Handling User Stories with Rails Scaffold, Custom Migrations, and ActiveStorage

Table of Contents

1. Introduction

This guide provides a step-by-step explanation for implementing user stories using Rails scaffold, custom migrations, and ActiveStorage. It is tailored to help developers set up and modify a Rails project to manage student profiles with specific features like email validation, profile picture uploads, and ActiveStorage integration.

2. Generating a Rails Scaffold for User Story 1: Creating a Student

- Scaffold Command
- Database Schema Generation
- Next Steps: Customizing Validations

3. Custom Migration for Email Validation

- Generating a Migration
- Modifying Migration File
- Running the Migration

4. Model Validations for User Stories 1 and 2

- Validating Fields: First Name, Last Name, Major, School Email, Graduation Date
- Uniqueness and Format of School Email

5. Setting Up ActiveStorage for Profile Pictures (User Story 3)

- Installing ActiveStorage
- Migrating the Database

6. Displaying the Profile Picture (User Story 3)

- Show View Logic: Displaying Uploaded or Default Image
- Image Variants and Resizing

7. Modifying StudentsController to Permit Profile Picture Uploads

- Strong Parameters for Profile Picture

8. Adding File Upload Field to Form for Profile Picture

- Updating Form for Profile Picture Input

9. Adding a Default Profile Picture

- Setting Default Image in Assets

10. Testing (TDD) for Student Model and Features

(looks better if you copy the last link and go to google drive)

What is TDD?

Definitions to know when using TDD

Types of tests

Benefits of TDD/Why use TDD

Best Practices/Principles for creating good tests

Scenarios Give When Then

How to get started?

Data Integrity

Fixtures

Test Database

Extra Resources

What is TDD?

- Test Driven Development, or TDD for short, is a software development method that involves writing tests. In the end it helps discover bugs earlier and aims to help deliver/write better quality code.

- TDD is not a testing method nor a design method, it's a development methodology.

Definitions to know when using TDD

Writing Failure Tests

- In other words, writing tests that will initially fail but later after adding some new functionality code they will pass

Refactoring

- Cleaning up the code essentially

More in depth

Types of Tests(TDD)

Automation test

- It allows for executing repetitive tasks, without a manual tester
- A tester writes scripts on their own and uses software or automation tools to test the software

Benefits

- Simplifies test case execution
- Improves reliability of tests
- Increases amount of test coverage
- Minimizing human interaction

Types of Automation Testing

- Unit testing
- Integration testing
- Performance testing

- Security testing
- Usability testing

Manual testing

- Uses the functions and features of an application
- A tester will then test on the application and test the predefined test cases

Benefits

- Fast and accurate feedback
- Less expensive
- Good for unplanned changes

Types of ManualTesting

- White box testing
- Black box testing
- Gray box testing

More Information on Testing

Benefits of TDD/Why use TDD

Benefits

- Early detected bugs
    - Reduces debugging time
    - Saves resources and time at the end of the day
- Improved code quality
    - Because tests are written before code so developers must think about what the code should be doing
    - Ensures that the models dont stop working when the code works, making the code clean and efficient
- Feedback as to what may not be working correctly

More benefits to TDD

Best practices/Principles for creating good tests

- Fast: Quick and easy to run tests based on your task(s)
- Independent: No test should have preconditions created by other tests
- Repeatable: Test behavior should not depend on external factors
- Self Checking: Each test should be able to determine whether is passes or fails
- Timely: Tests should be created/updated at the same time the code is being tested

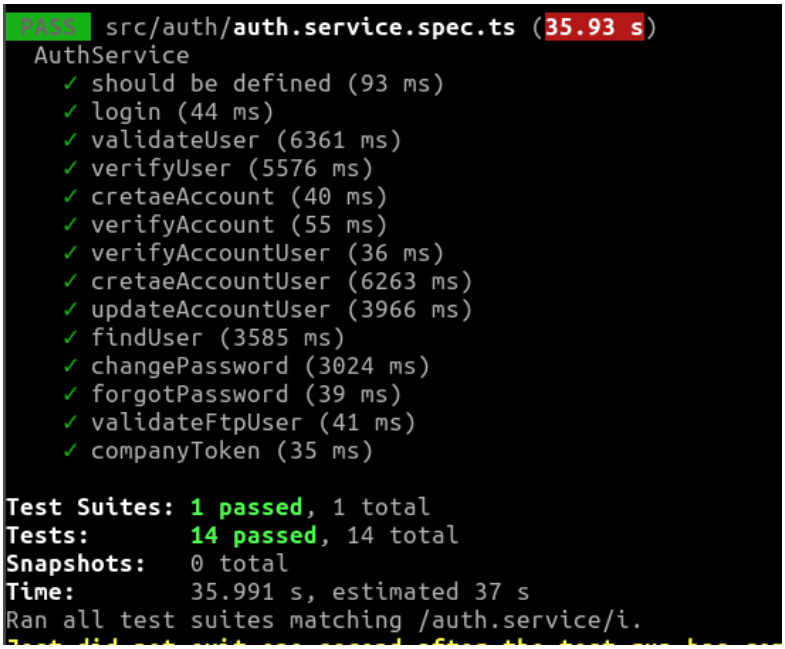
Scenarios Given When Then

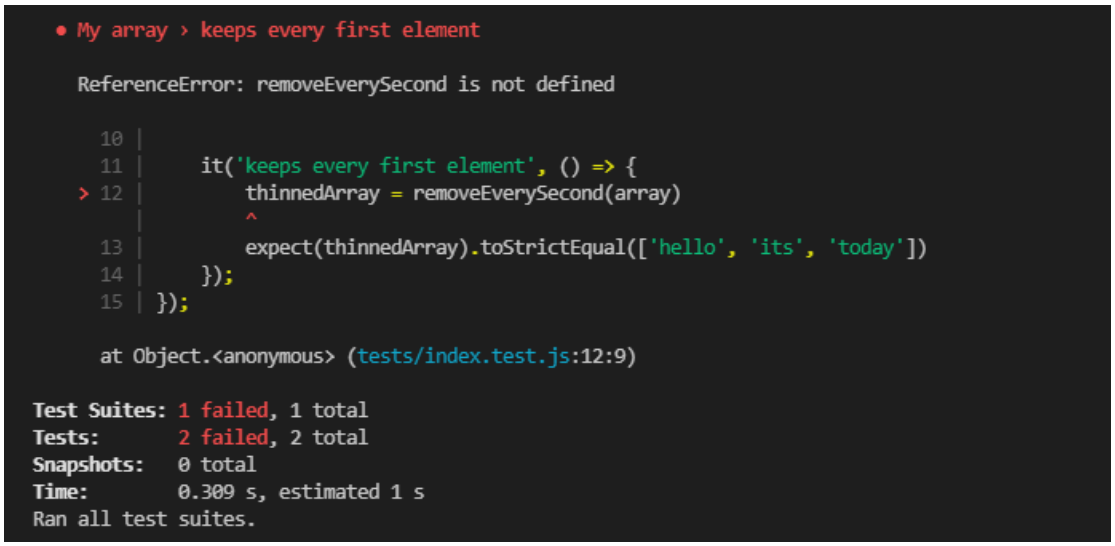
- This is the structure to define a flow of a test

Definitions of Given, When, and Then

- Given: The initial context/setup/preconditions
- When: The action/event that's being tested
- Then: The expected result of said test

When a test passes vs when it fails

- Passes: When a test passes that means that the code it working perfectly and is doing what it needs to do
  - 
```
PASS  src/auth/auth.service.spec.ts (35.93 s)
  AuthService
    ✓ should be defined (93 ms)
    ✓ login (44 ms)
    ✓ validateUser (6361 ms)
    ✓ verifyUser (5576 ms)
    ✓ cretaeAccount (40 ms)
    ✓ verifyAccount (55 ms)
    ✓ verifyAccountUser (36 ms)
    ✓ cretaeAccountUser (6263 ms)
    ✓ updateAccountUser (3966 ms)
    ✓ findUser (3585 ms)
    ✓ changePassword (3024 ms)
    ✓ forgotPassword (39 ms)
    ✓ validateFtpUser (41 ms)
    ✓ companyToken (35 ms)

Test Suites: 1 passed, 1 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        35.991 s, estimated 37 s
Ran all test suites matching /auth.service/i.
```
- Fails: When a test fails it could be for a number of things, such as-syntax errors, database/schema/app isn't updated, your wrote the test wrong, or it could be the actual code of what your testing
  - 
```
  ● My array › keeps every first element

    ReferenceError: removeEverySecond is not defined

      10 |
      11 |     it('keeps every first element', () => {
    > 12 |         thinnedArray = removeEverySecond(array)
         |         ^
      13 |         expect(thinnedArray).toStrictEqual(['hello', 'its', 'today'])
      14 |     });
      15 | });

      at Object.<anonymous> (tests/index.test.js:12:9)

Test Suites: 1 failed, 1 total
Tests:       2 failed, 2 total
Snapshots:   0 total
Time:        0.309 s, estimated 1 s
Ran all test suites.
```

    - To update existing rails application: “rails app:update” then run “rails db:migrate”

    - To update active storage: “rails active_storage:update” then run “rails db:migrate”

How to get started?

Definitions

- Run: Total number of individual tests executed
- Assertions: Are specific checks that are performed within a test. They are the actual test conditions that must pass for the test to be successful
- Errors: Something went wrong during the execution of a test
- Skips: This can happen for a number of reasons, best to check to see what your terminal says and ask Chat GPT why it got skipped

What file to be under

- In general it should be name_test.file-type

In our portfoilo_app it was:

- “student_test.rb”

Run a test command(in Rails)/How to run a test

- “rails test test/models/student_test.rb” - In our class example
- “rails test path/to/get/to/test.file” - In general(to run all tests)
- “rails test path/to/get/to/test.file:line_start_number” - To just run one test on a certain line

Test example

- This test should not be valid when a first_name is missing

#nil just means nothing/having no value or existance

```
test “should not be valid when first_name is missing” do
    student = Student.new(
        first_name: nil,
        last_name: “Smith”,
        school_email: “jane.smith@msudenver.edu”,
            major: “Mathematics”,
            minor: “Physics”,
```

graduation_date: “2025-06015”)

assert_not student.valid?, “Student should not be valid when the first_name is missing”

puts student.errors.full_messages

assert student.errors[:first_name].any?, “There should be an error on the missing first_name

end

Data Integrity

What is data integrity?

- Data integrity is the process of making sure that the data in the database is accurate, consistent, and complete.

Why is data integrity so important?

- Data integrity is important because it helps protect against data loss and leaks and makes sure that data is protected from unauthorized tampering

What are validations?

- Validations are rules applied to model fields to ensure that only valid data is saved to the database

Why are validations important?

- Validations are important because they will only allow valid data into the database, which means that the database will only have the correct data. This will save the company time, resources, and debugging later down the line.

Validation example from class

- This was on the student.rb file, “validates :first_name, last_name, :major, :minor, :graduation_date, presence: true” - All this is really saying is that the user has to put this information in before it will allow it to be put into the database and or continue with the sign up

Active record

What is an active record

- Allows you to validate the state of a model before it gets written into the database. Its an ORM(Object Relational Mapping) system, that connects an application’s objects to tables in a database

What does the active record do when testing a program

- Validates the model state, testing model associations, and performs CRUD operations
    - CRUD - Create, Read, Update, and Delete

Validation info

Fixtures

What is a fixture?

- A fixture is a predefined set of data that is loaded into your test database when running tests. They are very useful for static data!

Auto-loaded

- The fixtures are auto loaded into your test database and is give a unique key for every record

Accessible in Tests

- To access the fixtures in your tests all you have to do is call their key name

Isolation

- Every test should be isolated, as said above, but the fixture data is rolled back between every test(for a clean and usable environment)

Fixtures info

Test Database

Test isolation

- As said before tests need to be isolated and anything that goes with it, so it starts out with the same fixture data and then the database is reset to this state again before another test starts

Transactions

- The test has rolled back when the test finishes again making sure nothing else affects the next test, ensuring isolation

Repopulation fixtures

- Rails resets the database with the fixture data before every test, ensuring that no other test will be affected and again ensuring that the tests are isolated

Extra Resources:

Links in documentation

-https://medium.com/lets-do-it-pl/what-is-test-driven-development-why-tdd-is-so-important-e8ddade7010

-https://www.geeksforgeeks.org/types-software-testing/

-https://www.accelq.com/blog/tdd-test-driven-development/

-
https://www.computerhope.com/jargon/d/datavali.htm#:~:text=Validation%20or%20data%20validation%20means,compromised%20or%20corrupted%20during%20transmission.

-https://www.repeato.app/understanding-fixtures-in-programming/

Links not included in documentation

-https://docs.google.com/presentation/d/1BO_hqwvEsdOoaKSuv-PXKCOMtmn9FuDD8xoh2UiCS3U/edit#slide=id.g28aa5fbd04b_0_55

-https://guides.rubyonrails.org/active_record_basics.html

https://rubyonrails.org/
-(to open in google drive),
https://docs.google.com/document/d/15zMrT0DXyaHSPfKLPsGRfNVj3Rl-TQ2pxif4tx6B8_M/edit?usp=sharing

11. References

Hartl, M. (2022). *Ruby on Rails Tutorial: Learn Web Development with Rails* (7th ed.).

Rails Guides. (n.d.). *Active Storage Overview*. Rails Guides.

Rails Guides. (n.d.). *Getting Started with Rails*. Rails Guides.