Introduction to Hotwired ATS

Hello and welcome to Hotwired ATS! Hotwired ATS is a step-by-step guide to building modern Ruby on Rails applications using the Hotwire stack, CableReady, and StimulusReflex. This book is intended to give developers like you an approachable introduction to Hotwire and the surrounding ecosystem so that you can confidently build full-stack, reactive web applications with Ruby on Rails. My goal as a writer is for you to leave this book feeling prepared to use the Hotwire stack, CableReady, and StimulusReflex in your own applications.

Throughout this book, we will have opportunities to use Turbo Frames, Turbo Streams, Stimulus, CableReady, StimulusReflex, and Mrujs in a variety of situations. The application we build will be large enough to introduce a bit of complexity to the features we build while still being small enough to wrap our heads around. We will not build a commercial-grade, production-ready application in this book. Instead, we focus on finding ways to use the modern Rails stack to implement interesting features in an application larger than the typical toy applications used in most tutorials.

What are we building?

The application we will build together in this book is a stripped down version of an applicant tracking system, usually referred to as an ATS. An ATS is the software most HR teams use to post jobs, review applicants, manage their hiring process, and track their hiring results. An applicant tracking system is easiest to conceptualize as a specialized version of a CRM like Salesforce (some companies even use Salesforce as their ATS, poor souls). I chose to build an ATS in this book for two reasons.

First, if you are reading this book, you have probably used an applicant tracking system before. If you have ever applied to a job online, you were probably interacting with an applicant tracking system when you applied. If you are a hiring manager or interviewer at your company, you probably log in to an applicant tracking system to review candidates. The concepts of job postings and applications are about as universal as it gets. This basic level of understanding of what we are building is a nice place to build from.

Second, applicant tracking systems, like many business-facing applications, are basically just fancy spreadsheets. The core functions of an ATS, managing open positions and active applications, can be accomplished by a determined person in a spreadsheet. This need for spreadsheet-replacement applications, while not particularly exciting on the surface, opens up a set of shared problems that you will encounter in many of the applications you build in your career. Filtering lists of data, adding and removing resources, building charts and graphs, and sharing access to data are all common problems. Building an applicant tracking system allows us to use Rails to solve each of these problems.

Finally, for full disclosure, I spent ten years of my career building a large, widely-used applicant tracking system. I am comfortable with this particular problem space, which made it a little easier for me to put the pieces of the application together than it would have been to build something in a problem space I am less familiar with. We will not build anything near the level of complexity of a commercial applicant tracking system together, but we will build something that does a good impression of one.

What aren't we building?

As mentioned, this book is intended as a learning tool, not a commercial-grade application. As part of that, I eliminated as much as possible that was not directly related to learning more about Turbo, Stimulus, CableReady, or StimulusReflex. This means that the application we build will not be very usable on mobile devices — extra Tailwind classes do not contribute to our learning

goals — so we will work assuming that mobile can be safely ignored. In the same vein, the application we build in this book assumes that users have JavaScript enabled. This is generally a safe assumption when building a SaaS application for business users in the United States, but may not hold true for all applications and markets!

There will be 0 (zero) tests written in this book. I originally wrote the book with tests included throughout, but came to the conclusion that the significant amount of extra code was not contributing anything to the book's learning objectives. Writing tests does not help us learn more about how to use the core tools this book is interested in, so they were cut. I encourage you to write tests as you work if you feel they add to your learning experience, but we will not write them together in this book.

How should I use this book?

I'm going to preach for a minute. Apologies.

We are going to cover a lot of ground in this book, from `` `Rails new` `` to a reasonably complex application with public job postings, searching and filtering capabilities, inbound and outbound emailing, Stimulus controllers everywhere, and fancy charts. We will look at every line of code for every feature we build, and I will explain each change we make. There will also be moments where I explain more general concepts, discuss why I chose to use one tool over another, and go deeper into the details of a particular tool. There are even a few points where we build the same feature twice, so we can compare and contrast the options we have.

With all of that detail, and the quick pace that we move through each chapter, it may be tempting to start skimming the explanations and just copy and pasting the code without reading it. If you find yourself getting into copy and paste mode, try to pause, take a break, and come back when you are refreshed. The book will be most valuable if you take your time with it, reviewing new concepts and thinking about ways you can extend what we are building or adapt what we have built into your own projects. If you feel like you have

totally mastered a concept, feel free to move quickly and skim over the explanations, but try to resist the temptation, you never know when you might see something new or get a spark of inspiration.

There is a lot in this book, and we touch on a lot of pieces of the Rails ecosystem outside of the Hotwire stack, which makes the information density high. This is especially in the early chapters when all of the concepts are new. If you find yourself feeling overwhelmed by the amount of information coming at you, take a break and come back once the concepts you have already learned have had a chance to marinate. Reading this book is not a race, there are no prizes for finishing first! Please take your time and enjoy the journey.

Preaching over, a few practical notes on using this book now.

Most importantly, each chapter builds on the chapters before it — it is intended to be read like a book, starting with chapter one and ending with chapter eleven. This book will be much more valuable if you follow along chapter-by-chapter and build the application as you read. If you are a very experienced Rails developer, the first two chapters of the book are largely focused on setting up the application. While I encourage all readers to follow along with those chapters to make sure you have your bearings in the application, experienced Rails developers may wish to skim quickly through the first two chapters.

Each code block in the book has a copy button on the top right corner of the block. You can use that button to copy the whole block to your clipboard. I encourage you to copy and paste code blocks, especially big blocks of HTML. No one needs to type 15 Tailwind classes. Just be sure to read through the code blocks and the supporting text explaining the code after you copy them. To quickly navigate back to your place in a chapter, click on the section headings to get an anchor link directly to that section.

You can find the full code for each chapter on Github. Links are included in the book's table of contents and at the end of chapter. The links go directly to a PR with all of the changes in the chapter. If you get stuck or run into an error of some kind when following along, you can use the Github links to try to find

where things may have gone wrong. If you are truly stumped and cannot get a feature to work as outlined in the book, send me an email and I'll do my best to help: **david@colby.so**

As you follow along with each chapter, I recommend committing your changes regularly — you could commit as often as after finishing each section in a chapter. Being able to quickly rollback to a known good state is very helpful when you are reading along with unfamiliar code. Getting lost is easy, and git can help you get back to safety without losing much work.

Welcome again to Hotwired ATS, I am glad you are here. Let's start building!