Ruby on Rails

- Server-side web application framework written in Ruby
- Model-view-controller (MVC) framework
- Providing default structures for a database, we pages and services
- Encourages and facilitates the use of web standards such:
    a. JSON or XML for data transfer
    b. HTML, CSS and JavaScript for display and user interfacing
- Emphasizes the use of other well-known software engineering patterns and paradigms

History:

- David Heinemeier Hansson removed Ruby on Rails from his work on the project management tool Basecamp at the web application

JULY 2004 and FEBRUARY 2005

- He releases Rails as an open source but did not share commit rights to the project until the year of 2005 in February.

AUGUST 2006 and OCTOBER 2007

- Ruby on Rails reached a milestone when Apple announced that it would ship this framework with Mac OS X v10.5 “Leopard”, which was in October 2007.

MARCH 15, 2009

- They’ve released a new version of Rails which is Rails version 2.3 with major new developments in templates, engines, Rack and nested model forms.

DECEMBER 23, 2008

- Merb is another web application framework and announced it would work with the Merb project to bring the best ideas of it into Rails 3 ending the unnecessary duplication across both communities and this framework was merged with Rails.

Rails 3.1: Released on August 31, 2011, featuring Reversible Database Migrations, Asset Pipeline, Streaming, jQuery as default JavaScript library and newly introduced CoffeeScript and Sass into the stack.

Rails 3.2: Released on January 20, 2012 with a faster development mode and routing engine (also known as Journey engine), Automatic Query Explain and Tagged Logging. Rails 3.2.x is the last version that supports Ruby 1.8.7. Rails 3.2.12 supports Ruby 2.0.

Rails 4.0: Released on June 25, 2013, introducing Russian Doll Caching, Turbolinks, Live Streaming as well as making Active Resource, Active Record Observer and other components optional by splitting them as gems.

Rails 4.1: Released on April 8, 2014, introducing Spring, Variants, Enums, Mailer previews, and secrets.yml.

Rails 4.2: Released on December 19, 2014, introducing Active Job, asynchronous emails, Adequate Record, Web Console, and foreign keys.

Rails 5.0: Released on June 30, 2016, introducing Action Cable, API mode, and Turbolinks 5.

Rails 5.0.0.1: Released on August 10, 2016, with Exclusive use of rails CLI over Rake and supports Ruby 2.2.2+ versions

Rails 5.1: Released on April 27, 2017, introducing JavaScript integration changes (management of JavaScript dependencies from NPM via Yarn, optional compilation of JavaScript using Webpack, and a rewrite of Rails UJS to use vanilla JavaScript instead of depending on jQuery), system tests using Capybara, encrypted secrets, parameterized mailers, direct & resolved routes, and a unified form_with helper replacing the form_tag/form_for helpers.

Why Ruby?

Ruby originated in Japan and now it is gaining popularity in US and Europe as well. The following factors contribute towards its popularity:

- Easy to learn
- Open source (very liberal license)
- Rich libraries
- Very easy to extend
- Truly object-oriented
- Less coding with fewer bugs
- Helpful community

Although we have many reasons to use Ruby, there are a few drawbacks as well that you may have to consider before implementing Ruby:

1. Performance Issues − Although it rivals Perl and Python, it is still an interpreted language and we cannot compare it with high-level programming languages like C or C++.
2. Threading model − Ruby does not use native threads. Ruby threads are simulated in the VM rather than running as native OS threads.

Rails Strengths

Rails is packed with features that make you more productive, with many of the following features building on one other.

1. **Metaprogramming**

Where other frameworks use extensive code generation from scratch, Rail framework uses Metaprogramming techniques to write programs. Ruby is one of the best languages for

Metaprogramming, and Rails uses this capability well. Rails also uses code generation but relies much more on Metaprogramming for the heavy lifting.

2. Active Record
Rails introduces the Active Record framework, which saves objects into the database. The Rails version of the Active Record discovers the columns in a database schema and automatically attaches them to your domain objects using metaprogramming.

3. Convention over configuration
Most web development frameworks for .NET or Java force you to write pages of configuration code. If you follow the suggested naming conventions, Rails doesn't need much configuration.

4. Scaffolding
You often create temporary code in the early stages of development to help get an application up quickly and see how major components work together. Rails automatically creates much of the scaffolding you'll need.

6. Built-in testing
Rails creates simple automated tests you can then extend. Rails also provides supporting code called harnesses and fixtures that make test cases easier to write and run. Ruby can then execute all your automated tests with the rake utility.

7. Three environments
Rails gives you three default environments: development, testing, and production. Each behaves slightly differently, making your entire software development cycle easier. For example, Rails creates a fresh copy of the Test database for each test run.