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

**Table of Contents**

1. [Introduction](#_1._Introduction)
2. [Generating a Rails Scaffold for User Story 1: Creating a Student](#_2._Generating_a)
3. [Custom Migration for Email Validation](#_3._Custom_Migration)
4. [Model Validations for User Stories 1 and 2](#_4._Model_Validations)
5. [Setting Up ActiveStorage for Profile Pictures (User Story 3)](#_5._Setting_Up)
6. [Displaying the Profile Picture (User Story 3)](#_6._Displaying_the)
7. [Modifying StudentsController to Permit Profile Picture Uploads](#_7._Modifying_StudentsController)
8. [Adding File Upload Field to Form for Profile Picture](#_8._Adding_File)
9. [Adding a Default Profile Picture](#_9._Adding_a)
10. [Testing (TDD) for Student Model and Features](#_10._Testing_(TDD))
11. [References](#_11._References)

### 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**

To ensure the integrity of the data entered in the Student model, we implement model validations in Rails. These validations guarantee that key fields, such as first\_name, last\_name, major, school\_email, and graduation\_date, meet the acceptance criteria described in User Story 1.

* **First Name** and **Last Name**: We use the presence: true validation to make these fields mandatory. If either field is left blank, an error message will be triggered, and the record will not be saved.
* **Major**: Similarly, this field is required, and we use presence: true to ensure it is not left empty.
* **School Email**: This field requires more complex validation, as it needs to be both unique and correctly formatted. We validate its presence using presence: true, and we ensure uniqueness with uniqueness: true to prevent duplicate email entries in the system.
* **Graduation Date**: To validate this date field, we use presence: true to require its entry.

### <https://guides.rubyonrails.org/active_record_validations.html>

### 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

In User Story 3, students can upload a profile picture, which is handled via Active Storage. To add this functionality to the form, we need to modify the form view for the Student model.

First, ensure that Active Storage is set up in the app (refer to Section 5). Once that's complete, update the form to include an input field for file uploads:

<%= form\_with(model: @student, local: true) do |form| %>

<!-- Other form fields -->

<div class="field">

<%= form.label :profile\_picture %>

<%= form.file\_field :profile\_picture %>

</div>

<div class="actions">

<%= form.submit %>

</div>

<% end %>

This allows students to select and upload a profile picture when they create or update their profile.

### <https://guides.rubyonrails.org/active_storage_overview.html>

### 9. Adding a Default Profile Picture

If a student doesn't upload a profile picture, the application will display a default image. To implement this, we store a default image in the app’s assets folder and configure the Student model to return this image when no picture has been uploaded.

First, add a default image (e.g., default\_profile.jpg) to the app/assets/images folder.

Next, update the show.html.erb view to check whether the student has uploaded an image. If not, it will display the default:

<%= image\_tag @student.profile\_picture.attached? ? @student.profile\_picture : 'default\_profile.jpg' %>

This ensures that the profile always displays an image, even when the user hasn't uploaded one.

<https://gorails.com/forum/how-do-i-set-and-use-a-default-image-with-active-storage>

10. Testing (TDD) for Student Model and Features

* Test Setup: Student Fixtures
* Key Tests for Validation, Uniqueness, and Saving

### 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](https://guides.rubyonrails.org/active_storage_overview.html).

Rails Guides. (n.d.). *Getting Started with Rails*. [Rails Guides](https://guides.rubyonrails.org/getting_started.html).