CRUDdit

A reddit clone

Alex franklin, ehsan motlagh, nick pelletier

Table of Contents

[Project Description 2](#_Toc147345891)

[Technologies Used: 2](#_Toc147345892)

[Special Features: 2](#_Toc147345893)

[Additional Libraries: 2](#_Toc147345894)

[Challenging Items: 2](#_Toc147345895)

[List of URLs 3](#_Toc147345896)

[Database Design 4](#_Toc147345897)

[Use Case Diagram 5](#_Toc147345898)

[Web Page Mock-ups 6](#_Toc147345899)

[Questions 8](#_Toc147345900)

# Project Description

Our project will be a clone of Reddit, targeting some of its core functionalities. The primary focus will be to implement user registration and log in, enabling users to create and comment on posts and other comments, as well as “vote” on posts and comments. Users will also be able to create their own “subcruddits,” where posts relating to the subcruddits’ topic will be shared.

There will be three tiers of user roles: users, moderators, and administrators. Anyone can visit the application to view posts and comments, but they must be logged in as a user to post and comment. Moderators are responsible for managing specific subcruddits where they have extra privileges (disabling commenting, removing posts, and stickying comments and posts). Administrators have full privileges over all subcruddits, and can delete subcruddits, users, as well as view all deleted data.

They layout of CRUDdit will be modelled off the “old reddit” look, providing a minimalistic design that is user friendly and functional, and that suites the project’s scope. Most importantly, Reddit is a discussion forum. The most important functionality and the core of our application is creating and communities (subcruddits) as well as the ability to participate in long comment-based discussions under posts.

# Technologies Used:

* React
* NodeJS
* MySQL
* Express
* Git/GitHub, Trello
* AWS S3, AWS ECS, (AWS RDS?), Docker?

# Special Features:

* Sign in with google

# Additional Libraries:

* Sequelize
* Yup
* Formik
* Canvas
* Bootstrap/Tailwind
* Socket.io
* Multer

# Challenging Items:

* Storing user submitted images in our own cloud storage, creating relative path for database.
* Logging in and registering with Google
* Notifications

# List of URLs

* /
* /login
* /register
* /c/subcruddit/, /c/subcruddit/new, /c/subcruddit/top
* /c/subcruddit/submit
* /c/subcruddits/create
* /c/subcruddits/{new/top}
* /user/username, user/username/comments, user/username/submitted
* /admin/users, admin/user/id, admin/subcruddit, admin/subcruddit/post
* /admin/c/subcruddit/post
* /api/users/
* /api/users/login
* /api/users/:id
* /api/users/:username
* /api/subcruddits/
* /api/subcruddits/:id
* /api/posts/{new/top}
* /api/posts/:id
* /api/posts/votes
* /api/comments/{new/top}
* /api/comments/:id
* /api/comments/votes
* /api/moderators/
* /api/moderators/:id

# Database Design

**A diagram of a computer code

Description automatically generated with medium confidence**

Figure : Database Diagram

# Use Case Diagram

A diagram of a diagram

Description automatically generated

Figure Use Case Diagram

# Web Page Mock-ups

A screenshot of a computer

Description automatically generated

Figure Screenshot Reddit Homepage - Logged In User

A screenshot of a computer

Description automatically generated

Figure Homepage & Subreddit Wireframe

**A screenshot of a web page

Description automatically generated**

Figure Post with Comments Wireframe

# Questions

* What is the best way to organise the database in order to add notifications?
* How would we go about implementing a sorting algorithm for “Hot” taking into account new posts and vote count.
* What is the best way to organize the comments, given our database structure?
* If we create an AWS EC2 instance with Docker, would it be smart to maintain an S3 bucket independently?