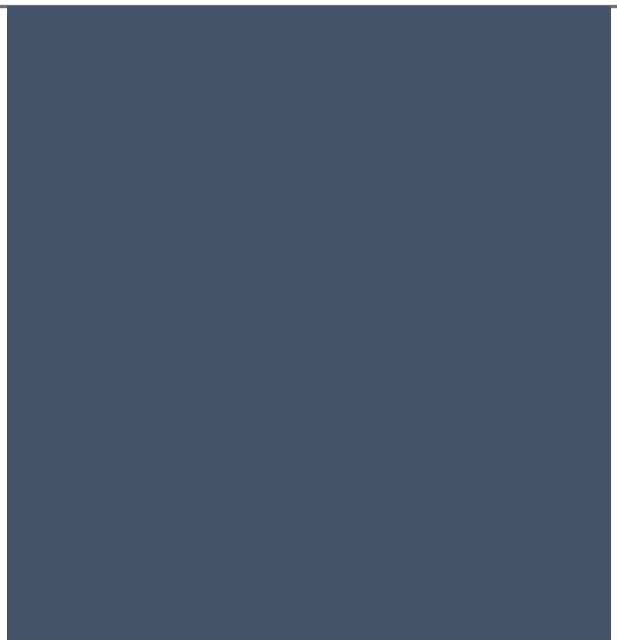

Helm’s Deep

Forrest Wong
T00603306

December 4th, 2020

Helm's Deep: A CodeCore Project - PLAR Credit Application

The following document details my project work at CodeCore and how other programmers should go about running the software I have written. The web application I have developed is called Helm's Deep.

Helm's Deep is a Reddit-like social media platform intended for free discussion amongst its members. Members of the community may post stories, forum contributions, photos, videos, and comments. The primary technologies used in the project include HTML, CSS, Bootstrap, JavaScript, Ruby, Ruby on Rails, and PostgreSQL.

While there is no single project that encompasses everything taught at CodeCore (an app could use Express or Ruby on Rails, but not both for instance), I believe applications such as Helm's Deep are highly representative. A major design choice facing many students is whether to build their web apps as monoliths or split the application between front-end and back-end frameworks (typically React and Ruby on Rails). I elected for the former, more monolithic approach.

Starting The Helm's Deep Application

1. The learning objectives implemented for the project include CRUD functionality, user authentication, user sessions, database relationships, styling, file uploading, media playback, pagination, and search engine implementation.
2. This application was developed on MacOS and these instructions will be specific to Unix systems. Do let me know if there is a compatibility concern regarding your target machine.
3. You will need to have PostgreSQL installed on your computer. The addition of pgAdmin may be also be helpful as a GUI interface for the administration of the database.
    a. https://www.pgadmin.org/
4. The username and password for the database connection in the project is specified in the file called "database.yml" which is located in the "config" folder. The username is "postgres" and the password is "password1".
    a. Your user configuration in pgAdmin will have to reflect this username and password combination in order for the Helm's Deep application to be able to access the database.
    b. Alternatively, you can change the username and password combination in the "database.yml" project file to match an existing database user configuration if you prefer.
2. Navigate into the "helmsdeep" application folder via terminal/shell
3. The application has several dependencies and may require some package installations. It is difficult for me to predict in entirety which dependencies may not be satisfied on your machine. You will likely require at least the following terminal commands:
    a. You will need to install the relevant Ruby version with "nvm install 2.7.1".

    b. You will need to run the “bundle” command in order resolve Ruby gem dependencies.
    c. Run the “yarn install” command to resolve yarn specific dependencies.
    d. Again, do confer with me if there are dependencies you are unable to resolve.
1. Next will be the creation and seeding of the database with the following commands:
    a. “rails db:create” will create the database in PostgreSQL.
    b. “rails db:migrate” will migrate the database tables, schema, etc.
    c. “rails db:seed” will seed the database with example data from the seed file.
2. You can then run the app with the command “rails s”.
3. Access the project in your web browser by visiting the “localhost: 3000” address.
4. You are now free to navigate the Helm’s Deep site. You can see that this social media platform allows for the creation of stories, forum posts, and new users. Users can upload media, comment, edit, delete, and search content.
5. The web app visual elements respond dynamically to varying display sizes.
6. Login to the super user account with the email “forrest@mail.com” and the password “password”. This gives you elevated privileges in the app which include access to an admin dashboard. The dashboard allows the app administrator to see all users, delete users, and remove their content on a site-wide basis.
7. A sample video and some photos have been included in a folder named “Media For Testing” within the .zip file. This content was included for your convenience to test out the media uploading and playback functionality.
8. Users can search the site’s content using the search bar at the top right of the interface. Search terms such as “transmit” or “capacitor” should yield positive search results relative to the seed data.

A Brief Discussion Of The Code In Helm’s Deep

Feel free to use a code editor of your preference to inspect my code. The application follows an MVC framework that is typical of Rails applications. Starting with the “app/config/routes.rb” file we can see that many resource paths have been created in the application. Generally, there will be a new set of resources (paths, controllers, etc.) created for every new database model generated.

Within the “app” folder navigate to the “controllers” subfolder to view a list of the controllers used in the application. The controllers are where the major logic of the application reside. Taking the “pictures_controller.rb” file as an example, we have logic to implement CRUD actions on instances of the picture model. Permissions are enforced using “before_action” methods that exist at the top of the file. Database integrity is enforced with validations at the model level. This prevents malicious users from potentially circumventing weaker forms of security such as form level validation.

Moving on to the “models” folder (another subfolder of the “app” folder) we have model files corresponding to each major type of content that exists on the site. Taking the “video.rb” file as

an example, we can see methods that allow the content to be searchable via the site's built-in search engine. Additionally, model validation prevents, say, videos from being uploaded without an associated user. A private method is present in order to set a default view count for newly uploaded videos.

Finally, the "views" subfolder within the "app" folder is where the site's front-facing HTML exists. These files are organized by their corresponding controllers. Each HTML file actually has the extension of ".html.erb" in order to allow for embedded ruby to give the webpages extra functionality. As an example, Ruby code is embedded in the "index.html.erb" file located in the "posts" subfolder for the purposes of pagination.

This concludes a brief tour of the code for this application. I do hope this document has struck an appropriate balance between providing detail and avoiding tedium. I am sure you will want to conduct your own investigation regarding code quality and implementation decisions. This README file is also accessible at the top right corner of the site's user interface as per your instructions. Feedback and suggestions are welcome. Thank you.