EXP. NO:11

IMPLEMENTING TDD RULES USING RUBY ON RAILS

Aim:

To implement TDD rules(Red ,Green ,Refactor) to develop a typical model code using Ruby on Rails framework.

Test-Driven Development (TDD):

Definition:
Test-Driven Development (TDD) is a software development methodology where tests are written before the actual code is implemented. It follows a cycle of **Red-Green-Refactor**:

1. **Red:** Write a test that fails because the functionality does not yet exist.
2. **Green:** Write the minimum code necessary to make the test pass.
3. **Refactor:** Refactor the code to improve its structure and readability while ensuring that all tests still pass.

Benefits of TDD:

- Ensures code correctness from the beginning.
- Encourages modular and maintainable code.
- Helps catch bugs early in the development cycle.

Ruby on Rails:

Definition:
Ruby on Rails (RoR) is a web application framework written in Ruby. It follows the Model-View-Controller (MVC) pattern and emphasizes convention over configuration, making it easier and faster to develop robust web applications.

Features of Ruby on Rails:

- **MVC Architecture:** Separation of concerns for cleaner code.
- **Active Record:** Simplifies database interactions.
- **Convention over Configuration:** Reduces the need for explicit configuration.
- **Gems:** Extends functionality using a rich library ecosystem.
- **Scaffolding:** Generates boilerplate code quickly.

Installing Ruby on Rails

Prerequisites

- Ruby (version 3.3.0 or later is recommended).
- A database like SQLite (default), PostgreSQL, or MySQL.

Steps to Install Rails

1. **Install Ruby: sudo apt install ruby-full(#on Ubuntu)**
   Download Ruby from the official site or use a version manager like RVM or rbenv.
2. **Install Bundler: gem install bundler**
   Bundler manages gem dependencies for your projects.
3. **Install Rails: gem install rails**
   Install Rails using the gem command.
4. **Verify Installation: rails -v**
   Confirm that Rails is installed correctly.
5. **Set up a new Rails project:**
   Create a new Rails application:
   ```
   rails new prime_number_generator
   cd prime_number_generator
   ```

Implementation of the Prime Number Generator

1. RED Phase

In the RED phase, we write tests first. At this stage, no implementation exists, so the tests will fail.

Test Code

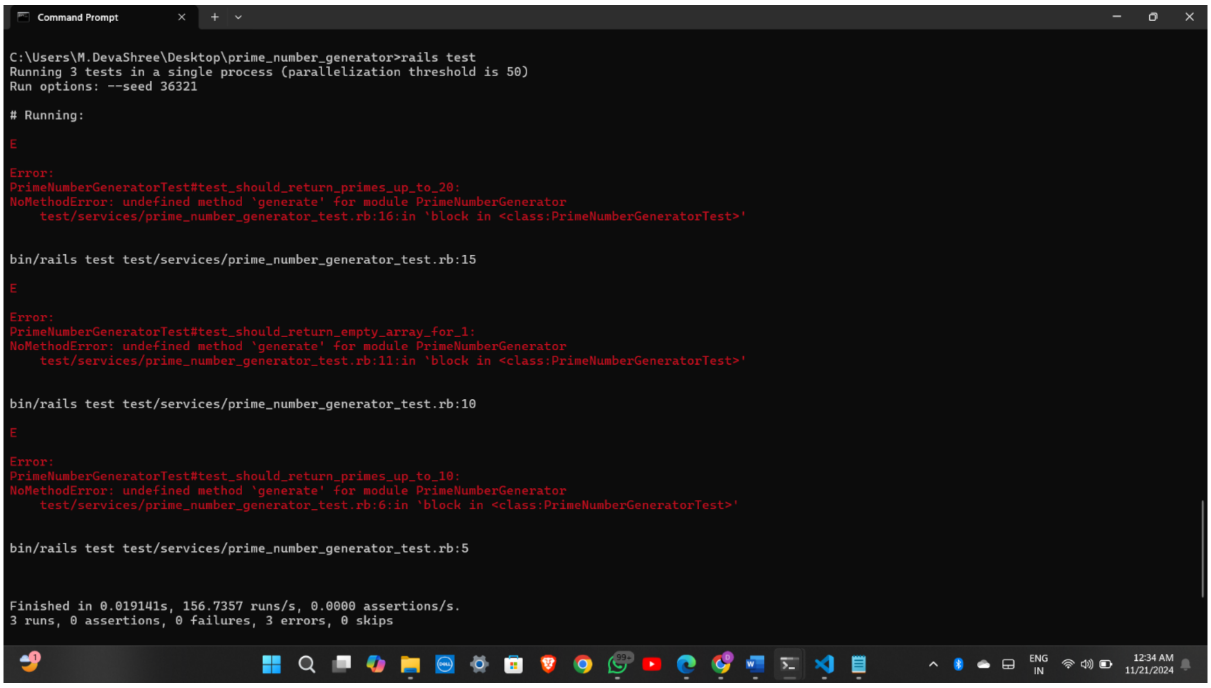
Create the test file: test/services/prime_number_generator_test.rb.

```
require 'test_helper'

class PrimeNumberGeneratorTest < ActiveSupport::TestCase
  test "should return empty array for 1" do
    assert_equal [], PrimeNumberGenerator.generate(1)
  end

  test "should return primes up to 10" do
    assert_equal [2, 3, 5, 7], PrimeNumberGenerator.generate(10)
  end

  test "should return primes up to 20" do
    assert_equal [2, 3, 5, 7, 11, 13, 17, 19], PrimeNumberGenerator.generate(20)
  end
end
```

Run the Tests

Execute the tests to confirm failure: rails test

OUTPUT:



```
C:\Users\M.DevaShree\Desktop\prime_number_generator>rails test
Running 3 tests in a single process (parallelization threshold is 50)
Run options: --seed 36321

# Running:

E

Error:
PrimeNumberGeneratorTest#test_should_return_primes_up_to_20:
NoMethodError: undefined method `generate' for module PrimeNumberGenerator
    test/services/prime_number_generator_test.rb:16:in `block in <class:PrimeNumberGeneratorTest>'


bin/rails test test/services/prime_number_generator_test.rb:15

E

Error:
PrimeNumberGeneratorTest#test_should_return_empty_array_for_1:
NoMethodError: undefined method `generate' for module PrimeNumberGenerator
    test/services/prime_number_generator_test.rb:11:in `block in <class:PrimeNumberGeneratorTest>'


bin/rails test test/services/prime_number_generator_test.rb:10

E

Error:
PrimeNumberGeneratorTest#test_should_return_primes_up_to_10:
NoMethodError: undefined method `generate' for module PrimeNumberGenerator
    test/services/prime_number_generator_test.rb:6:in `block in <class:PrimeNumberGeneratorTest>'


bin/rails test test/services/prime_number_generator_test.rb:5


Finished in 0.019141s, 156.7357 runs/s, 0.0000 assertions/s.
3 runs, 0 assertions, 0 failures, 3 errors, 0 skips
```

2. GREEN Phase

In the GREEN phase, we write the minimal code necessary to make the tests pass.

Implementation Code

Create the file: app/services/prime_number_generator.rb.

```
class PrimeNumberGenerator
  def self.generate(limit)
    []
  end
end
```

3. REFACTOR Phase

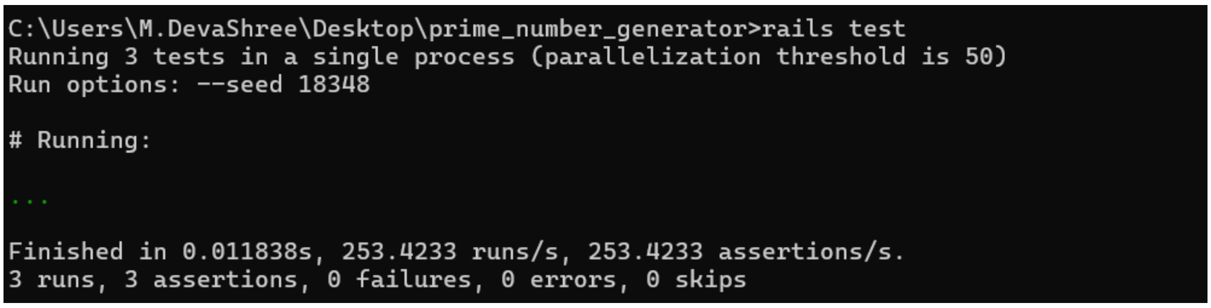
In the REFACTOR phase, we improve the implementation incrementally while ensuring all tests pass.

Step 1: Add Prime Logic

Update prime_number_generator.rb:

```
class PrimeNumberGenerator
  def self.generate(limit)
    primes = []
    (2..limit).each do |num|
      primes << num if is_prime?(num)
    end
    primes
  end

  private

  def self.is_prime?(num)
    return false if num < 2
    (2..Math.sqrt(num)).none? { |i| num % i == 0 }
  end
end
```

OUTPUT:



```
C:\Users\M.DevaShree\Desktop\prime_number_generator>rails test
Running 3 tests in a single process (parallelization threshold is 50)
Run options: --seed 18348

# Running:

...

Finished in 0.011838s, 253.4233 runs/s, 253.4233 assertions/s.
3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```

Description	Allotted Marks	Obtained Marks
Preparation	20	
Design/Implementation	20	
Viva	15	
Output	10	
Record	10	
Total	75	

RESULT:

Thus, the implementation of TDD(red,green and refactor)rules using ruby on rails has been done successfully.