**Portfolio(Ruby On Rails) Project Report**

 We recently completed a project implemented with Ruby on Rails. This is a writeup of the tools and strategies we used and what we learned in the development process of ruby on rails application.

The project is developed under the guidance of Prof. Nirmal Goud sir.

Submitted To: By:

Prof. Nirmal Goud Mr. Indra Kumar Mhaski

Mr. Sahil parte

**Online Personal Portfolio**

**INDEX**

* Acknowledgement.
* Introduction.
* Features Of Application(SRS).
* Detaild ER Diagram.
* Deployment.
* Conclusion.

**ACKNOWLEDGEMENT**

We gratefully acknowledge for the assistance, cooperation, guidance and clarifications provided by Prof. Nirmal Goud sir during the development of the OnlineExams System website. Our extreme gratitude to Mr.Nirmal Goud sir who guided us throughout the project. Without his willing disposition, spirit of accommodation, frankness, timely clarification and above all faith in us, this project could not have been completed in due time. His readiness to discuss all important matters at work deserves special attention. We would also like to thank whole of the faculty of the college for their cooperation and important support.

We would also like to say special thnks to Mr Jordan Hudgens whome I learnt how to develop web application using Ruby On Rails framework, GitHub and Agile methodology of software development.

**INTRODUCTION**

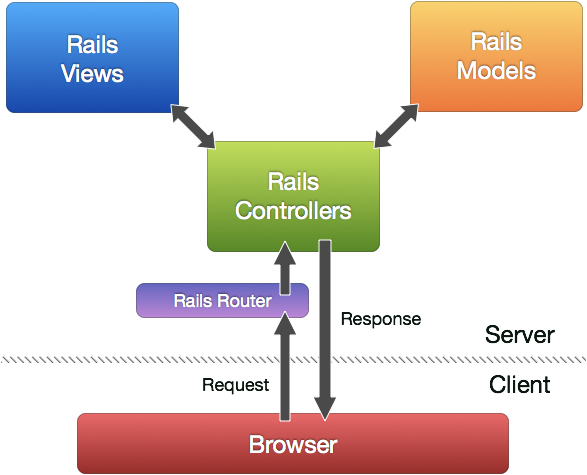
This chapter will give a short explanation of some abbreviations and keywords, which will be used further in the report. The information included in this section aims to make reading this report easier to follow by the reader.

**Web Application:**

In software engineering, a web application or web-app is an application that is accessed with a web browser over a network such as the Internet. It is a computer software application, coded in a browser-supported language such as HTML and Javascript, and reliant on the web browser to render the application executable. Web applications are popular because the browser acts as a client, sometimes called a thin client. The ability to update and maintain Web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity. Web applications are used to implement many different kind of tasks such as web mails and calendars, online shops and auctions, currency converters and etc.

**Web Framework:**

A web application framework is a software framework designed to support the development of dynamic websites, web applications and web services. Web application frameworks facilitate rapid application development by allowing the programmer to define a high-level description of the program. Many frameworks provide libraries for database access, template and session management, and often promote code reuse. One architectural pattern of a web framework is the Model View Controller (MVC) architecture, designed to separate the data model with business rules from the user interface. Most MVC frameworks follow a push-based architecture. These frameworks use actions that do the required processing, and then "push" the data to the view layer to render the results. Ruby on Rails and Spring MVC are good examples of this architecture. An alternative to this is pull-based architecture, sometimes also called "component-based". These frameworks start with the view layer, which can then "pull" results from multiple controllers as needed. In this architecture, multiple controllers can be involved with a single view. Below is a figure which visualizes the MVC architecture.



**Don't Repeat Yourself:**

Don't Repeat Yourself or DRY is a process philosophy aimed at reducing duplication. The philosophy emphasizes that information should not be duplicated, because duplication increases the difficulty of change, may decrease clarity, and leads to opportunities for inconsistency. DRY code is created by data transformation, which allows the software developer to avoid copy and paste operations. DRY code usually makes large software systems easier to maintain and becomes increasingly important in applications that use multi-tier architectures.

**Convention over Configuration:**

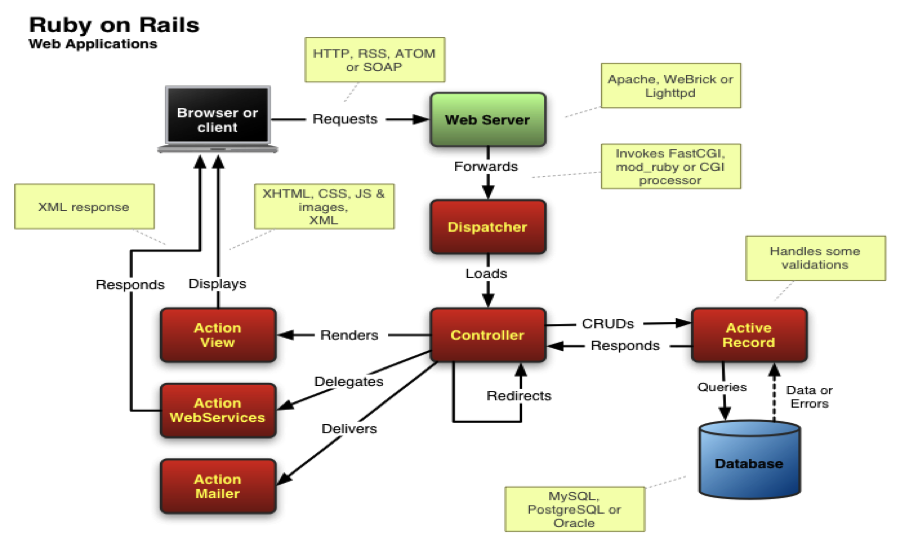
Convention over Configuration or CoC is a software design paradigm which seeks to decrease the number of decisions that software developer needs to make, gaining simplicity, but not necessarily losing flexibility. CoC essentially means a developer only needs to specify unconventional aspects of the application. For example, if there's a class User in the model, the corresponding table in the database is called users by default.

**Ruby:**

Ruby is a dynamic, open source script programming language with a focus on simplicity and productivity. It is object-oriented, platform-independent and supports multi-threading. In Ruby, every data type is an object, every function is a method. Every bit of information and code can be given their own properties and actions. Ruby has exception handling features, like Java or Python, which makes it easy to handle errors. The syntax of Ruby is broadly similar to Perl and Python. Since its public release in 1995 (Ruby 0.95), Ruby has drawn devoted coders worldwide. In 2006 (Ruby 1.8.6), Ruby achieved mass acceptance. The latest stable version of Ruby (Ruby 1.9.1) was released on 30 January 2009. It introduces a lot of changes and it is nearly two times faster compared to version 1.8.6.

**Ruby On Rails Framework:**

Ruby on Rails, RoR or just Rails is a web development framework written in Ruby language. It is designed to make programming web applications easier. Rails makes it possible for the programmer to write less code, while accomplishing more than many other languages and frameworks. Rails is organized around the Model, View, Controller architecture. MVC benefits for isolating the business logic from the user interface, keeping programming code DRY and making it clear where different types of code belong for easier maintenance of the application. Below is the diagram which describes the Ruby On Rails architecture.



**Features Of Application**

**Technical Information:**

This project is going to be created in Ruby On Rails web framework which is written in Ruby programming language, we will be using various gems available for Rails like Devise, Petergage, Kaminari etc. We will use MySql as a database for our backend development, the project will also use JavaScript frontend libraries like HTML sortable and Jquery, we will be using Git and GitHub as our version control system.

**Features of the application:**

The main objective of our application is to provide information about the author(in this case me) to the visitors of the portfolio, in order to achieve this goal as an author of the site I will need following features to be there in the application.

1. As an author, I should be able to manage the content of my portfolio page.
2. As a visitor anyone should be able to watch my portfolio page.
3. As an author, I should be able to post and manage my work sample examples on site.
4. As a visitor, anyone should be able to watch and comment on my work sample examples on site.
5. As an author, I should be able to post and manage my blogs.
6. As a visitor, anyone should be able to watch and comment on my blogs.
7. There should be a contact page in the site so that any visitor can easily get information about how to contact me.
8. Before making a comment a user must be authenticated either via Devise or Google.

**Technologies which will be used in the project:**

* **Front-End**

1. HTML.
2. CSS.
3. JavaScript.
4. BootStrap.
5. JQuery.
6. Embedded Ruby.
7. HTML Sortable.
8. Font Awesome.

* **Back-End**

1. Ruby On Rails.
2. MySql as a Database.
3. Puma as a web server.

* **Gems**

1. MySql2. 6. BootStrap.
2. OmniAuth. 7. Redis.
3. Cocoon. 8. Kaminari.
4. Devise. 9. Font-Awsome-Rails
5. CarrierWave.

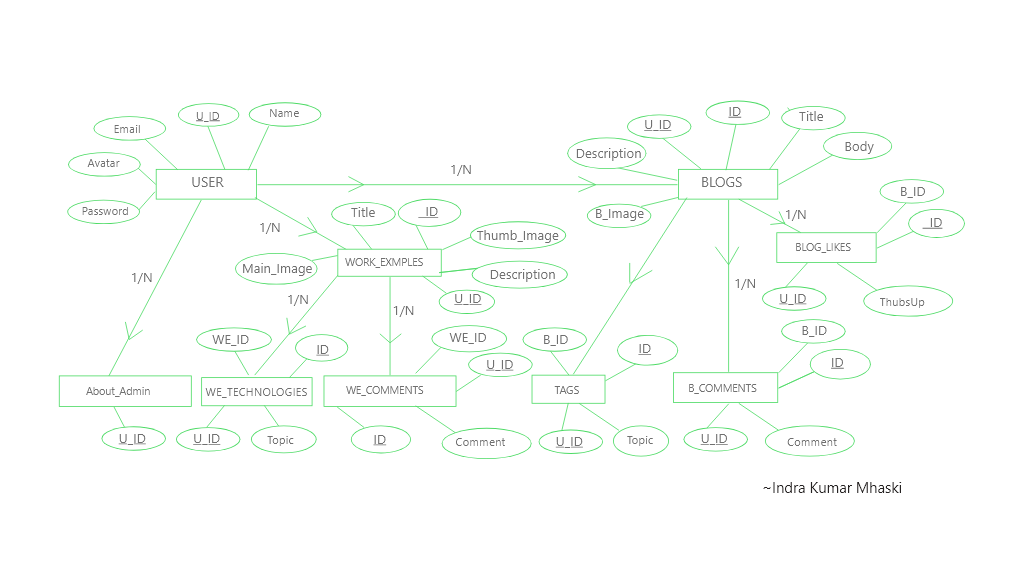
**Security Features of Application:**

The application will be secured by the Devise and Google authentication the passwords of the users will be secured by the Bcrypt encryption that means in any case no one will be able to get the password and secret information of users.

More details of SRS can be found at here: <https://github.com/indrakumarmhaski/MyRailsPortfolio/blob/master/Indra%20Portfolio%20SRS.pdf>

**Detaild ER Diagram**

**Description:** Here we are presenting the detailed ER diagram of our project.



**Let us discuss each entity and its attribute in detail:**

* **User:**

User is a primary entity of our application basically there will be two types of users first is the admin whose portfolio is going to be created(in this case me) other users will be visitors who are going to visit our application, it means we have following two cases:

1. When a user is an admin: In this case, the user will have all the administrative powers. for example, he will be able to delete any comment on his work example and blogs, he will also be able to post new blogs and work examples on site.
2. When a user is a normal visitor: In this case, the user should only be able to view all the content of the application and make comment on blogs and work examples.

In both the cases a user will have the following attributes:

1. Name: String.
2. Id: Integer.
3. Email: String.
4. Encrypted\_password: String.
5. Avatar: String.

* **Blogs:**

If a user is an admin then he can have n number of blogs and that blog can have n number of comments and likes on that particular blog but for now we are discussing what attributes a blog should have? The answer is following attributes a blog should have:

1. Title: String.
2. Description: Text.
3. Body: Text.
4. Id: Integer.
5. Blog \_Image: String.
6. Topics: String.

* **Blog\_Comments:**

This table of the database is going to be used for storing the comments of blogs, in order to accomplish this goal the table should have the following attributes:

1. Blog\_Id: Integer.
2. User\_Id: Integer.
3. Id: Integer.
4. Body: Text.

* **Blog\_Likes:**

This table will contain the likes on a particular blog, it will have the following attributes:

1. Blog\_Id: Integer.
2. User\_Id: Integer.
3. Id: Integer.
4. Thums\_Up: Integer

* **Work\_Examples:**

This table will contain the work examples of an admin and will have the following columns:

1. Thumb\_Image: String.
2. Main\_Image: String.
3. Id: Integer.
4. User\_Id: Integer.
5. Title: String.
6. Description: Text.

* **Work\_Example\_Technologies:**

This table of database belongs to Work\_Examples table and it will contain following attributes:

1. Id: Integer.
2. Work\_Example\_Id: Integer.
3. User\_Id: Integer.
4. Topic.

* **Work\_Example\_Comments:**

In this table, we will store the comments of users on author’s work examples, this table will have the following attributes:

1. Id: Integer.
2. Work\_Example\_Id: Integer.
3. User\_Id: Integer.
4. Comment: Text.

* **About\_Admin:**

This table is going to be used for storing data of the admin which can change frequently for example ‘about’, ‘education’ etc. This table will have the following attributes:

1. Id: Integer.
2. User\_Id: Integer.
3. About: Text.
4. Education: Text.

**NOTE:** The attribute’s data type are not given in standard SQL’s data types, rather than they are Ruby On Rails data types.

**Deployment**

When comes to deployment of a Ruby On Rails application there are many hosting companies which offers bets services, Digital Ocian and Heroku are among best of them.

For the testing purposes and just to get the app running without paying anyting to any hosging componey Heroku provides this opurtunity to the developers, but they also remove your every kind of heavy data from their server in order to keep running their server smoothly and they also put your application on sleep mode if no one is using it these are some of the limitations of free hosting of application on Heroku but still for the testing purpose and keeping the portfolio online without paying anything we have used the free hosting service of the Heroku, you can walk through the online portfolio at:

https:// indraportfolio.herokuapp.com

**Conclusion**

At the end, in conclusion I would like to talk about the development journey of this application, it took around twenty days to build out this application and throughout the development I used GitHub as a version control system of the app you can find the project source code at <https://github.com/indrakumarmhaski/MyRailsPortfolio>

We made 139 commits over 24 different feature branches in the development of this application these numbers are enough to show the important of version control system in the software industry.

We also used PivotalTracker for the agaile development of the application you can find the project at <https://www.pivotaltracker.com/projects/2190716> this link of application on PivotalTracker is also enough to demonstrate how we can develop a project using agaile methodology.