CS1530A-G06
Max Dudek
Jacob Diecidue
Jared Ranalli
Gurmail Mathon
John Fessler

1.0 Scope

A text-based social media site where users can create text posts and links on either (1) their personal page, or (2) a community page based on a specific topic (e.g. hockey, technology, music, etc). Posts on community pages can be voted up or down (through likes or dislikes), to determine how high it appears in the feeds of people who follow that community (similar to a web forum like Reddit).

1.1 Functions

1. Account Creation
    a. Users are able to register and create a secure account on the website (Devise API) with a unique name and password.
2. Account Pages
    a. Each user will have a page linked to their account. This page displays the user's posts and public information (name, bio, links).
3. Feed Pages
    a. This is the user home page after sign-in where all posts from people and communities followed will appear. This will implement the Recommended Content (6) function below.
4. Text Posts
    a. Users will be able to upload text which will be stored in our database (postgresql). These posts will appear on their profile pages and any community pages associated with the content (by hashtags).
5. Likes/Dislikes
    a. Users will be able to like/dislike a post.
    b. The post will move up or down in the feed based on how many likes/dislikes it accumulates over a certain period of time
6. Recommended Content
    a. Each user will have a recommended content page based on their likes/dislikes
    b. Recommended content will be similar to tags the user has already liked/posted
7. Content Driven Communities
    a. Communities are pages specifically made for certain content. This will appear like a user account page, but it is public and anyone is able to post to the page.

    b. Community pages receive their own unique hashtags, which when someone posts using the hashtag, the post is automatically displayed on the community page.
    c. Anyone is able to create a community, and the community creator is considered its “owner”
    d. The owner is able to edit the pages bio and descriptions
8. Search
    a. The user has the ability to search for user/community pages, topics, and other posts.
    b. A feed is produced of matching users, pages, and posts.

1.2 Performance

- Every user will be able to post a maximum of 65536 (2^{16}) total posts to their page or to a community page
- Posts are limited to 10,000 characters
- Usernames can be up to 20 characters long

1.3 Limitations

- Media like photos/audio/video can not be uploaded directly. However, links can be posted to this type of content hosted elsewhere
- There is no option to privately message other users
- All posts are public, there is no way to hide your page from certain people
- Users can not leave comments on posts. However, it is possible to create a new post in response to an existing post by linking to the original post

2.0 Tasks

1. Set up the Ruby **development environment** for each member of the team, so that they can spin up a locally hosted “Hello World” site
2. Prepare the **Postgres database** to store user account information: usernames, and securely stored passwords
3. Create a bare-bones interface for **creating accounts/logging in**, and connect it to the database
4. Create **user and community pages**, and allow users to **post** on them
5. Allow users to **follow** user and community pages, and have those posts appear in their **feed**
6. Implement **like/dislike system**, which affects how high posts are in the feed
7. **Recommend** the user posts, based on the tags of posts that they like
8. Create a more **aesthetic user interface design**

3.0 Resources

3.1 Hardware

- We will all use our own machines to develop the web app. For the purposes of testing and demonstration, the website will be hosted locally, and not on an external server.

3.2 Software

- Ruby on Rails web application
- PostgreSQL for the database
- Languages: Ruby, HTML/CSS, SQL, Javascript

3.3 People

	Leader	Project Manager	Backend	Frontend (GUI)	Testers	Documentation	Sales
Jake		**X**	**X**				**X**
Max	**X**		**X**			**X**	**X**
Jared				**X**			**X**
John					**X**		**X**
Gurmail					**X**		**X**

Jake is the Manager/backend dev because he has experience with Ruby on Rails. Max is the other backend dev because he has a bit of experience with JavaScript/HTML.

CS 1530 g06 Second Milestone

Software Plan

4.0 Cost (based upon tasks from 2.0)

Step 1. LOC Cost Table

Function	Optimistic	Most	Pessimistic	Expected	Deviation
Dev. Environment	300	400	600	350	100
Postgres database	100	200	400	150	50
User and community pages	500	650	800	575	150
Following and feed works	350	450	600	400	100
Like/dislike system	125	225	300	200	50
Recommendations	700	900	1200	850	200
Fix interface	150	250	350	200	75

Expected LOC (Lines of Code) = 2725

Deviation = 725

67% range = 2000 to 3450

99% range = 550 to 4900

($/LOC) = $14

Cost = $14 * 2725 = $46,325

Step 2. Labor Cost / Task Technique

Function	Requirements	Design	Coding	Test	Subtotal
Dev. Environment	0.5	2.0	1.0	0.5	4.0
Postgres database	0.5	2.0	1.0	0.5	4.0
User and community pages	2.0	4.5	4.0	3.0	13.5
Following and feed works	2.0	5.0	4.0	3.5	14.5
Like/dislike system	1.0	3.0	2.0	1.0	7.0
Recommendations	3.0	5.0	4.0	3.0	15.0
Fix interface	0.5	1.5	1.0	0.5	3.5
Total (man-weeks)	9.5	23.0	17.0	12.0	61.5
Rate ($/man-weeks)	$250	$450	$400	$350	
Cost	$2375	$10350	$6800	$4200	$23,725

Step 3. Avg. Cost (estimated cost) = (46,325 + 23,725) / 2 = $35,025

CS 1530 g06 Second Milestone

5.0 Schedule (based upon tasks from 2.0)

	1	2	3	4	5	6	7	8	9
createAccount									
- requirements	0	x							
- design	0	x							
- code		0		x					
- test			0		x				
readHashtags									
- requirements				0		x			
- design				0		x			
- code						0		x	
- test						0		x	
makePost									
- requirements			0	x					
- design			0	x					
- code				0		x			
- test				0		x			
createCommunity									
- requirements	0		x						
- design	0		x						
- code				0		x			
- test				0		x			
deletePost									
- requirements			0	x					
- design			0	x					
- code				0			x		
- test				0			x		

	1	2	3	4	5	6	7	8	9
showUserPage									
- requirements			0		x				
- design			0		x				
- code					0		x		
- test					0		x		
likePost/dislikePost									
- requirements					0		x		
- design					0		x		
- code							0		x
- test							0		x
showCommunityPage									
- requirements			0		x				
- design			0		x				
- code					0		x		
- test					0		x		
showFeed									
- requirements			0		x				
- design			0		x				
- code					0		x		
- test					0		x		
followUser									
- requirements		0		x					
- design		0		x					
- code						0		x	
- test						0		x	

Quartz

Group 6 Social Media Project

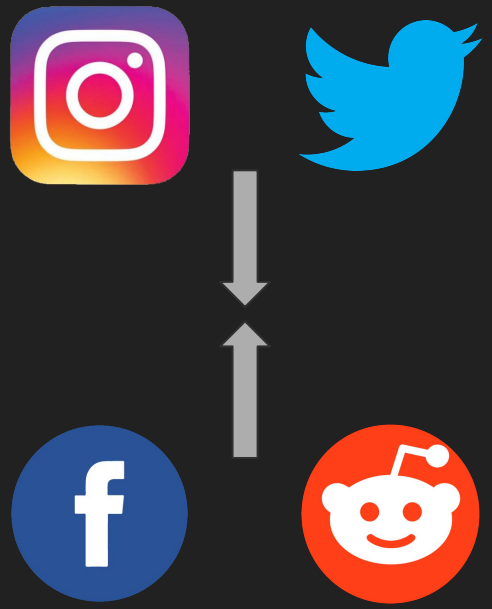
Created By:
Jacob Diecidue, Max Dudek, Jared Ranalli, Gurmail Mathon, John Fessler

The premise

Two types of feeds on websites:

- Follow individual users
  - Instagram, Twitter
- Follow communities / specific interests
  - Facebook communities, Reddit

Quartz lets you follow both users and interests, in a chronologically ordered feed



Functionality of Quartz

- **User Accounts** - Secure Creation/Login
  - **User Feeds** - “Timelines”
- **Content Driven Communities** - #Content
  - **Community Pages** - Where all posts are visible
- **Text Posts** - Posted by Registered Users
  - **Like/Dislike System** - On all posts
- **Recommended Content** - Seen on user feeds
- **Search** - For topics or posts

What Are Our Limitations?

- Media can’t be uploaded directly. Links will instead be used
- No private message functionality
- All posts are public
- No comments. Instead a new post can be created linking to the original post

How Are We Creating Quartz? (Software)

- Quartz is a Web Application
  - Ruby on Rails (Framework)
  - Postgresql (Database)
  - Languages:
    - Ruby, Javascript, HTML, CSS, SQL

Tasks

1. Setup development environment
2. Prepare the Postgres database
3. Create bare bones interface for creating accounts/logging in
4. Create user and community pages, allow users to post on them
5. Allow users to follow pages, have posts appear in feed
6. Implement like/dislike system
7. Implement recommendations
8. Make interface look better

Team Roles

	Leader	Project Manager	Backend	Frontend (GUI)	Testers	Documentation	Sales
Jake		X	X				X
Max	X		X			X	X
Jared				X			X
John					X		X
Gurmail					X		X