CVWO Assignment 1 – Not Your Typical Bloggging Assignment

Wong Kang Fei
A0138862W
kfwong@comp.nus.edu.sg

Main Objective

- Acquire basic programming skills for typical software development cycle through developing a website.
- Learn about best practices and coding techniques.
- Familiar with SQL and other development tools.

Personal Objective

- Develop using Laravel framework for the project. AngularJs for front-end if time permit.
- Deployment of the application to AWS/Openshift.
- Maintain a git repository with proper branching techniques used in typical team workflow.

Analysis of Project Requirement

- **Why I choose to use Laravel in this assignment**
  - It was originally my summer break self-learning objective. CVWO assignment gave me a reason to test out the framework and develop a practical example.
  - Popular framework use by companies and startups, mastering it would be great addition to my skillset
  - Streamline development workflow, dependency management (Composer) makes it so much easier to maintain external library.
  - MVC, Eloquent ORM, and other awesome feature will be mentioned in subsequent use cases.

- **Development Environment**
  - Arch Linux
  - PhpStorm IDE & Sublime Text
    - Sick of Eclipse
    - Personal preference is to use IDE for large scale development and analysis, text editor for quick and minor changes.
  - Laravel 5.1
    - Laravel includes a minimal webserver for testing, so Apache Webserver is unnecessary for this assignment.
    - Laravel enforces MVC pattern, so the project structure should naturally modular.
  - MariaDB
    - Eloquent ORM will be used in conjunction with Laravel.
    - Laravel provides Migration & Seeding functionality, defining tables & database population is a breeze.

- **Analyzing Project Use Case**
  - **Back-End**
    - Authentication and Membership is handled by Laravel out-of-the-box. Understanding the API and the service request cycle should be enough to handle the requirement.
    - Hashing of password is handled in Migration definitions.
    - Registration & changing password is optional, but it's provided in the core, a little bit of touch up should do the trick.
    - All URI should comply with RESTful definition, following Laravel's route naming convention.
    - Prevent CSRF attack with tokens. (Laravel has plugin to simplify the process)
  - **Documentation**
    - README will be provided in the git repository to guide the installation.
    - Consider create a bash script to setup environment quickly. (startup.sh)
    - Schema description is self-explanatory through migration definition. (Under Project>database>migrations)
  - **Front-End**
    - Expected to use Bootstrap for faster prototyping.
    - Either Jquery or AngularJS can be used, if time permits.
    - Laravel provides Blade templating engine to quickly set up a view and seperate them from controller's logic.
  - **Versioning**
    - git repository setup at Github. (https://github.com/kfwong/cvwo-assignment-1)
    - I would try to follow the suggested teamworkflow here if possible, https://mockupstogo.mybalsamiq.com/projects/diagrams/Git%20Workflow. As for initial stage of solo development might not be necessary to create so many branch, will adjust accordingly as project grows.

Deployment Plan

- First choice would be Amazon AWS, alternatively RedHat OpenShift.
- AWS gives me more flexible configuration environment, while Openshift is restrictive on the software installed on the platform.
- My previous experience with Openshift was great, despite SSH accessing is slow. Deployment is easy with simply git-push. I do not have to worry about database configurations and environment settings, since it's PaaS managed.
- I choose AWS over Openshift because I need a faster SSH interface, also reliablity (Openshift hibernate the nodes in free-tier if the server there's no traffic, subsequent first access takes longer time.)
- Auto-deployment & testing with Jenkins, good to have but rather overkill in this assignment. Will explore the last if time permitted.

Use Case Diagram

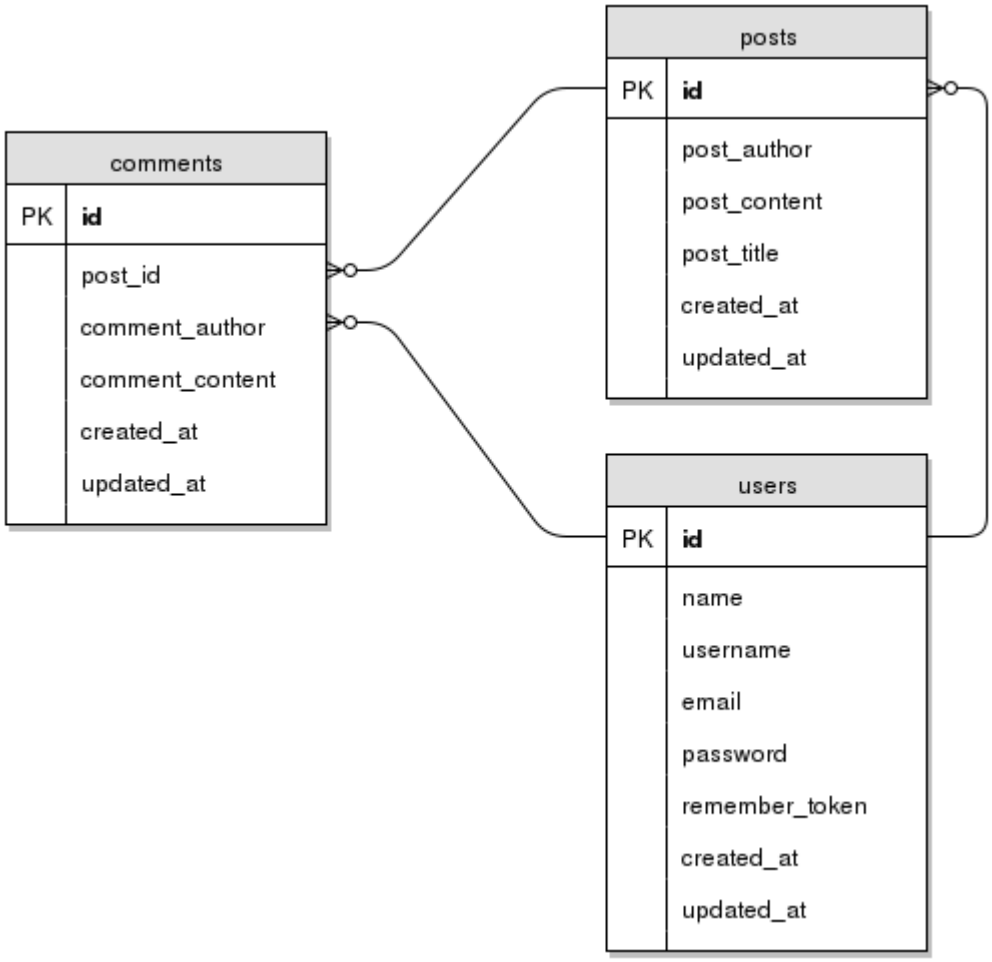
* Writer/Member are only able to perform Update/Delete on posts that created by him/herself.
** Writer/Member can only perform Update on his/her personal profile.

Wong Kang Fei
A0138862W
kfwong@comp.nus.edu.sg

Use Case Description (Informal)

- I've decided to add a few more use cases into the application as my personal sidequest to explore the functionality of Laravel.
  - Membership Registration
    - Actor: Guest
    - Guest can register an account.
    - Goal: Explore Laravel built-in Authentication/Authorization feature, and Socialite plugin that integrate with 3rd party social platform (eg: facebook, google+, twitter).
  - Search Content
    - Actor: Writer, Guest
    - Writer or Guest can search content by entering keywords.
    - Goal: To explore Eloquent API in-depth. Alternatively, my initial research points me to Solr or Larasearch.
  - Comments
    - Actor: Writer
    - Writer can create comment on a Post.
    - Goal: Can be done easily using Disqus 3rd party plugin, but it defeats the purpose of learning Laravel. Aim to explore how I can create a structure that provides threading feature.
  - Share Blog Post
    - Actor: Writer or Guest
    - Writer or Guest can share a blog post to 3rd party social platform.
    - Goal: Just a feature adding use case.
  - User Personal Profile
    - Actor: Writer
    - Writer can manage own personal profile.
      - Anyone can view it.
    - Goal: Another feature adding use case.
  - Like Blog Post
    - Actor: Writer
    - Writer can like a blog post.
    - Goal: To investigate how to create a better structure to keep track of 'likes'. Initial thoughts: Should it be a seperate table or a field in the existing Post table?

ER-Diagram


posts
PK
id
post_author
post_content
post_title
created_at
updated_at
comments
PK
id
post_id
comment_author
comment_content
created_at
updated_at
users
PK
id
name
username
email
password
remember_token
created_at
updated_at

Executing Plan

- Phase 1.1: Analysis & Planning (2 DAYS: 10 DEC 2015 ~ 11 DEC 2015)
  - Project Requirement Analysis & Planning
  - Setting Up Development Environment
  - Setting Up Github Repository
  - Database Schema Design
- Phase 1.2: Prototyping (5 DAYS: 11 DEC 2015 ~ 15 DEC 2015)
  - Database Schema Implementation (Laravel Migration)
  - Entity Model Creation
  - Membership Registration & Login
  - Blog Post CRUD
  - Database Seeding (Laravel Factory Model & Seeding Support)
  - Implement Authorization Guards
- Phase 1.3: Development into Mid-Fidelity (3 DAYS: 15 DEC 2015 ~ 17 DEC 2015)
  - Seperating layouts, reuse masterpage template
  - Basic styling (CSS, Bootstrap)
  - Testing
- Phase 1.4: Deployment (1 DAY: 18 DEC 2015)
  - Deployment to AWS
- Milestone: Mid-Submission @ 19 DEC 2015
- Phase 2.1: Explore other features and personal goals (15 DAYS: 20 DEC 2015 ~ 5 JAN 2016)
  - Comments, Like/Share/Search/AngularJS, other feature adding use cases.
- Milestone: Final-Submission @ 5 JAN 2016

Thoughts on assignment & difficulties faced

- Taking Wordpress as inspiration.
- Medium (https://medium.com/@tomwhitwell/52-things-i-learned-in-2015-c5c74eed24e0#.v3hf5h2ay)
- Route definittion takes precedences. One of the problem I faced was one of the resource pointing to the wrong Controller method, and I realize it was the order of route definitions. It's rather difficult to spot this with the Route::resource method with all the middleware complexity.
- Laravel Elixer with Gulp to manage aseets and task is new to me. In my previous projects, I pull in dependency using npm and refer the resource directly in node_modules folder. Since Laravel recommended to manage assets in the resources folder, it's necessary to manually copy the files over to different location after npm update. Gulp simplify this task. Moreover, I find it extremely convenient gulp is able to compile sass/less and combine all javascript/stylesheets into one single file, thus I only have to include one file in my header.
- PhpStorm has issue with Laravel syntax autocompletion. I've only get it partially working but it still prompts for warning for certain syntax. Nevertheless, the application is working fine.