Milestone 3 for **CodeFeed**

Alec Trievel, Tim Platts, Jeremy Kato,

Joseph Kostial, & Albert Yang

**Table of Contents**

1. Preliminary User’s Manual
   1. Product Overview
   2. Getting started
      1. Login
      2. Help Mode
      3. Sample Runs
   3. Modes of Operation
   4. Advanced Features
   5. Glossary of Commands and System Outputs
2. Requirements Spec
   1. Product overview and summary
   2. Information Description
      1. User interface (A Preliminary User manual)
      2. High Level Data Flow Diagram
      3. Data Structure Representation
      4. Data Elements Dictionary
   3. Functional Description
      1. Functions Represented by IC cards
      2. Processing Narrative
      3. Design Constraints
      4. Diagrams
   4. Performance Requirements
   5. Exception Conditions / Exception Handling
   6. Implementation Priorities
   7. Foreseeable Modifications and Enhancements
   8. Sources of Information
   9. Acceptance Criteria
   10. Revision History
3. SPG Tool Input and Output

# Preliminary User’s manual

## Product Overview

CodeFeed is a social media interface designed for software engineers, by software engineers. This web application will implement certain functionalities from already popular social-media platforms, including: Twitter, Reddit and Stack Overflow. However, CodeFeed will combine and modify the best features of these platforms to create a unique and refreshing user experience. This platform will be sure to attract developers who wish to create quality and inspirational codebases.

## Getting Started

### Log in

Users will be presented with a login page, with input fields for their username and password once they enter the site. If the user enters a correct username/password combination, they will be logged in and brought to a home page. If the user does not enter correct login info, they will stay on the login page and be presented with warning text notifying them that the information they entered was not recognized. There will also be an option on the login page to create a new account, if the user does not already have one.

### Help mode

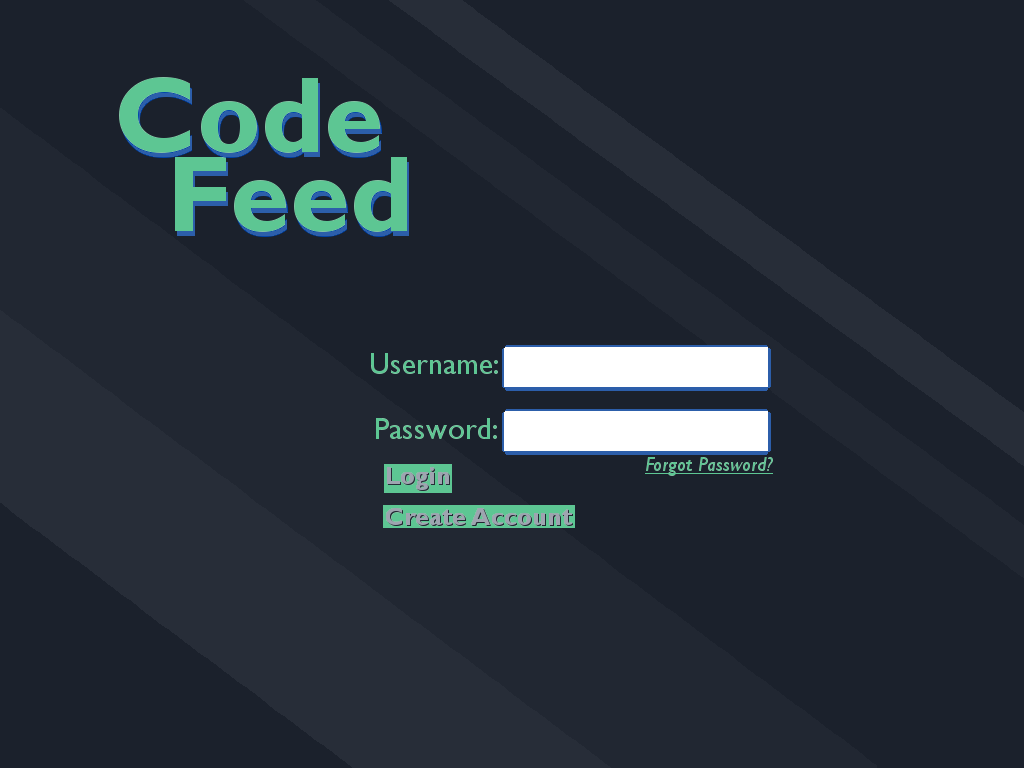
If a user needs help navigating the site or using the application’s many features, there will be an option on the menu bar which will allow them to enable help mode. Help mode will have two options: a search feature (tentative) and a tour feature. The search will allow the user to look for specific keywords, which will be linked to our documentation. After all, since this application is designed for developers, they should have no problem reading some form of documentation. If the user chooses the tour mode, a simple animated tour will pop up. The tour will explain certain features of the application with enough detail to get them started, but not as much as a set of documentation would. The tour will be a small model with an overlay background that also highlights the current area of the page being explained. The user will have the option to end the tour, go to the next “slide,” or go to the previous “slide,” if applicable.

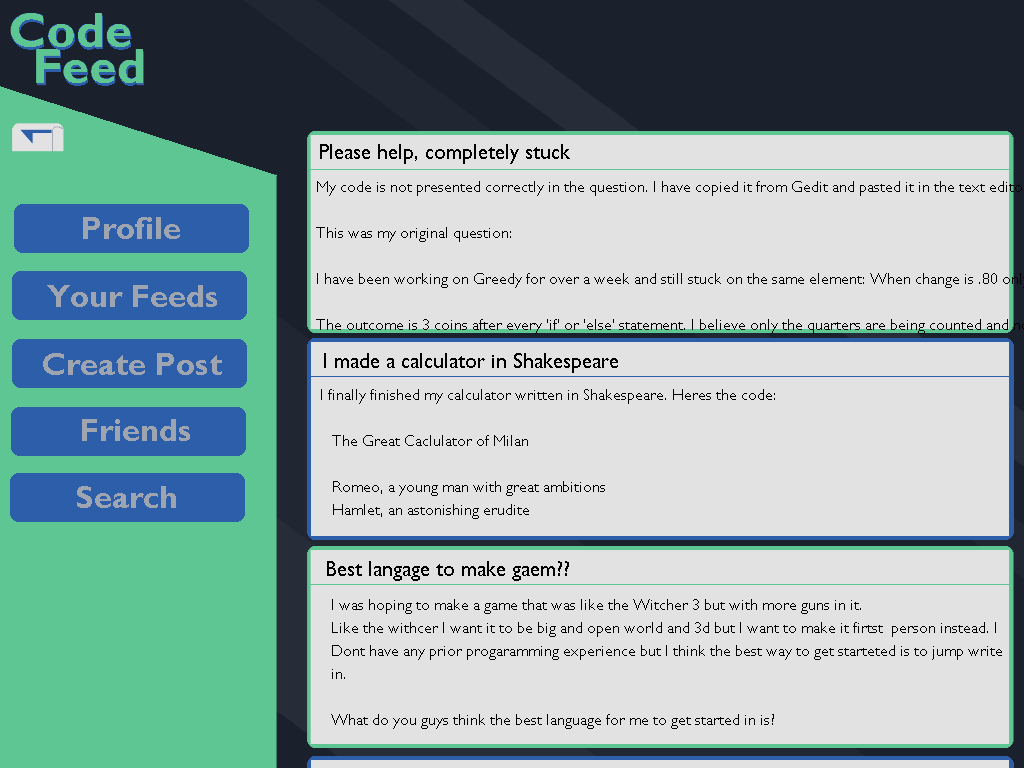
### Sample runs

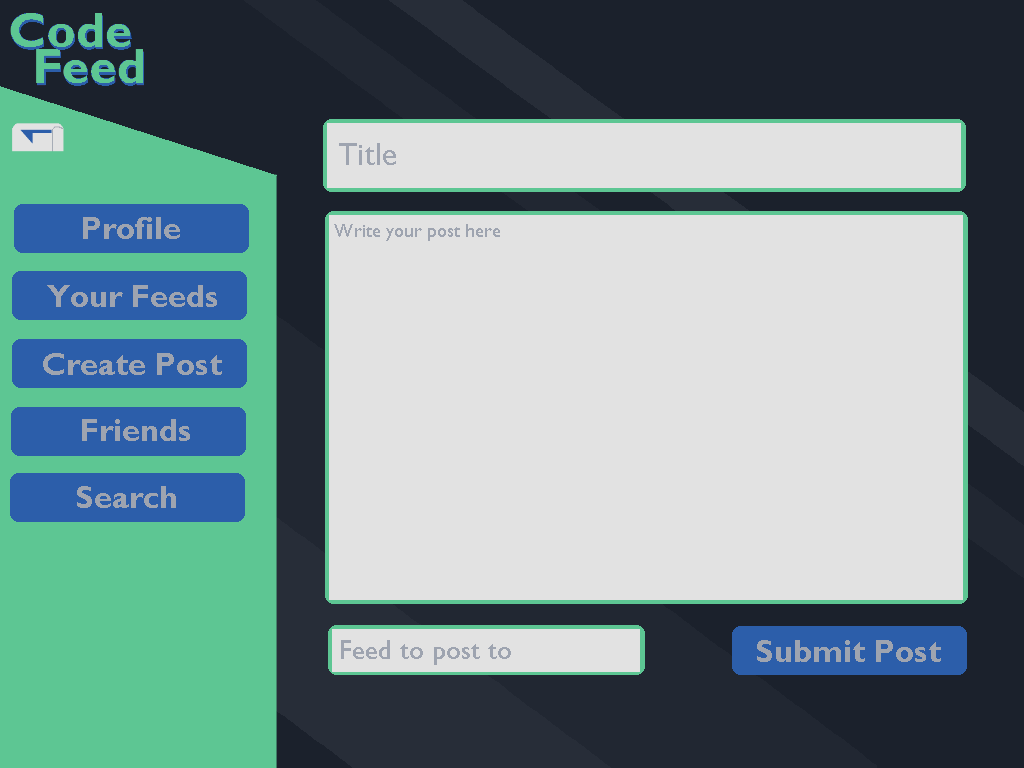
**Create a Post**

To create a new post, the user must follow the following steps. First type in your username and password and login to CodeFeed. Next click the “Create Post” button the sidebar to the left. Type in the title for your post, the main content, and the feed to post to. After you are done, click the “Submit Post” button at the bottom right. See the steps below for a more detailed description.

1. At the login screen pictured below, click on the textbox titled “Username” and type in your username. Then click the textbox titled “Password” and type your

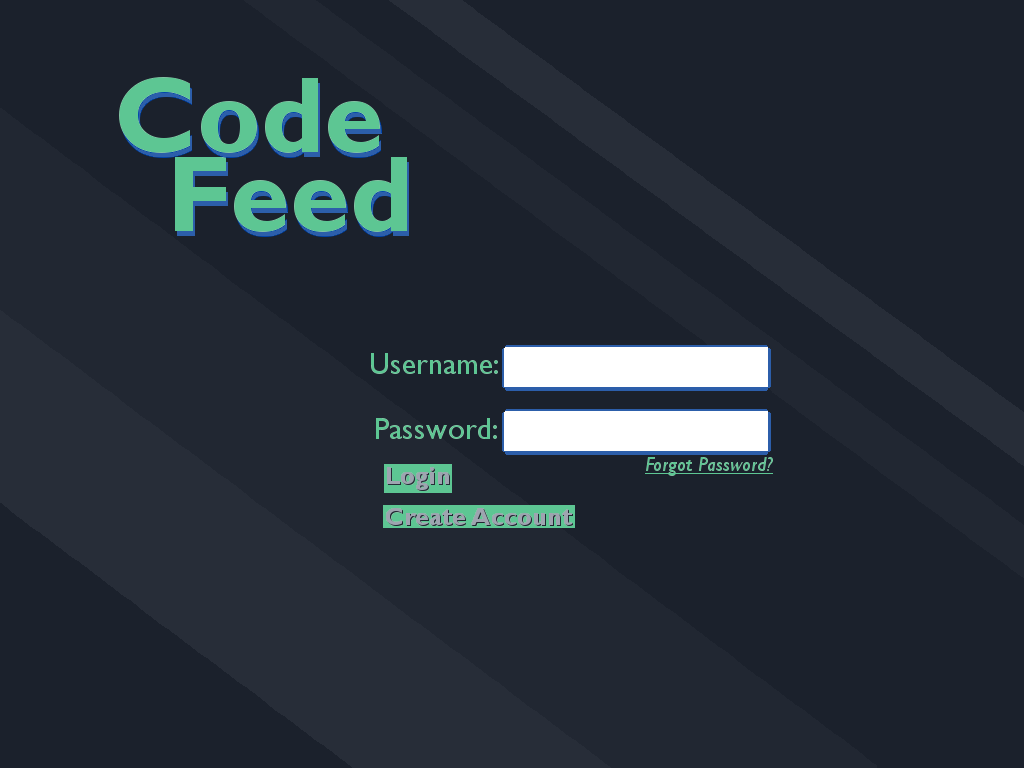
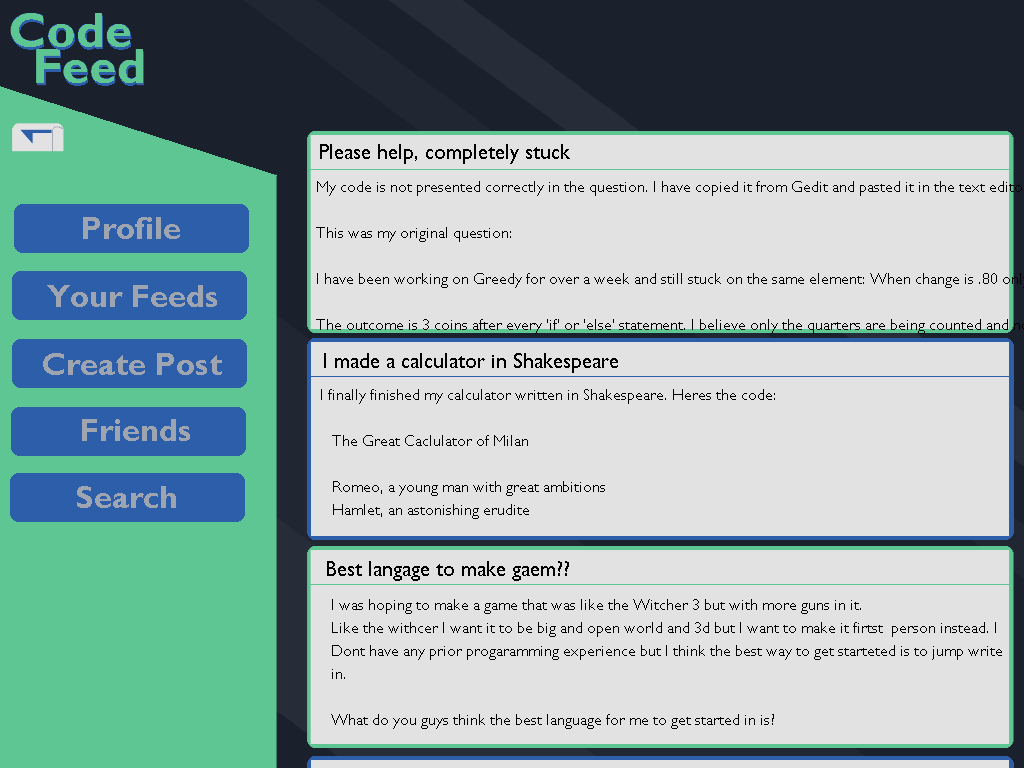
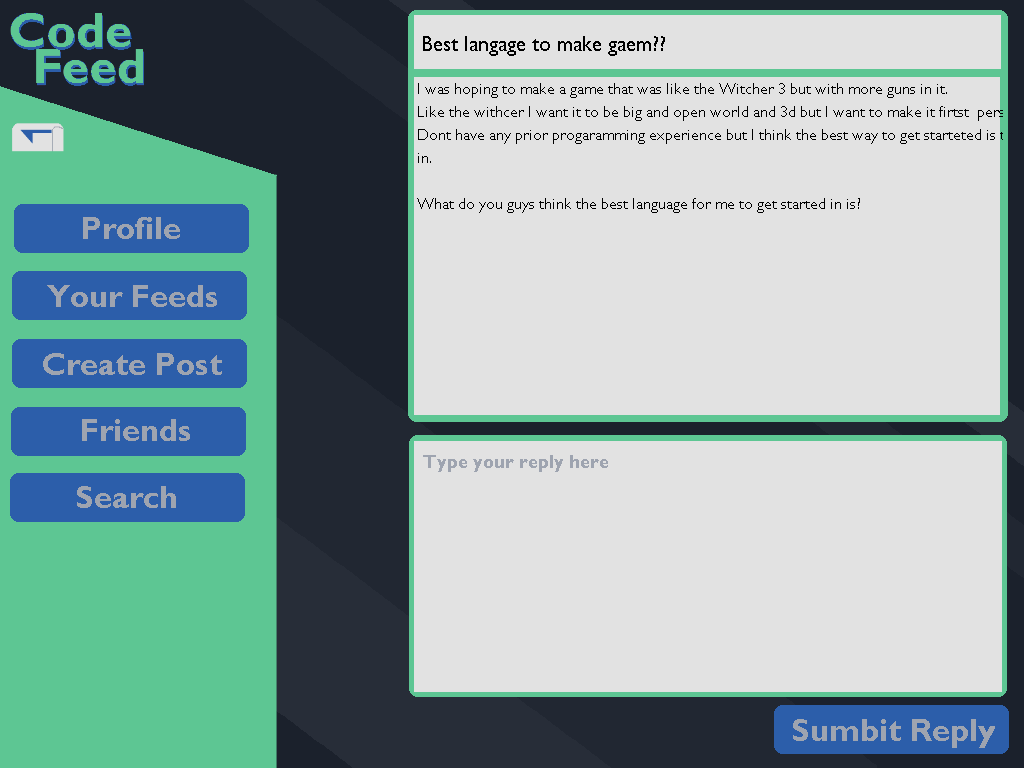
password. Then login by clicking the button titled “Login”.

1. You will be taken to your home screen as shown below. From here click on the “Create Post” button on the sidebar to the left.
2. This will bring you to the post creation page. First click on the top text box labeled “Title” and type in a title for your post. Then click on the large text box in the middle labeled “Write your post here” and type in the main content for your post. Next click on the small text box at the bottom labeled “Feed to post to” and enter the name of the feed you want to post to. When your happy with your post, click on the “Submit Post” button at the bottom right.



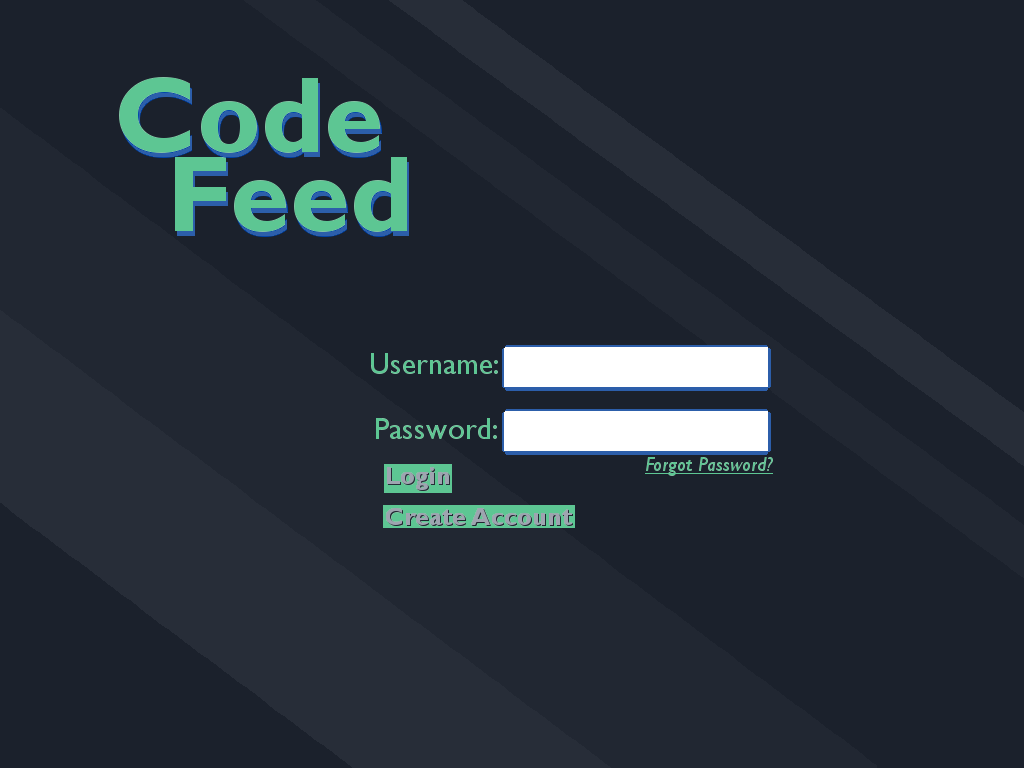
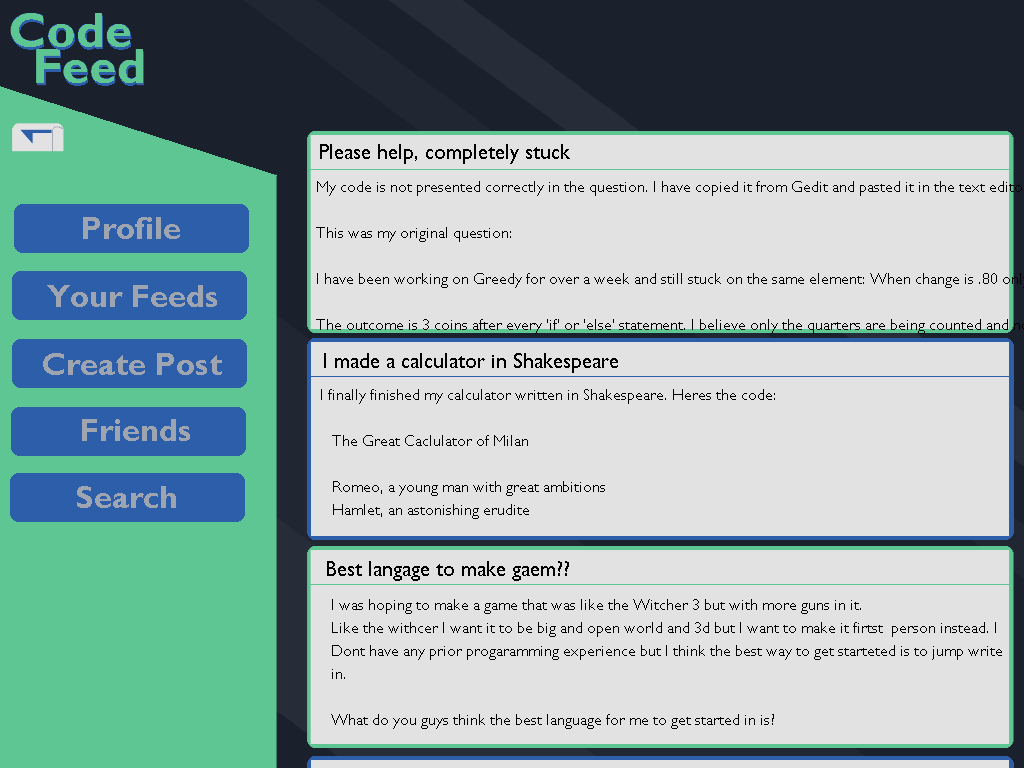
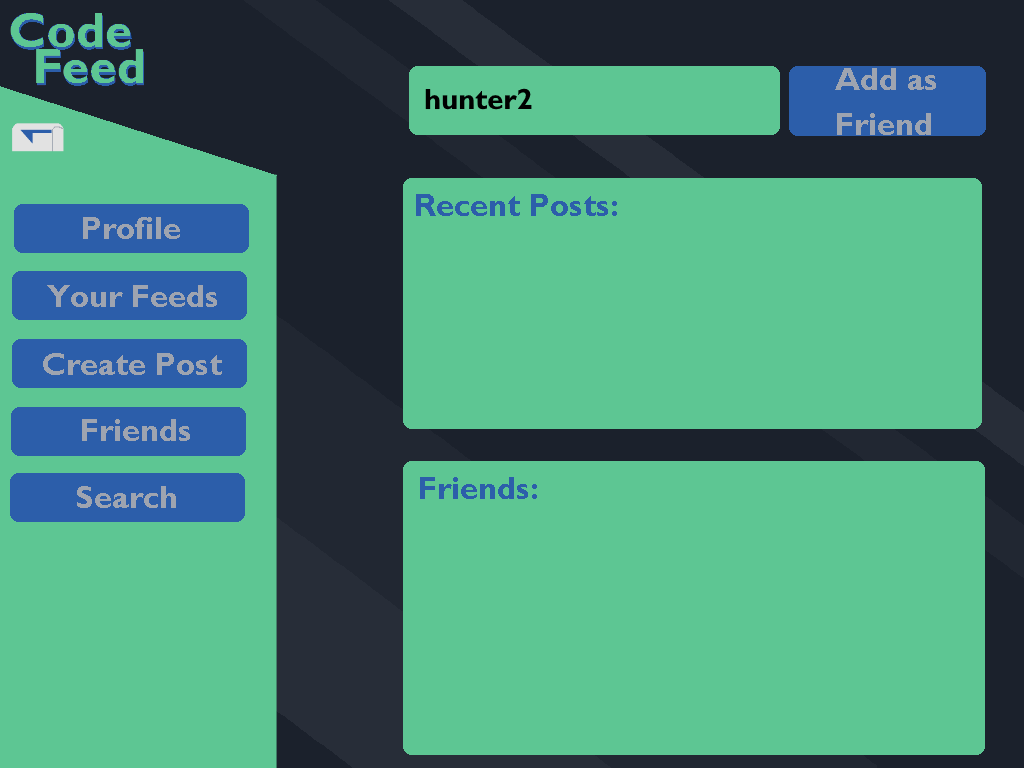
**Reply to a Post**

To reply to a post, the user must follow the following steps. First type in your username and password and login to CodeFeed. Next select and click on the post you want to reply to in the list on the right. Click on the button labeled “Reply”. From here click on the bottom text box titled “Write your reply here” and type in your reply. Finally click on the button labeled “Submit Reply”. See the steps below for more detailed instructions.

1. At the login screen pictured below, click on the textbox titled “Username” and type in your username. Then click the textbox titled “Password” and type your password. Then login by clicking the button titled “Login”.
2. You will now be at your main viewing page. To the right is a list of posts from the feeds you follow. Find a post you want to reply to and click on it.
3. From here click on the button labeled “Reply” below the post you selected.
4. You should now see the screen pictured below. Click on the textbox at the bottom of the screen labeled “Write your reply here” and type in your reply. When you are finished click on the button labeled “Submit Post” at the bottom 

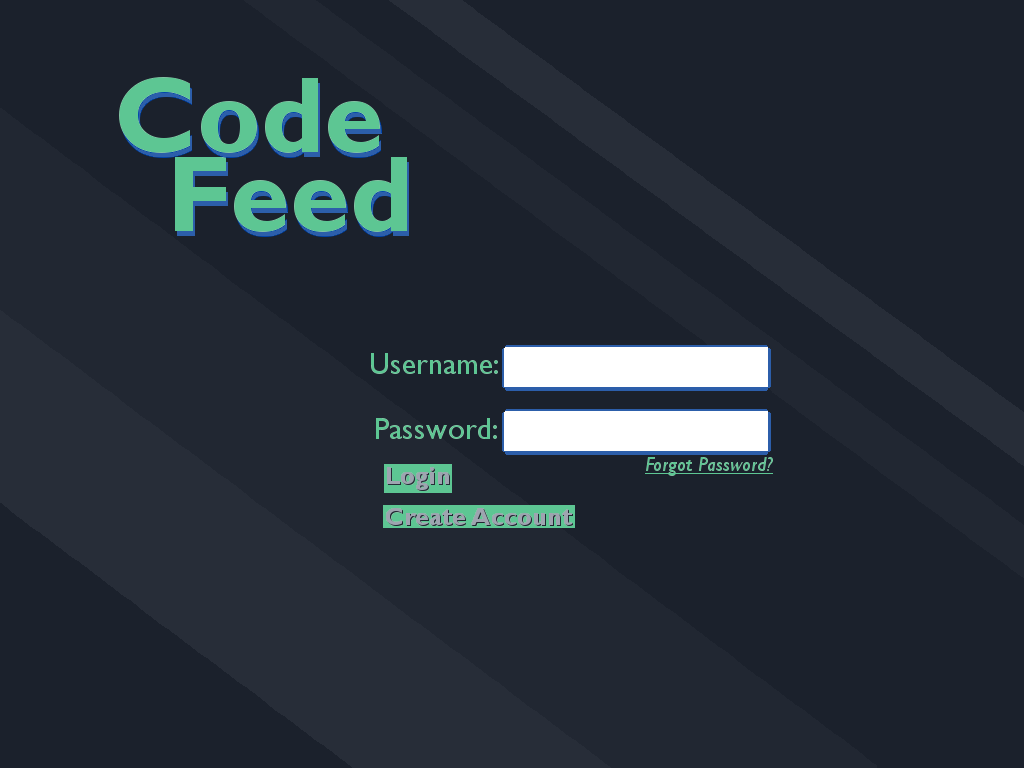
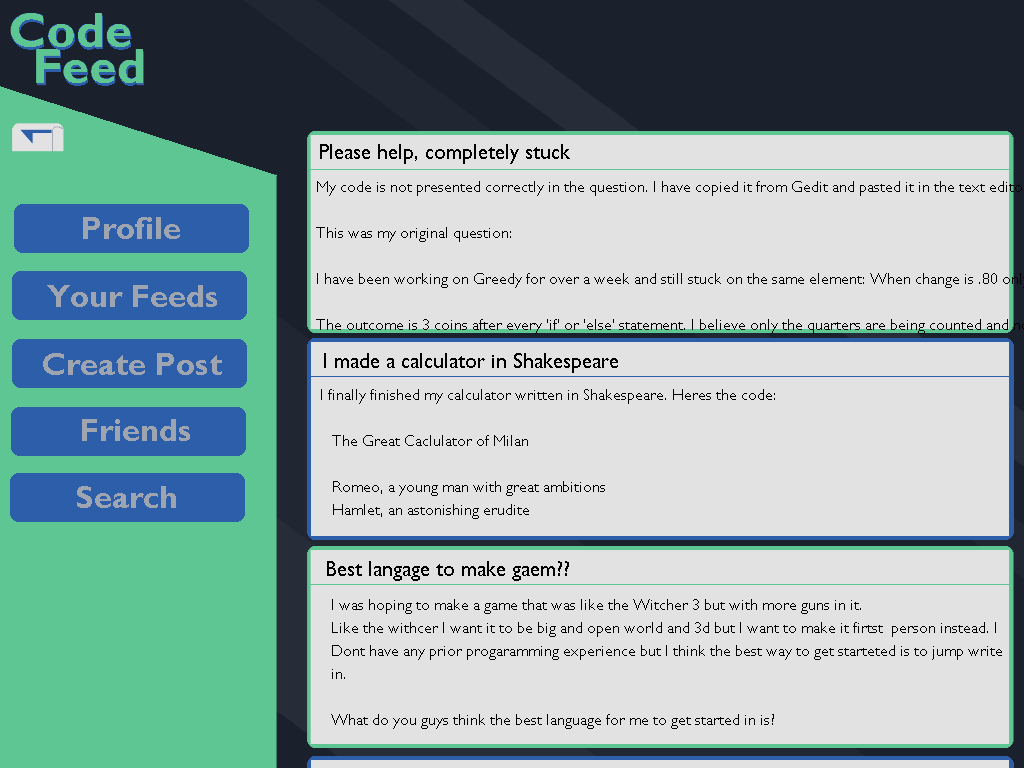
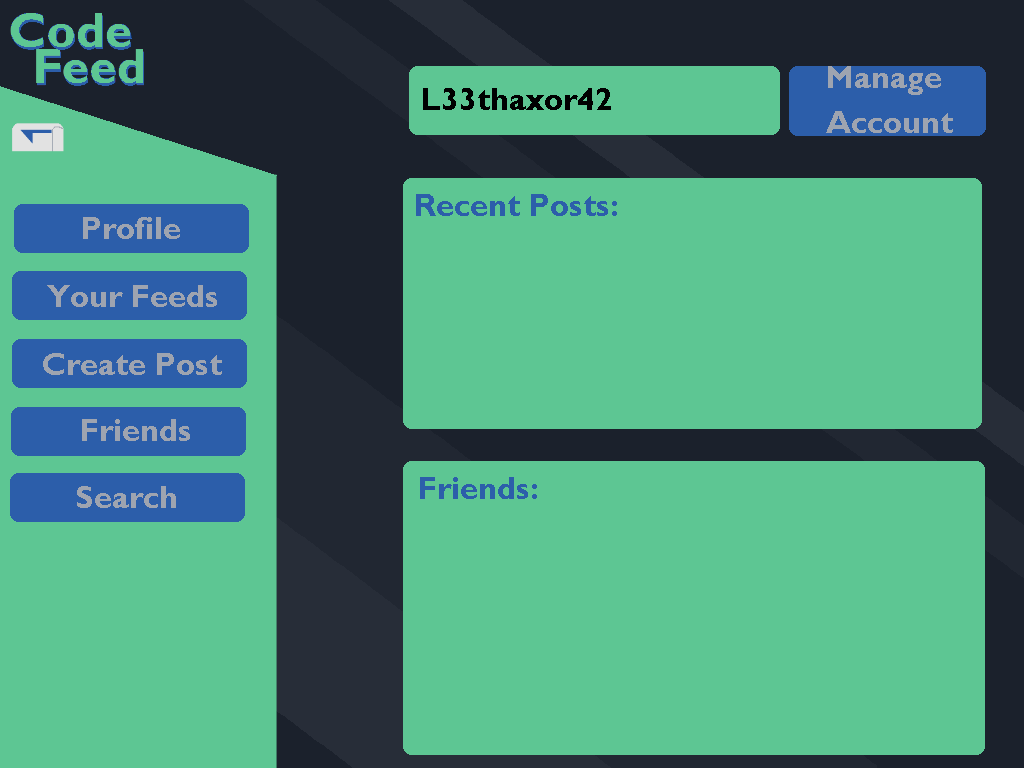
**Add a User As a Friend**

To add another user as a friend, a user must follow the following steps. First type in your username and password and login to CodeFeed. Next you must navigate to the profile page of the user you wish to add as a friend. One way to do this is as follows. Find a post the user has created or replied to in your feed to the right of the screen. Click on this post. From here, click on the name of the user you wish to add as a friend to be taken to their profile page. Lastly, click on the button labeled “Add as Friend”. See the steps below for more detailed instructions.

1. At the login screen pictured below, click on the textbox titled “Username” and type in your username. Then click the textbox titled “Password” and type your password. Then login by clicking the button titled “Login”.
2. You will now be at your main viewing page. To the right is a list of posts from the feeds you follow. Click on one the posts.
3. You should now see the post and all the replies to it. The replies are in a list at the bottom of the screen. Every reply begins with the username of the person the wrote the reply in bold font followed by their reply. Click on one of the usernames to be taken to their profile page.
4. You should now see the profile page as shown below. At the top right of the page is a button titled “Add as Friend”. Click on this button to add the user as your friend.

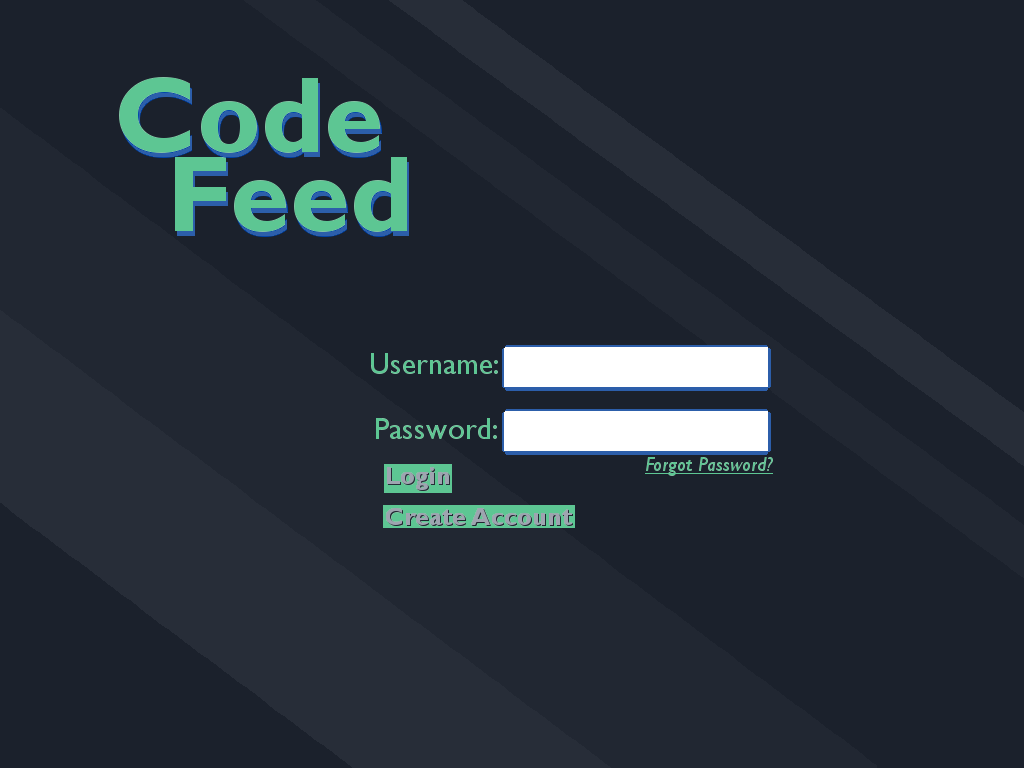
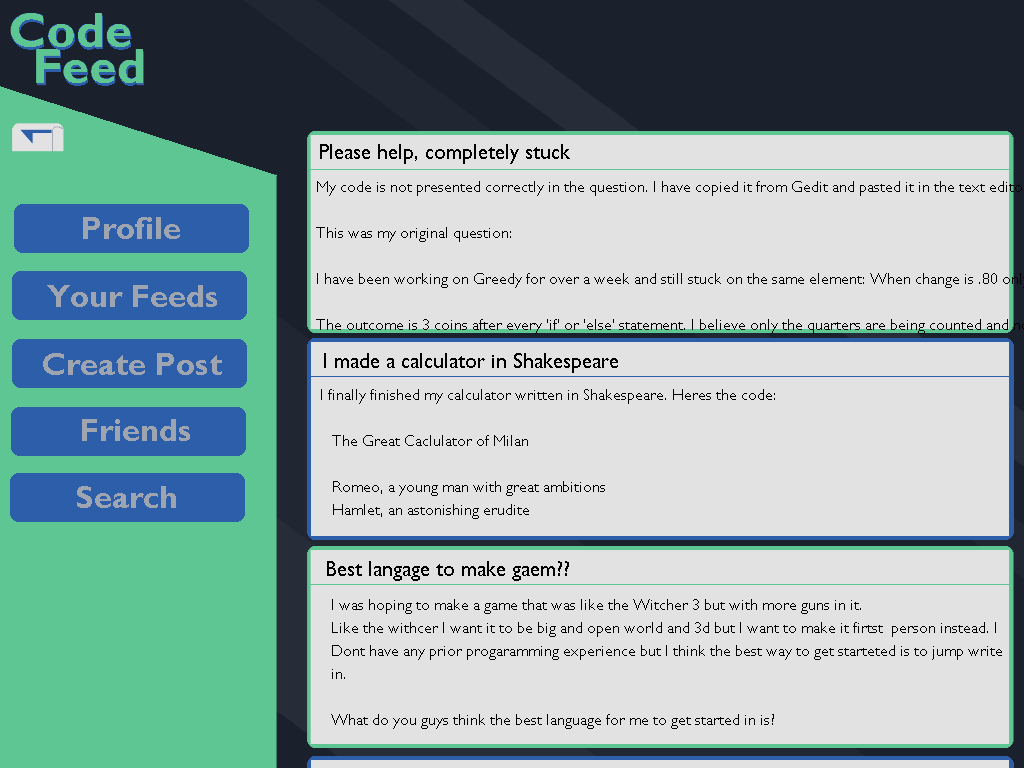
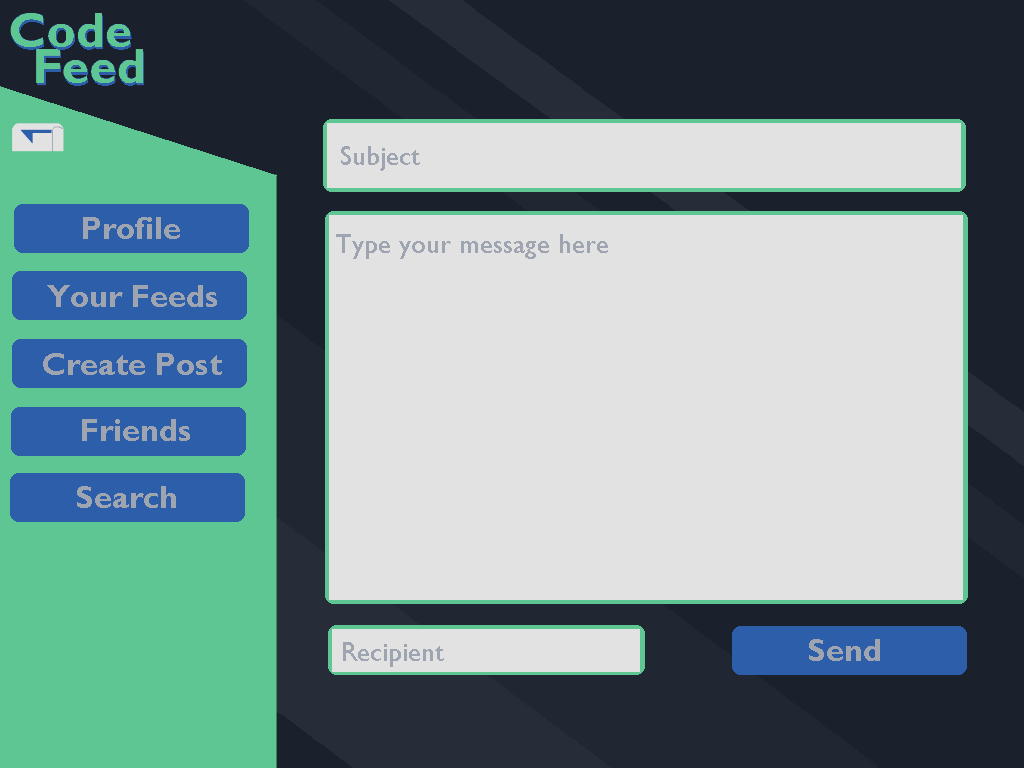
**Change Password**

To change your password a user must follow the following steps. First type in your username and password and login to CodeFeed. Next click on the button labeled “Profile” on the sidebar to the left. Then click on the button labeled “Manage Account” on the top right of the screen. Enter your current and new password into the textboxes provided. Finally click on the button labeled “Change Password” to change your password. See the steps below for more detailed instructions.

1. At the login screen pictured below, click on the textbox titled “Username” and type in your username. Then click the textbox titled “Password” and type your password. Then login by clicking the button titled “Login”.
2. You will now be at your main viewing page as shown below. Click on the button labeled “Profile” on the sidebar to the left of the screen.
3. You will now be at your profile screen as shown below. Click on the button labeled “Manage Account” at the top right of the screen.
4. You will now see the screen showed below. Click on the textbox titled “Current Password” and type your current password. Next click on the textbox titled “New Password” and type what you would like to change your password to. Finally click on the button labeled “Change Password” to finish changing your password.

**Send a Private Message**

To add another user as a friend, a user must follow the following steps. First type in your username and password and login to CodeFeed. Next you must click on the button shaped like a mailbox at the top of the sidebar to the left. Then click on the button labeled “New Message” at the top right of the screen. Type in the subject, message, and recipient in the text boxes. Finally click on the button labeled “Send” at the bottom right of the screen. See the steps below for more detailed instructions.

1. At the login screen pictured below, click on the textbox titled “Username” and type in your username. Then click the textbox titled “Password” and type your password. Then login by clicking the button titled “Login”.
2. You will now be at your main viewing page as shown below. Click on the button shaped like a mailbox at the top left of the sidebar on the left side of the screen.
3. You will be taken to your inbox as shown below. To send a new message click the button labeled “New Message” at the top right of the screen.
4. You will now see the screen shown below. Click on the text box labeled “Subject” and type the subject of your message. Next click on the text box labeled “Type your message here” and type your message. Next click on the text box labeled “Recipient” and type the username of the person you want to send the message to. Finally click on the button labeled “Send” at the bottom right of the screen to send the message.

## Modes of Operation

As seen in the sample runs above, the main modes of operation are represented on the above storyboard and various pages of the site.

* Login Screen
  + Users are first greeted by the login screen
  + Users can login
  + Users can navigate to another screen to allow them to create an account for CodeFeed.
  + Users can reset their password, allowing them to submit their email or username and have a password reset link emailed to them.
* Navigation
  + Once having logged in, users will be brought to the dashboard page, showing them a couple of feeds of posts based on the user’s preferences. Additionally, the sidebar, the main means of navigation, will be visible here and on each page of the site. The sidebar leads to the following options:
    - Create Post
    - Feeds
    - Profile
    - Friends
    - Search
* Post Creation
  + Users can create posts anytime by clicking the button on the sidebar
  + The post screen prompts the user for a post title, body, and a category to place it in.
  + Once complete, the user can press submit and it will be added to the appropriate feeds that it belongs to.
* Profile
  + Users can edit their profiles to their liking and see their past posts. If they click on other users’ names, they’ll be brought to a similar screen without the edit options.
  + Biography - Users can create a short bio describing themselves and their interests.
  + Settings - If users want to edit any of their preferences or change their password, they can do so here.
  + Past posts - Users can view the previous posts and comments that a person has made.
  + Friending - If viewing someone else’s profile, users have the option to send friend requests to follow their activity
* Friends
  + Users can add other users as friends and follow their posts.
  + From this view, you can see all posts made by friends
  + Posts can either be sorted by creation date, by upvotes, or by activity (meaning either recently posted or recently commented on)
* Feeds
  + Users can navigate various categories of posts based on what they’ve indicated they’re interested in following
  + My Feeds - The default feed when you navigate here. Shows you posts from all categories you follow.
  + All - Shows posts from all feeds
  + Specific Categories - Users can also select specific categories to browse instead.
  + Posts can either be sorted by creation date, by upvotes, or by activity (meaning either recently posted or recently commented on)
* Search
  + Users can search for posts, comments, categories, or other users here.

## Advanced Features

This list of advanced features is a compilation of features not originally designed for CodeFeed, but would be a nice addition in future iterations of the project.

**Moderators**

This feature would add the functionality for users to be designated as moderators for different sub-feeds. Users would be able to create sub-feeds and moderate them, giving them the ability to monitor what is posted, ban other users, and set other restrictions for their sub-feed. This is inspired by Reddit’s subreddit system, and while it is not something that is a definite goal, it would be an interesting feature to explore once we finish our main project.

**Following other users**

As it is now, users will be able to follow sub-feeds, which will contain posts from various users. This feature would allow users to follow other individual users, so that they would be able to see when that specific user made a contribution. This way, if a user was more interested in the content provided by a specific user, rather than a sub-feed, they would be able to more easily view that content without manually going to said user’s profile.

**More messaging options**

Currently, we plan on allowing text communication between users. However, if we have time at the end of the project, it might be good to allow other sorts of messages (e.g. images, gifs, videos, special emojis). This would give users more ways to communicate with one another, and a richer messaging function would certainly improve the user experience.

## Glossary of Commands and System Output

This glossary is primarily comprised of the various things that a user can either interact with or do on CodeFeed’s website.

* **Browse:** Users can browse various posts on CodeFeed by different properties. These include by creation date, most upvotes, the category, or the user that posted it.
* **Category:** Posts on CodeFeed can be supplemented with categories that help describe what the post is about. For instance, a question about the Java programming language would be placed in the Java category. Users can then browse categories to see all posts in a given category.
* **Create Account:** A user can create an account on CodeFeed by providing their email, a display name, and a password to access their account. Their account will then be activated by sending an activation email to the user. The email must be unique - users cannot share an email account.
* **Comment:** Users can comment in response to posts or other comments. Comments consist of a text body and a post or a comment to reply to. Comments can be voted on.
* **Follow:** Users can follow other users to be able to see their posts first while browsing. Users can do this by viewing another user’s profile and clicking a button to follow them.
* **Login:** The home page of CodeFeed is where the user can login, providing their username and password, where they will then be authenticated or denied access.
* **Post:** Posts are the main elements of CodeFeed. Users can create posts by providing a title, a body of text, and categories that the post belongs to. Posts can be commented on and voted on. Posts can also be browsed by their properties - the user who posted it, the total votes, how recently it was created, and its categories.
* **Upvote / Downvote:** Users can express whether they find a post or comment useful, helpful, interesting by voting on it. Posts and comments that are upvoted receive one point and will be given priority over other posts and comments when being sorted by total votes. Posts and comments that users deem to be of poor quality are pushed to the bottom and given lowest priority when being displayed. Additionally, CodeFeed tracks the total balance of votes that a user has received for each of their posts and comments. Lastly, a user can only vote on a given post or comment once - if they wish to change their vote or remove it, it overwrites their previous vote.
* **User Profile:** The user profile is where users can see their own (as well as others) information about themselves. Here they see their display name, their biography, and their recent posts and comments.

# Requirement Spec

## Product Overview and Summary

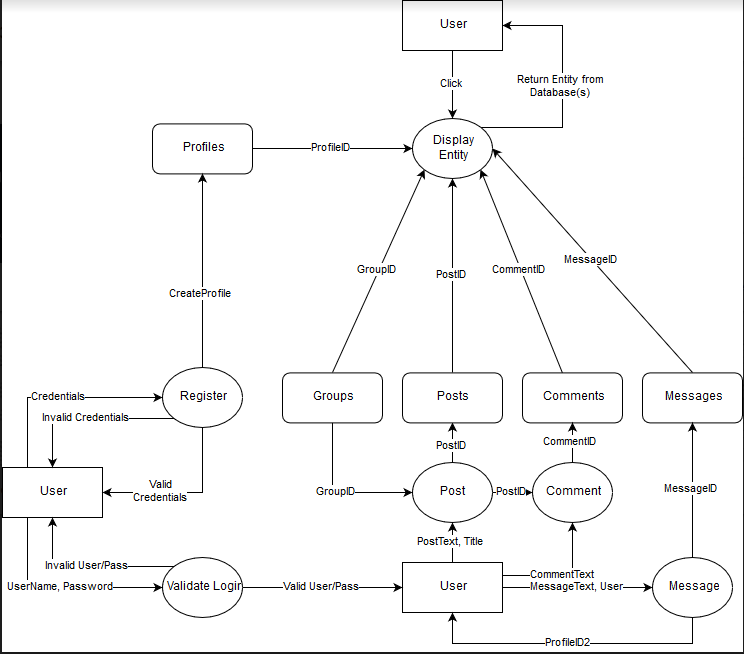
CodeFeed is a social media interface designed for software engineers, by software engineers. This web application will implement certain functionalities from already popular social-media platforms, including: Twitter, Reddit and Stack Overflow. However, CodeFeed will combine and modify the best features of these platforms to create a unique and refreshing user experience. This platform will be sure to attract developers who wish to create quality and inspirational codebases. The application will run on the most stable and newest versions of Python, Flask, SQLAlchemy, Google Chrome, JavaScript, HTML5, CSS3, and PostgreSQL. This will ensure the application is modern, scalable, and professional across all aspects of the stack, while also being easy to test and maintain. To test the application, we will use a combination of unit, integration, and mutation testing.

## Information Description

### User interface (A Preliminary User Manual)

The preliminary user manual can be found in section 1 of this document.

### High level data flow diagram



### Data Structure Representation

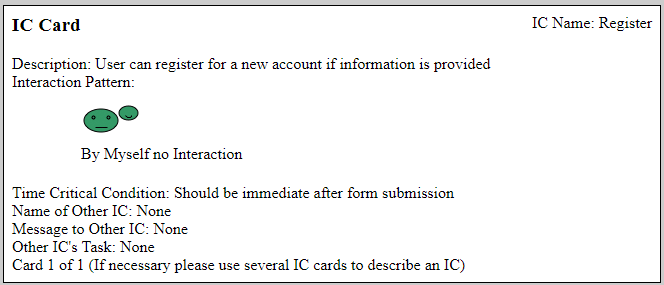
* User (userID, username, name, email, biography, creationDate, isValidated, lastLogin)
* Category (categoryID, name, description)
* Thread (threadID, categoryID, userID, title, body, creationDate)
* Comment (commentID, threadID, body, creationDate)
* Friendship (friendshipID, user1ID, user2ID, creationDate)
* ThreadVote (voteID, catergoryID, userID, type)
* CommentVote (voteID, commentID, userID, type)
* Message (messageID, user1ID, user2ID, body, creationDate)

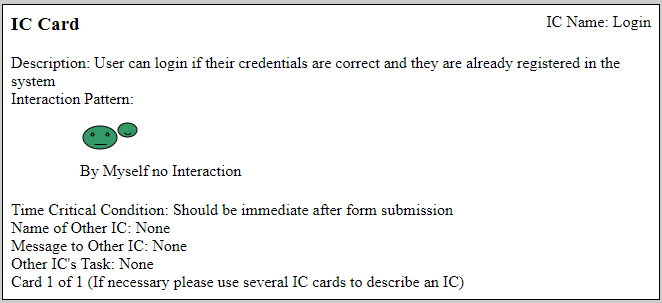
### Data Elements Dictionary

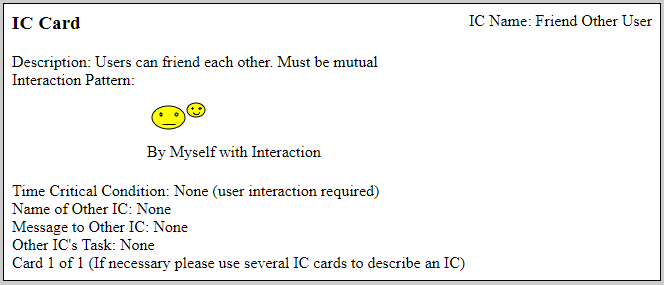
* userID: User, Thread, Comment, Friendship, ThreadVote, CommentVote, Message
* username: User
* name: User, Category
* email: User
* biography: User
* creationDate: User, Thread, Comment, Friendship, Message
* isValidated: User
* lastLogin: User
* categoryID: Category, Thread
* description: Category
* threadID: Thread, ThreadVote
* commentID: Comment, CommentVote
* title: Thread, Comment
* body: Thread, Comment, Message
* commentID: Comment
* friendshipID: Friendship
* voteID: ThreadVote, CommentVote
* type: ThreadVote, CommentVote

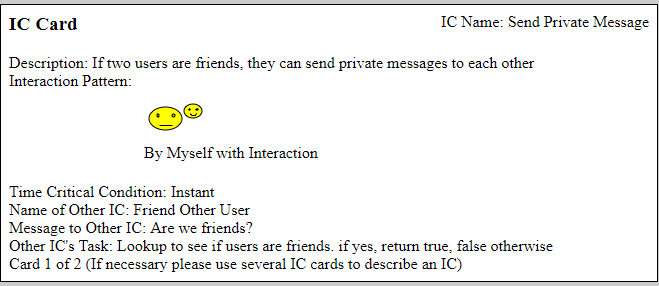
## Functional Description

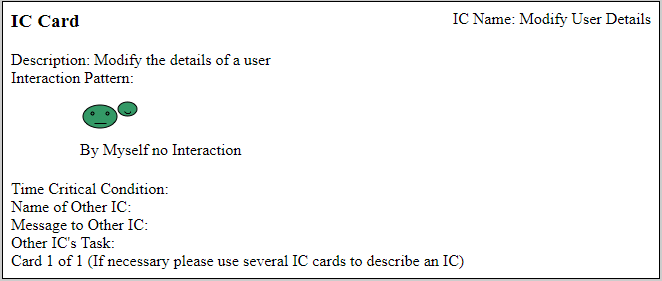
### Functions

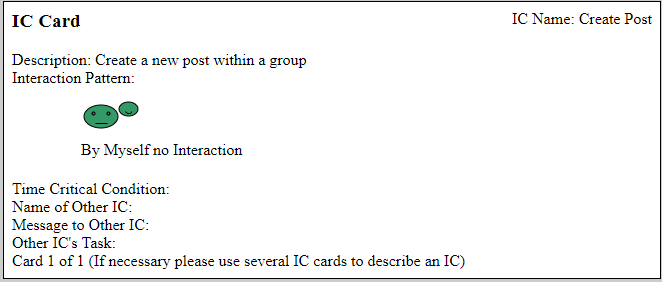


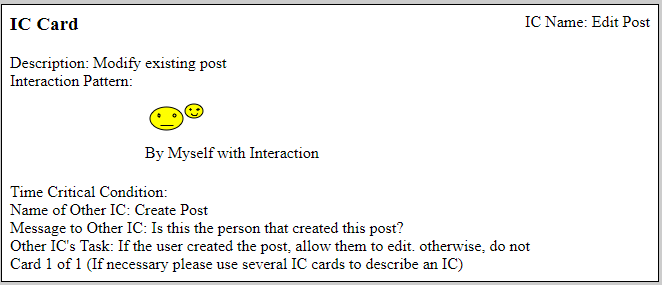


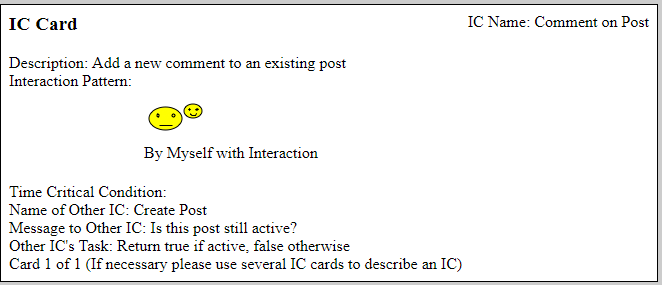


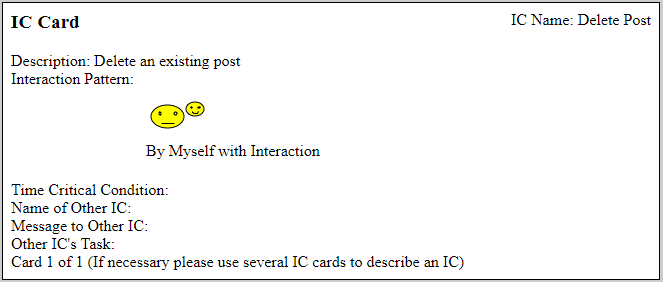












### Processing narrative

**Register**

The user navigates to the register account page, and enters the requested information. This will include: name, date of birth, email, username, and password. Once the user hits submit, the information on the page will be sent back to the server and validated. If the data is valid, the information will be saved in the database, and the user’s account will have successfully been created.

**Login**

The user navigates to the login page, and enters their username and password. Upon form submission, the data is validated, and an assertion occurs to make sure the information reflects what is in the database. If not, the user will see a message telling them their username or password is incorrect, or they need to register an account. Otherwise, they will be redirected to their personal homepage.

**Friend Other User**

If the user wishes to send private message to another user, they must be friends first. This can be accomplished by first searching for the user, then clicking an “add friend” button. This lets the other user have the option of accepting or rejecting the friend request. If the user accepts, then they will be able to send private messages. Otherwise, nothing has changed.

**Send Private Message**

This card required the actions from “Friend Other User” to be true. This means the users must be friends. One user can send a private message to a friend by going to their friend tab, clicking the other user, and selecting the “send message” button. This will bring up a pop up allowing the user to type in a message and hitting “send.” Once the message is sent the other user, the other user will have the option of reading the message, deleting it, or ignoring it.

**Modify User Details**

After logging in, the user can choose to modify their profile on the navigation bar on the left. This will be another form-like page where the user can change any of the fields on the page. This will include their username, password, and email (for now). Once the user submits the form, it will be validated on the server, and the database will be updated to reflect the changes.

**Create Post**

Once a user is registered and logged in, they have the option of creating a new post. The post will consist of a title and body. The title is a short description of the overall post, and the body is a lengthier description of what the overall post is about. As long as both of these fields are filled out, the form will be submitted and the new post will be saved into the database.

**Edit Post**

As long as the user created a post, they will have access to modify the body of the post only. The title will remain unchanged to ease searching functionality. The user can navigate to their created post and click the “edit” button. They can then modify the body of the post and click submit. Once the form is submitted, the changes will be reflected in the database and on the page.

**Comment on Post**

Any user who is logged into the system can comment on any post, including their own posts. The user will navigate to the post’s page, then click on the “comment” button. The user can then type their comment into the popup, then click “submit.” This will send the comment’s text to the server, which will then be stored in the database. The page will be refreshed at this point to allow the new comment to be generated on the page.

**Delete Post**

As long as the user created the post, they will have the ability to delete the post entirely. This action is permanent and cannot be reversed. This will also delete all comments associated with the post as well. The user can navigate to the post the created, and then click the “delete post” button. They will then be asked if they truly wish to delete the post because the action cannot be undone. If they click “Yes,” the request will be sent back to the server. The server will then be responsible for deleting the post and all associated comments with the post. A message will be returned to the user that the post has been deleted, and all other users who try to navigate to this page will receive a graceful 404 error.

### Design constraints

**Register**

Only one account can be registered per email. The person must be at least 18 years old to register. Besides checking the formation of the email (using a regex pattern to determine if it follows a “x@some\_domain” style), there will be no additional checks to see if the email is valid. Otherwise, anyone with valid information can register for the application.

**Login**

The user must be registered before attempting to login. A user can only be logged in from one device at a time (this is a tentative design idea).

**Friend Other User**

A user can request to be friends with any other user in the system. There are no restrictions here.

**Send Private Message**

Private messages can be sent if and only if the two users are friends with each other.

**Modify User Details**

The user cannot modify their date of birth. All other fields can be modified.

**Create Post**

All users can create a post if they are in the system. There are no restrictions here.

**Edit Post**

As long as the user created a post, they will have access to modify the post. However, they will only be able to modify the body of the post, not the title.

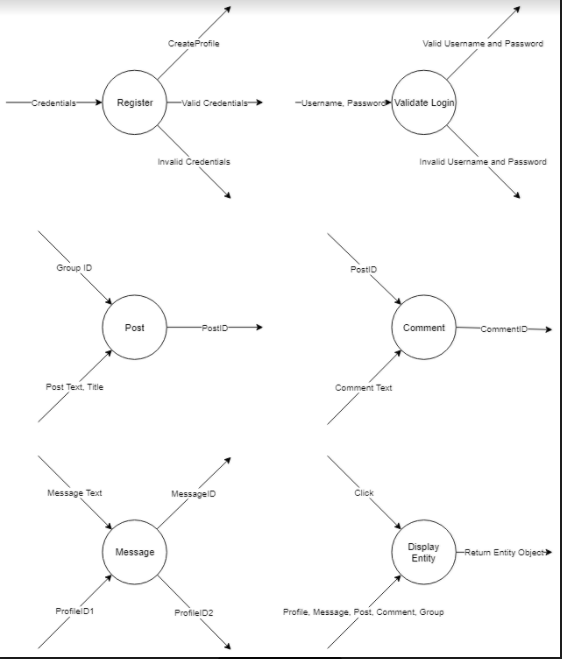
**Comment on Post**

All users can comment on any post as long as they are registered in the system and the post still exists. There are no restrictions here.

**Delete Post**

As long as the user created the post, they will have access to delete the post at any time.

### Diagrams



## Performance Requirements

There are several requirements that will have to be handled in order for CodeFeed to run smoothly. This is assuming normal operating conditions (e.g. server is live, user is connected to the server, user has sufficient bandwidth, etc.) Among these are:

* The system should be able to handle at least 3,500 concurrent users
* The system should be able to manage at least 200,000 user profiles
* The system should be able to fetch and display a page of posts in under a second
* The system should be able to process a new post and update the database in less than a second
* The system should be able to process a new reply and update the database in less than a second
* The system should be able to process a new message and update the database in less than a second

## Exception Conditions / Exception Handling

For now, throwing exceptions must be one of these 3 conditions: client side errors, server side errors, and corrupted data. Here is how we plan to handle each of these errors if they occur:

* The system should handle all web-based errors through the browser console
* The system should return HTTP protocol status codes based on what went wrong (see table bellow)
* When an error occurs, the first step in development prior to product launch is to identify what could go wrong with that error in future steps of the data flow. The second step is to return useful information describing the cause of the exception but at the same time, not leak out important trade secrets / source code
* Exceptions should be handled in such a fashion that the interface is still functional but the function producing the exception is inoperable
* Like accepting and returning data, the application outputs the error in a JSON format. Error JSON format should include the thread ID, and type of error (NullPointer/RuntimeError, etc.)
* Upon Corrupted Data in DB/JSON communique, ideally (a part of) the DB should be formatted, and replaced with a previous working state of the DB. In JSON communique, HTTP should just throw a 400 Status Code
* When we return HTTP protocol status codes, we will only return a few of them. Here is a table of supported status codes. 4XX is client error, 5XX is server error

Below is a table of different HTTP status codes and how CodeFeed will handle them.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **HTTP Status Codes Relevant to CodeFeed** | | | | |
| Code | Name | Description | Supported? |
| 400 | Bad Request | The server cannot or will not process the request due to an apparent client error (e.g. malformed request syntax, size too large, invalid request message framing, or deceptive request routing) | Y |
| 401 | Unauthorized | Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. | Y |
| 402 | Payment Required | Requires a currency transaction to complete. | N |
| 403 | Forbidden | The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort. | Y |
| 404 | Not Found | The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible. | Y |
| 405 | Method Not Allowed | A request method is not supported for the requested resource; for example, a GET request on a form that requires data to be presented via POST, or a PUT request on a read-only resource. | Y |
| 408 | Request Timeout | The server timed out waiting for the request. According to HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time. | Y |
| 500 | Internal Server Error | A generic error message, given when an unexpected condition was encountered and no more specific message is suitable. | Y |
| 501 | Not Implemented | The server either does not recognize the request method, or it lacks the ability to fulfil the request. Usually this implies future availability | Y |
| 502 | Bad Gateway | The server was acting as a gateway or proxy and received an invalid response from the upstream server. | N |
| 503 | Service Unavailable | The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state. | Y |
| 504 | Gateway Timeout | The server was acting as a gateway or proxy and did not receive a timely response from the upstream server. | N |

## Implementation Priorities

Due to limited resources, we will focus our efforts on delivering as many of the most core features first, and work on less essential features after the components that they build on are functioning.

**Basic Operations – Top Priority**

These are each 100% necessary and must be completed before the project is done.

* Database Design
  + Have determined the database management software to use
  + Have defined a schema for the database that the backend will rely on
  + Have creating a working instance of the database that supports the basic functions that the backend will implement
* Basic Backend Functionality
  + Have determined all the software and software packages that the server will run on
  + Have a working server that can interface with the database and the frontend
  + Have the ability to handle user creation, login, and basic abilities to browse posts of the website
* Basic Frontend Functionality
  + Have determined all the JavaScript, HTML, CSS, and any other implementation specific tools to use
  + Have a working website that works at least on desktop web browsers
  + Have webpages to allow user creation, login, and basic abilities to browse posts on the website

**Important Operations – Prioritized**

Each of the following is within the goals of what CodeFeed should be able to do upon completion. It is acceptable to not every item here done, but most should be delivered.

* Database Operation
  + Database should run efficiently and smoothly; there should be no performance issues.
  + Passwords are to be stored securely using strong cryptographic hashing as well as salt values per user.
  + Database should allow for all currently planned relations between entities - includes upvoting / downvoting posts and comments, commenting on posts, following other users, and having categories to tag posts with.
* Backend Functionality
  + Backend should have support for all currently planned functions - includes upvoting / downvoting posts and comments, commenting on posts, following other users, and having categories to tag posts with.
  + Backend should be able to implement basic SSL/TLS functions in order to be secure.
* Frontend Functionality
  + Each page of the website should be formatted well and be easy to read
  + The webpage frontend should have support for all currently planned functions - includes upvoting / downvoting posts and comments, commenting on posts, following other users, and having categories to tag posts with.
  + The website should support SSL/TLS to be secure with data.

**Extra Feature - Low Priority**

Features in this category are non-critical and are only to be started should the previous priority lists are complete. These features are “nice to have” but aren’t as crucial as the others.

* Frontend Extra Features
  + Have an attractive logo
  + Have an attractive, modern style for the website
  + Support mobile devices’ web browsers

## Foreseeable Modifications and Enhancements

**Likely changes**

**Aesthetic modifications**

As we begin build the front end of our site, it is reasonable to expect that we will make many aesthetic changes. At first, our pages may be less visually appealing and more focused on implementing all the features of our site, but as we begin to polish our project, we expect to make some changes in order to make the site more professional and clean, while still maintaining all its intended functionality.

Modifications such as changing the location or placement of a menu, or updating the styling of our landing page, are examples of the changes we will likely be making as the project progresses.

**Performance enhancements**

Some of the most important changes to our project will come in the form of performance enhancements. Initially, we may be more focused on providing the correct functionality, but as we develop our site into more of a finished product, it will be important to make changes that increase the speed of the site and the efficiency of its operations.

This may involve making changes to the way we read or write data from our database, display information on the page, and most likely will also mean improving our code to make sure we are writing functions to execute in the most efficient way possible.

**Unlikely changes**

**“Big picture” modifications**

Currently, we believe we have a very well-thought-out plan for the general functionality and focus of our site. However, it is important to acknowledge that we may end up realizing that a certain intended feature of our site is no longer something we will need, or that a functionality we had no intention of including is actually something we want to focus on. These types of changes do sometimes become necessary, although at the moment we are very satisfied with the general plan for our project and the direction in which we are currently heading.

## Sources of Information

The following descriptions and links will be the resources we use for CodeFeed. Other references may be used at the developer’s expense, but these should be the primary sources:

* What is a RESTful API? - <https://restfulapi.net/>
* MDN JavaScript Docs - <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
* Backend Resources (Includes testing and databases)
  + Python Standard Library - <https://docs.python.org/3/library/index.html>
  + PEP 8 Style Guide - <https://www.python.org/dev/peps/pep-0008/>
  + Flask - <http://flask.pocoo.org/docs/0.12/>
  + Flask-Admin - <http://flask-admin.readthedocs.io/en/latest/index.html>
  + Flask-Assets - <http://flask-assets.readthedocs.io/en/latest>
  + Flask-Celery - <http://ask.github.com/celery/>
  + Flask-fillin - <http://pythonhosted.org/Flask-fillin/>
  + Flask-Heroku - <http://pypi.python.org/pypi/Flask-Heroku>
  + Flask-Login - <http://pythonhosted.org/Flask-Login/>
  + Flask-Mail - <http://pythonhosted.org/Flask-Mail/>
  + Flask-OAuth - <http://pythonhosted.org/Flask-OAuth/>
  + Flask-QueryInspect - <http://pypi.python.org/pypi/Flask-QueryInspect>
  + Flask-Security - <https://pythonhosted.org/Flask-Security/>
  + Flask-SQLAlchemy - <http://flask-sqlalchemy.pocoo.org/>
  + Flask-Testing - <http://pythonhosted.org/Flask-Testing/>
  + Flask-Uploads - <https://flask-uploads.readthedocs.org/en/latest/>
  + Flask-WebTest - <http://flask-webtest.readthedocs.org/>
* Frontend Resources (Includes testing)
  + jQuery - <https://code.jquery.com/>
  + Underscore.js - <https://www.underscorejs.org>
  + Mocha - <https://mochajs.org/>
  + Bootstrap - <https://v4-alpha.getbootstrap.com/>
  + Anime.js - <http://animejs.com/>
  + SweetAlert - <https://sweetalert.js.org/>

## Acceptance Criteria

**Acceptance Criteria for Non-Users of CodeFeed**

1. If I am a non-user of CodeFeed, I am able to sign up for an account
2. I can sign up for an account by entering the following information into the Register page: name, email, desired username, and desired password. After the information is validated, the account will be created
3. I cannot view any page without an account besides Landing page, Login, and Register.
4. The system will notify me via email that my new account needs to be verified
5. I can then click on this link to verify my newly created account. I am then a registered user of CodeFeed

**Acceptance Criteria for Users of CodeFeed**

1. If I am a registered user of CodeFeed, I am able to login via the Login page. I must provide my username and password and hit “Login” in order to do so
2. I will be able now be able to view all pages of CodeFeed as long as I remain logged into the system
3. I will be able to friend other registered users of CodeFeed. I can do so by clicking on the user’s name or searching for the user, then clicking the “add friend” button. The other user will be required to accept the friend request before the two users are friends. Additionally, if the user and I are already friends, this option will be disabled.
4. I will be able to accept or reject friend requests from other users
5. I will be able to create new messages to send to my friends and my friends only. If we are not friends, I cannot send a message to you
6. I can create a new post within a group. To do so, I will navigate to the group, and click “new message” within the group. I will then be prompted to fill in the required fields (see Posts criteria) in order to create a new post within that group
7. I can also delete a post if and only if I am the user who created it
8. I can create comments on the posts within groups. To do so, I will navigate to the post, and click “add comment.” This comment has the same requirements as a post. After I complete all fields, my comment will be logged
9. I can also edit my profile. My profile contains all my basic information I needed to sign up. To do so, I will click on the “my profile” button, then click “edit profile” on this page. I will then fill out the fields, click “save changes,” and then my profile will be updated
10. I will also be able to change my password, but not my username on the profile page

**Acceptance Criteria for Posts of CodeFeed**

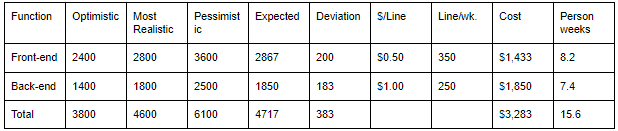
1. As a post in CodeFeed, I can be created or deleted by some user. Only the user who created me can delete me
2. On creation, I will require a message title, a message body, a user to create me, and a group to belong to
3. Users can leave comments on me, which are just subsets of the me (has all the requirements of a post, but local to this post only)

**Acceptance Criteria for Messages of CodeFeed**

1. As a message in the CodeFeed system, I will be sent from user to user who are friends with each other
2. I will contain a title, message body, and who sent the message
3. I can be read or ignored by the user who receives

## Revision History

**LOC Cost Table**



Expected LOC (Lines of Code) = 4717

Deviation = 383

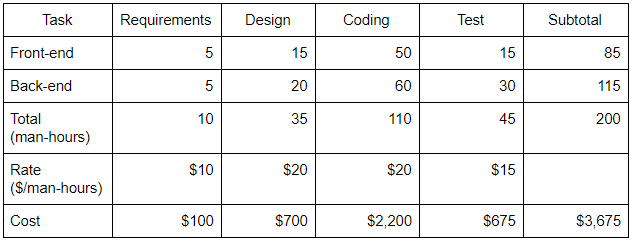
68% range (1 deviation) = 4334 to 5153 LOC

99% range (3 deviations) = 3568 to 5866 LOC

$/LOC = (varies, see chart)

Cost = $1,433 (front-end) + $1,850 (backend) = $3,283 (total)

**Labor Cost/Task Table**



Avg. Cost (estimated cost) = (3,283 + 3,675) / 2 = **$3,479**

*The new dataflow diagram can be found in section 2.2.2*

# SPG Tool Input and Output

