Software Development Plan for **CodeFeed**

Alec Trievel, Tim Platts, Jeremy Kato,

Joseph Kostial, & Albert Yang

**Table of Contents**

1. Scope
   1. Functions
   2. Performance
   3. Limitations
2. Tasks
   1. Summary of tasks
3. Resources
   1. Hardware
   2. Software
   3. People and responsibilities
4. Costs
5. Schedule

*Page Number*

1

1

1

1

1

1

1

1

1

1

1

1

# Scope

CodeFeed is a social media interface for 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.

## Functions

As with virtually any social media platform, there are a variety of functions all users should be able to perform. CodeFeed will initially be able to perform the following functions, but other features could potentially be added later:

* Users should be able to create, delete, and modify their own accounts
* Users should be able to navigate to subgroups that are indicated by certain operating systems, programming languages, topics, etc.
* Users will have their own homepage listing individual, unique attributes
* Users should be able to delete, create, modify, and like/dislike posts
* Users should be able to direct message other users
* Users should be able to friend other users

## Performance

For the time being, CodeFeed should be able to support a userbase roughly the size of a large, public, college campus. e.g. Pitt’s enrollment total is roughly 30,000 students. The application will be built to scale in case it would ever need to be deployed in the future.

Since this application will collect data some may consider sensitive, security will be of the utmost concern. Having these enhanced security measures could possibly hinder performance, but this is a necessary evil in today’s data-driven society.

Additionally, users will be accustomed to real-time updates regarding their content. In other words, once content is posted to the server and saved to the database, the user should immediately see the updates reflected on their display. This feature will be difficult, yet possible, to render in real-time, so its design choices could possibly affect the overall performance of the app.

## Limitations

There are several limitations that will initial be placed on the application due to various security concerns, time restraints, and otherwise superfluous functionality. These include:

* Cannot utilize web-cam services/drivers to communicate.
* Cannot use capital/currency on this application.
* Cannot share current location information.
* Cannot voice call other users.
* Cannot report users for transgressions/verbal abuse/spam.

# Tasks

The tasks will be divided up into portions that represent a traditional full-stack application – a Model View Controller (MVC). There will be a front-end user interface/user experience (UI/UX), back-end code to handle data transformation and display (controller), and a database to handle storing the various pieces of data (model).

The UI must be constructed consisting of the following main elements in mind:

* A landing / home page to greet the user
* A viewing page that displays the categories the user is following
* A viewing page that displays threads/posts sorted by creation date
* A viewing page that displays all threads/posts within a specific category
* A viewing page that shows a thread/post and all replies to it
* A viewing page that shows a user’s profile with their biography and recent posts
* A page for editing the user profile
* A page for submitting and editing threads
* A page for submitting and editing replies
* A page containing a user's inbox to send a receive private messages

The UX must be constructed consisting of the following main elements in mind:

* Users should be able to follow a category
* Users should be able to submit a new thread/post
* Users should be able to like or dislike the thread/post
* Users should be able to like or dislike the replies to the thread/post
* Users should be able to submit their own replies to threads/posts
* Users must be able to friend other users through their profile
* There should be a limited number of server calls to provide a smooth experience. This can be accomplished by making RESTful AJAX calls

The back-end will be responsible for various, “behind the scenes,” functions:

* Pulling data from the database, converting it to JSON, and returning this data based on RESTful HTTP requests
* Re-routing the user if certain actions are taken
* Displaying appropriate errors if something unexpected occurs instead of crashing
* Securely handling, hashing, and saving data to the database

Lastly, the databases must be managed to handle the following elements:

* User profiles: including the username, friend lists, following categories, and thread history
* Threads: including the poster, thread content, posting time, subcategories, and total likes/dislikes.
* Replies: including the poster, posting time, and contents
* Passwords: which must be hashed, salted, and securely stored. The application will *never* store the user’s password in plaintext at anytime
* Private messages: that must include the message, sender, recipient, read status, and sending time

# Resources

## Hardware

The hardware used in development and deployment will be the same. The application will run on the latest 64 bit, Windows 10 environment with 16 GB of RAM and an Intel i7 processor running at 2.00 GHz.

## Software

The software used in development and deployment will be the same. The application will rely on Python 3.6.x, the Flask 0.12.x microframework, and various Flask extensions. The entire extension list is still to be determined, but will most likely include the latest, stable releases of Flask-SQLAlchemy, Flask-RESTplus, Flask-Marshmallow, and Flask-Security. The database of choice will be the 10.1 version of PostgreSQL, but will follow ORM techniques using Flask-SQLAlchemy.

Additionally, the application will be designed to run on the latest version of Chrome (63.0.3239.132). This will ensure the application integrates the latest, stable versions of HTML5, CSS3, and JavaScript (ECMAScript 6).

The developers are free to use whatever IDE they choose in order to encourage a familiar development environment. They must be able to incorporate Github versioning control software into their IDE of choice, however.

## People

Since our design incorporates the features of a full-stack application (front-end, back-end, database), it is essential that all fields are well represented. Additionally, testing an application of such magnitude is a must, so the role of quality assurance analyst will be paramount for a successful launch. Each field will require two developers each. This will allow for maximum communication between groups and the project manager, pair programming, and an overall larger, creative environment.

Below is a matrix of all the personal and their roles. These roles were decided based on confidence levels in each field, and distributed in a way so the work is spread out to assert an efficient workflow.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Project Manager | Front-end | Back-end | Database Admin | Quality Assurance | Salesperson |
|  |  |  |  |  |  |  |
| **Alec** | **X** | **X** |  |  | **X** |  |
| **Tim** |  | **X** |  | **X** |  |  |
| **Albert** |  |  | **X** |  | **X** |  |
| **Jeremy** |  |  |  | **X** |  | **X** |
| **Joseph** |  |  | **X** |  |  | **X** |

# Costs

# Schedule

**Management Questionnaire**

1. **Have you got the commitment of the executive management for the project?**

Yes. We will divide into 2 or 3 groups of 2 people each. Each of these groups will have their own responsibilities and an executive manager. The executive manager will report directly to me with updates, questions, or concerns. This effectively removes the need to "micro manage" each group, which in turn allows the project to flow better

1. **Do all the executive managers know what they have to do?**

Yes. All EMs have been assigned general tasks and weekly tasks they must complete

1. **Has agreement been reached on the resources needed?**

Yes. All the hardware and software we need are either free or we already own it

1. **Have the key milestones been agreed?**

Yes. We will follow the major milestones created by Dr. Chang in addition to weekly milestones

1. **Has the project plan been agreed by the executive management?**

Yes. The plan has been agreed on by all members of the group as well

1. **Are problems/solutions passed to the executive being promptly resolved?**

Yes. Each EM will attempt to file and solve the issues as they arise. If it is a serious issue, the PM and possibly entire group will be notified

1. **Does everyone on the project know what is expected of them?**

Yes. Each group member will be assigned their own roles, tasks, and jobs that need to be completed. We will be in constant communication with each other to assure all jobs are completed.

1. **Do you know what tasks the company have to do?**

Yes. The company will try to develop the software with the following principles in mind: simplicity, a satisfying user experience, and an easy to manage codebase. This may mean cutting or adding features on the fly, but by doing so, we will assert our product is scalable and as progressive as possible.

1. **Do you know what the consultant will provide to the company?**

Yes. The consultant will help guide the developers into making a platform that the end-users will be excited to use. The users should also have a clean and easy-to-understand experience when using CodeFeed, so the consultant’s job will be paramount to assure these features are prevalent in the final release of the application.

1. **Have you identified any inhibitors and ensured they are not on your team?**

One of the largest inhibitors we discovered was that not all members have the same experience level or knowledge of certain aspects of a full stack application. These members will have to learn and read documentation on their own time. I expect them to learn, but I have no issues with holding small training sessions if needed

1. **Have you agreed how, and when, communication with the staff will be handled?**

Yes. We will use Slack to communicate all information outside of meetings

1. **Have dates been set for steering group meetings? Are steering group meetings being held on schedule?**

Yes. We will meet every Friday for at least 2 hours

1. **Will you make the target date?**

While we cannot guarantee this since we did not start coding yet, I am confident we can reach the goal. The group seems to be cohesive, and we all seem to be motivated enough to make the project work.