***Technology and Infrastructure***

As previously shown, we have used a sitemap and a basic user interface design to demonstrate basic workings of our planned website. Now, we are going to talk about the Rails framework and architecture we will be using in construction of our web application. In addition to using Ruby on Rails, we will be using JavaScript (jQuery and plug-ins) and CSS for designing the eye-pleasing pages as well as built-in interactivity to our pages. Moreover, most user activity will stay on the same page thanks to Ajax.

First off, I am going to give an overview of a Rails application when a user types “**rails new app\_name**” and creates a new Rails application. This will generate a bunch of files in a directory under the specified application name.

***Overview of File Structure of Rails Application***

App\_name (main directory/ ones bolded indicated most important aspects)

1. **app (contains assets folder -> stylesheets, javascripts, images)**
2. bin
3. **config (for the report, only routes.db is discussed)**
4. **db (migrations)**
5. lib
6. log
7. public
8. test (for testing models)
9. tmp
10. vendor
11. other files (Gems etc. )

Under the app folder inside your application directory, there are 6 folders: assets, controllers, helpers, mailers, models and views. And in this folder, most of your code is written here. For front–end code such as Javascript and CSS, they can be written separately and stored inside the assets directory. For ruby code, most of the code will be written in models, controllers and views. Obviously, the Rails framework adopts the model-view-controller (MVC) approach by allowing users to separate tasks under this 3 broad categories. Another important folder is the config folder which stores files that allows users to route different pages together using ruby code.

***Database Management***

For database management, like performing CRUD on the selected database, this is done primarily through the db folder, which houses the migrations folder. In Rails, actions on the database can be performed using “migrations”. For instance, if the user wants to add a column to a table, he can alter the migration file and call “rake db:migrate” to update the database with the new migration file. Through migrations, the user can avoid having to write nasty SQL statements to change or modify certain parts of the database.

One particular thing about rails, if you want to perform queries on the database, you can simply call “rails console” on your command prompt and it will give you a console where you can query the database using object oriented language. So if you want to check a record in the User table, you can just type : “User.find(1)” , to find a record where the user id is 1.

***Models***

Another important thing about rails is that tables are called models (ruby classes). To create a model in Rails, the user can call “rails g model **model\_name** [field\_name:field\_type]” to construct a class in the models folder and a migration file which specifies how the table should be created in the database. As mentioned previously, any desired modification to the table can be done through the migration file.

For relationships between entities in database modeling, they are done through the models in Rails, where special ruby syntax is used to specify many-to-many or other types of relationships.

To view all completed models after migrating, the db folder stores the schema.rb file which displays all ruby classes that were generated so that the user can keep track of all migrations regarding built tables.

***Views and Controllers***

The most important part of the Rails application is the controller folder, which contains the controller files which interacts with models and views of the application.

To understand how the controller works, we must also understand how html files are stored in Rails. In a Rails application, html code are stored in separate folders under a **views** folders. The folders named here are related in name to ruby files stored in the **controller** folder.

Eg. A controller file named media\_controller.rb has all the actions required by all html pages stored in the media folder (under views folder).

Controllers

* media\_controller.rb

Views

* media
  + index.html.erb
  + new.html.erb

Each html form inside Rails must correspond to a method inside the controller. So for every file inside the media folder, the prefix matches index and new methods located inside media\_controller.rb.

So how do they interact? For example, if you want to submit a form from a html page, the values from the form are passed to the controller method (called an action in Rails) specified by the route inside routes.rb in the config folder. The method processes the values from the form and redirects the user accordingly. Since it is a class method, it is up to the creator to modify what it does.

When a controller needs to interact with a model, a method inside the controller file creates an object of that model class which is inside the models folder. Models basically serve as classes which are used inside controllers.

For html.erb files that requires javascript files, the <%= javascript\_include\_tag “some\_file.js” %> tag includes the appropriate javascript file from assets directory without needing the user to specify its path. Similarly for CSS files, the <%= stylesheet\_link\_tag “some\_file.css” %> does the same.

The clever use of routes and controller methods allows information to be passed from model to controller and to views and vice-versa.