users permissions and update the database should those permissions change.

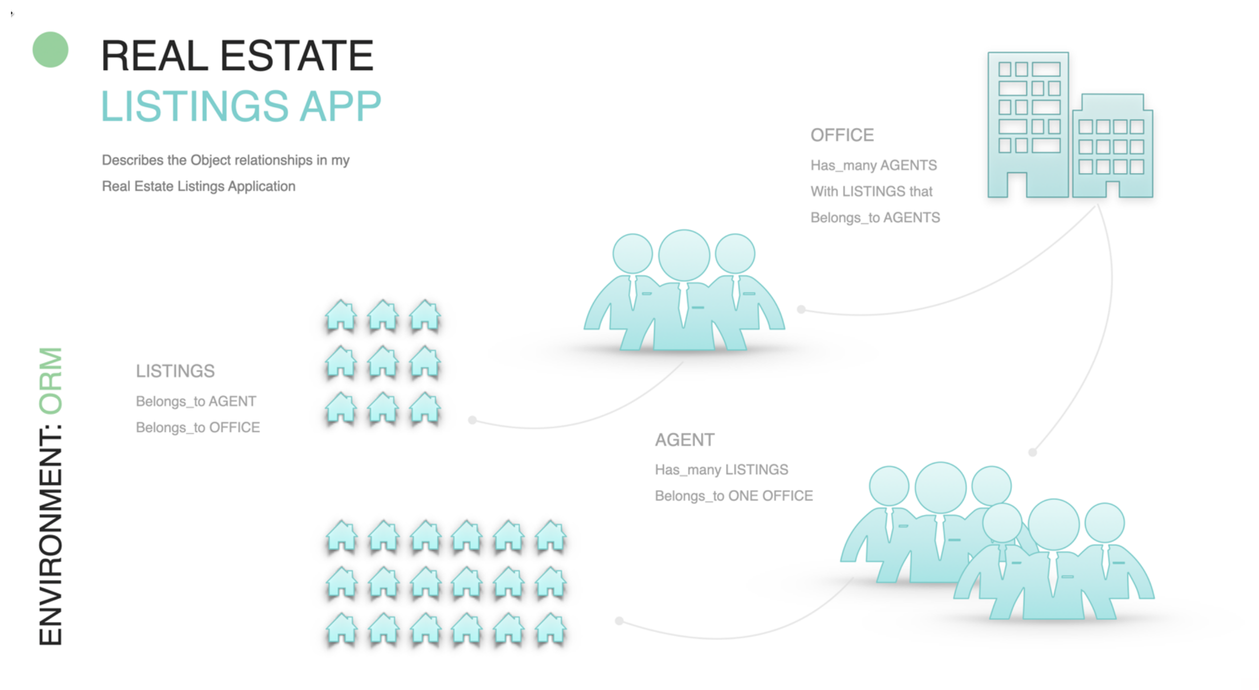
Sessions works with permissions to validate if the current user is logged in or not, establishes that current user's permissions and tracks the users activities while visiting the site.

Users should be able to register or sign-up. This will require: sign-up, login, and redirects to handle the user login/logout.

Local & Remote Hosting Challenges

After the project is completed and before I submit it for review, I'll need to plan on hosting it on a remote server. For the purpose of this session it was suggested we use Heroku.

Now that I have all that figured out, I can actually start planning my code.



ENVIRONMENT: ORM

REAL ESTATE LISTINGS APP

Describes the Object relationships in my
Real Estate Listings Application

OFFICE
Has_many AGENTS
With LISTINGS that
Belongs_to AGENTS

LISTINGS
Belongs_to AGENT
Belongs_to OFFICE

AGENT
Has_many LISTINGS
Belongs_to ONE OFFICE

AGENT CONTACT INFO TABLE

Agents have the ability to update/edit personal information and to track their listings.

FIELDS	DATATYPE	NULL	KEY	EXTRA
Agent_ID	INTEGER	No	PRI	Auto_Inc
Listings_ID	INTEGER	No	FK	Listings
Username	VARCHAR	Yes		
Password	VARCHAR	Yes		
Office	VARCHAR	Yes		
Email	VARCHAR	Yes		
Company	VARCHAR	Yes		
Contact	VARCHAR	Yes		
Photo	BLOB	Yes		
Created_At	TIMESTAMP	Yes		
Updated_At	TIMESTAMP	Yes		

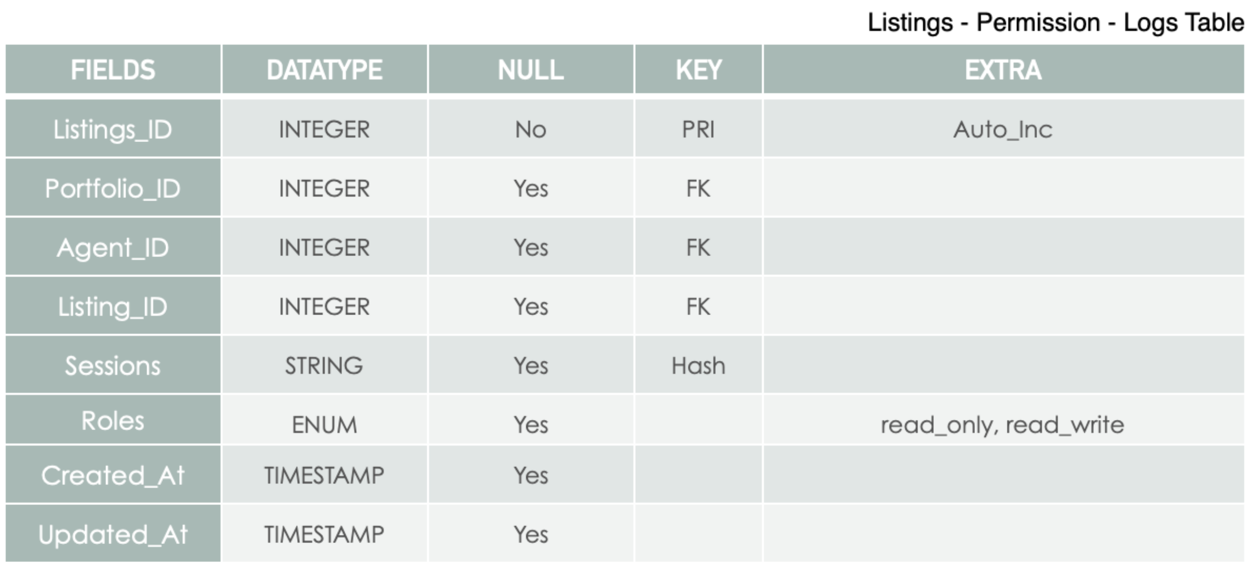
(OFFICE/FINANCIAL) PORTFOLIO TABLE

Functioning as a financial and visual portfolio for potential investors, prospective clients and other agents or offices, agents manage their portfolios and have control over who has access to the portfolio. The data also functions as a newsletter.

FIELDS	DATATYPE	NULL	KEY	EXTRA
Portfolio_ID	INTEGER	No	PRI	Auto_Inc
Agent_ID	INTEGER	No	FK	Agent
Listing_ID	VARCHAR	No	FK	Listings
Address	INTEGER	Yes		Address, City, State, Zip
Keywords	VARCHAR	Yes		Marketing SEO
Status	VARCHAR	Yes		Sold, New, Rent, Purchase, Lease
Shared_with	ENUM	Yes		(private, shared, public)
Date_Listed	DATE	Yes		4/11/2018
Bedrooms	VARCHAR	Yes		4BR
#Bathrooms	VARCHAR	Yes		5BA
Summary	BLOB	Yes		Description
Square_Footage	VARCHAR	Yes		2500 SQFT
Asking_Price	INTEGER	Yes		$325,000.00
Photo 1	BLOB	Yes		Featured Image
Photo 2	BLOB	Yes		
Photo 3	BLOB	Yes		
Photo 4	BLOB	Yes		
Created_At	TIMESTAMP	Yes		
Updated_At	TIMESTAMP	Yes		

LISTINGS DATA TABLE

All new listings are added to this table, which functions as a tracker for managing portfolio data and views for each listing.



Listings - Permission - Logs Table

FIELDS	DATATYPE	NULL	KEY	EXTRA
Listings_ID	INTEGER	No	PRI	Auto_Inc
Portfolio_ID	INTEGER	Yes	FK	
Agent_ID	INTEGER	Yes	FK	
Listing_ID	INTEGER	Yes	FK	
Sessions	STRING	Yes	Hash	
Roles	ENUM	Yes		read_only, read_write
Created_At	TIMESTAMP	Yes		
Updated_At	TIMESTAMP	Yes		

OBJECT.RELATIONAL.MAPPING

Because Ruby objects don't map to the way databases store data, we use ORM to do this mapping. If you think of an application as having a front-end, middle-tier and back-end, then this concept falls within the logic of separating out the UI, logic and data in the application.

ACTIVE RECORD - DRY