Software Requirements Specification

for

A Facebook-like Social Media Platform

Version 1.0 approved

Prepared by Brett Baltz, Robert Dilworth, HeFeng Ou, William Neeley

Bagley College of Engineering | Mississippi State University

09/06/22

1. Introduction

1.1 Purpose

This document will delineate the breadth and mechanics of the website titled, "Hand-in-Hand." At the time of writing, this software is currently on release number one; similarly, the software's revision instance is one as well, given that no other occurrences exist.

The purview of this record will cover the entirety of the project, from the creation of users to the interplay of statuses/timelines. Likewise, this document will encompass the specification/layout of database schemas and the encapsulation and presentation of data (or input and output).

1.2 Document Conventions

The following SRS document will not adhere to an elaborate compositional layout. In terms of the formation of text, the file will consistently use the font, Ariel, with minimal stylistic modification, be it bolding, underlining, or italicizing.

For the ranking of requirements, every specification outlined by the stakeholder will be atomically categorized twofold. That is, each provision will be arranged into a suitable hierarchy based, first and foremost, on the importance of the feature conveyed by the client and, finally, on the rational compartmentalization of the project's team of developers. At the end of the screening process, each requirement statement will have its priority/rating.

1.3 Intended Audience and Reading

Given the ordering of events, this document was penned in such a way to target the following audiences: the project’s collaborators (as this will serve as a roadmap of sorts), project managers (as certain crucial administrative decisions are contained within), and testers (as the responsible team will be following a white-box testing paradigm). All other constituents were not considered during the construction of this file due to the limited scope of the project. For instance, the product will be programmed in such a way that its use does not hamper the customer, leading to the exclusion of users from the intended audience.

The remainder of the document will cover standard material regarding the web application (or software), from its overall description, system features, and other non-functional requirements.

Based upon our audience selection, the intended viewer will, hopefully, possess an intimate knowledge of the project, granting the reader agency while examining the document. Each section and subsection applies to each reader type. In this way, the order in which the content is consumed is inconsequential, but there is a caveat: the file should be reviewed in its entirety, from cover to cover. Granted, following the material in its presented configuration is sufficient as well.

1.4 Product Scope

The aim of our product–and this document to a lesser extent–is to produce a website that mimics the key components of the social media juggernaut, Facebook. Our small-scale project could be equated to a less powerful Myspace, with fewer additive features.

Our objective is to design and develop a Facebook-like social media platform that will have several different functionalities based on the stakeholder's expectations. With our web application, we aspire to enrich our users by (A) imparting convenient connectivity, by way of consumer

interactions, (B) supporting real-time status updates, by way of user timelines, and (C) maintaining responsive feedback, by way of user post-approval/critique in addition to the dissemination of engaging uploads.

In replicating the technology, our website will offer the same fringe benefits common to all social media platforms. Namely, it will bridge the gap between geographical, ethnic, and cultural boundaries, leading to a more connected and unified world. By fostering discourse amongst loved ones, friends, and acquaintances, the web application will deepen/strengthen communal bonds.

However, to achieve our lofty ambitions, the following goals must be satisfied. First, our product requires exposure; without web traffic, our platform will dwindle, crushing our vision. Second, if we manage to garner a following, we must maintain that user base; without consistent usage, our product will peter out. Finally, our branding/marketing must be recognizable; without an easily distinguishable token, we run the risk of fading into an already saturated market.

1.5 References

Presently, this file does not make use of external sources. Granted, if any sophisticated figures, charts, or conventions appear in the document hereafter, they were most likely acquired through prior coursework or made available by instructors/professors.

2. Overall Description

2.1 Product Perspective

Our web application is a unique, stand-alone product completely independent from any existing software. However, once completed, the website will be integrated into our constituent's/stakeholder's application framework. At this point, the service will become a blip in the enterprise's library of exclusive software.

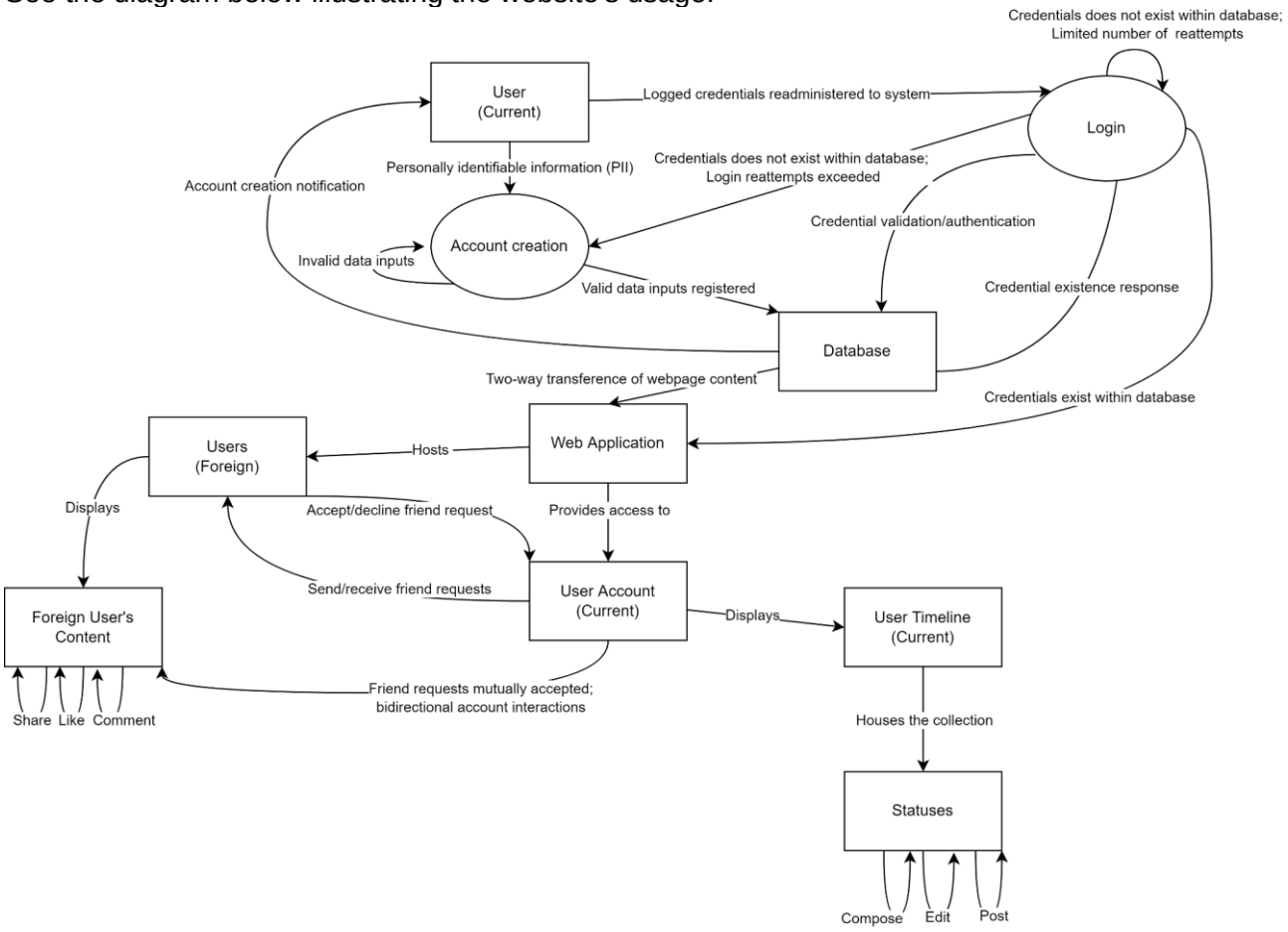
2.2 Product Functions

The following functionalities will be supported by the website. Notably, a user will be able to:

- create/login into his or her account, connect with his or her friends, and access their timelines
    - A ‘timeline’ is a collection of user posts that are organized in such a way that the most recent addition appears first while all other entries follow logically based on the upload time. A timeline can be thought of as a stack. The insertion of posts occurs at the top of the pile. As a result, the last entry of the structure will be the user’s first blurb, notification, etc.
    - A ‘post,’ in the context of the project, is considered text-based information transferred over a metered connection such as the Internet. While currently undetermined, the website may eventually support the sharing of images, but the team will account for that function if the need arises. This feature would likely require a service such as Amazon S3 Buckets to streamline the process, which adds a non-trivial amount of overhead to the undertaking.
- compose, edit, and post statuses on his or her timeline
- send and accept friend requests to and from other users
    - This privilege will, in turn, grant the current user the ability to befriend another user, which enables them to interact with the assets of the recently acquainted associate.

- like, comment, and share the status/timeline post of their friends
    - A ‘like’ is a numerical tally presented to a user when faced with foreign (non-self) posts. A high like ratio indicates that the content of the blurb resonated with, entertained, or uplifted a vast user audience. Seeing as how there is a positive counter, it would not be unusual if a ‘dislike’ or negative enumeration were available. Granted, large corporations have strayed away from its implementation, so our team may conform to the prevailing trend.

See the diagram below illustrating the website's usage.



2.3 User Classes and Characteristics

Due to the limited scope of the project, our software will have a minimal number of user class entities. The individuals who will be using the software are as follows: (A) generic (general) users and (B) moderators (administrators).

To clarify, a generic user can exercise standard privileges regarding the website. Rather, they are limited to only the features and tools made available to them on a 'surface level.' This entails the use of the login mechanism, account alteration, foreign timeline engagement, and post-modification.

Similarly, a moderator is a generic user with elevated privileges within the web application. In terms of set theory, a general user is a subset of the domain of administrators. Moderators can exercise the same features as genetic users, but they are also responsible for the general upkeep of the system, from deleting/censoring posts, to banning/terminating problematic users, to updating/resolving functionality defects.

Concerning other metrics, a generic user will more frequently access their account but will lack the technical expertise and elevated privileges of moderators. Moderators, on the other hand, will less frequently access their accounts but will maintain their systemic modification capabilities in addition to their formal knowledge/skill set.

In terms of the hierarchy of users, the most important entity would have to be moderators as they can alter the collection of web pages viewed by the users; granted, without the user base, the web application would atrophy.

2.4 Operating Environment

Given the hardware specifications of our group's developers, our product would most ideally run in a Windows operating system running Windows 10 or greater with a Chromium-based browser such as Google Chrome, Brave, etc. While our product will be minimally optimized due to time constraints, it is perhaps sufficient for the user to have a setup computationally equivalent to a modern smartphone. Rather, the minimal RAM, CPU, and storage requirements have not been determined, ergo a user is encouraged to use the most up-to-date hardware components and software modules.

In terms of coexisting applications, the application will rely upon an in-house database using Flask-SQLAlchemy to store textual information such as the user's credentials, posts, like count, and list of friend usernames. For multimedia content such as images and videos, the application will depend upon Amazon's simple storage system (S3), a paid service, or Cloudinary, a free service, for rudimentary web-based file storage. Given the limited scope of the project, the team will likely opt for the unpaid service. Take note that cross-platform functionality is not a focal point of our product's design; in that way, we will disclose the best 'configuration' to access the application at a later date.

2.5 Design and Implementation Constraints

The most compelling limiting factor regarding the project is equal parts time and equal parts experience. For a vast majority of the development team, this will be the first large-scale project they have contributed to. This undertaking is not only novel but also quite involved, building upon concepts introduced in prior coursework. In an ideal world, every team member would have a similar coding background, enabling a more conducive collaborative experience. Granted, to fully realize the product, the team will need to be not only motivated but willing to learn. Web page construction, database interfacing, and cloud object storage service (OSS) integration are not Herculean tasks, but the learning curve associated with the comprehension and coding of the concepts could prove Sisyphean. On top of it all, the time crunch of meeting periodic deadlines does not aid in quality code production.

As previously mentioned, the group is constrained by our limited experience with programming languages, which impedes the exploration of equivalent or more simplistic modes of tackling the project. Similarly, cybersecurity and the protection and safeguarding of data are often considered after the fact, leaving systems and services vulnerable to attack. For instance, with SQL, our database could suffer from SQL injection. With cloud or remote storage, our image/video repository could be arbitrarily erased or hijacked by a third party. Overall, the imperative to produce working programs often supersedes security, which is tolerated in smaller projects but is undesirable in scalable projects.

Finally, the distinction between applicable object-oriented schemas and appropriate procedural paradigms is often vague when the scope of the project increases with complexity. There is an undisputed art to elegant object-oriented encapsulation and inheritance, but an incremental methodology is not only tried-and-true but almost always guaranteed to work–despite

the complexity and contrivance of code. All in all, the project's success is highly dependent on external factors, one of which is a choice of programming convention. This especially becomes more significant when we are eventually charged to maintain and update our code.

3. System Features

The ensuing functionalities listed below are recorded in the logical order in which they would be organically accessed during the first instances of utilizing the product.

With any web application, the first task a user is likely to do is (1) create an account with the service. Next, once a user's credentials have been logged into the software's database, those self-same credentials are used to (2) login into the system. Once logged in, a user is immediately granted (3) access [to their] personal timeline; the first screen of the webpage will likely present the current user's timeline. Then, as a user scrutinizes their home page (of sorts), they will likely then want to (4) connect with friends (or other users of the application).

Eventually, the user will feel the urge to customize or fill out their page; they will be allowed to do so via (5) composing, editing, and posting statuses on their timeline.

Meanwhile, once the current user finishes altering their timeline, they will likely aspire to (6) send and accept friend requests from other users. With the accumulation of online acquaintances, the various users of the application will yearn to interact with their peers; this will be facilitated via (7) liking, commenting, and sharing status/timeline posts of associates.

The sequence of features, as presented above, satisfies the necessary components to facilitate the unencumbered use of the system.

3.1 Create an Account

4.1.1 Description and Priority

- Account creation is characterized by the following. First, the user will provide their full name and email address to register their identifying information into the web application's database. As a user fills out the account creation form, certain attributes/fields will have to adhere to strict guidelines. For instance, every username must be unique, and the password must have 8 or more characters among other binding policies. If the user's information is validated, the one who accessed the website is added to the database, permitting access to the system when re-feed his or her authorized credentials.
- Seeing as how most of the web application's capabilities assume that the user has already created an account, the priority of this functionality is relatively High.

4.1.2 Stimulus/Response Sequences

- Access the website's main page. You are presented with various prompts from "Create Account" to "Login" to "Forgot Username/Password." Click the "Create Account" button on the website's initial loading screen.
- Input information into the required fields such as (username, first name, last name, email, password, and password confirmation.
- Once the inputs have been authenticated, if it
  is
  valid, pressing the submit button should redirect the user to the website's homepage (or the user's timeline). An error should appear and the user will be redirected back to the account

creation
screen.

4.1.3 Functional Requirements

(1) Instantiate a database to house the user's personally identifiable information (PII)

(2) Present and construct web page forms for the account creation screen with various attributes or fields.
   (a) Each field should be dictated by various validation tests. If the content populated in the textbox is invalid, the application should throw an error and require the re-entering and alteration of fields.
(3) Fashion a button for submission, that, when pressed, will validate the user's inputs, ensuring that the provided information adheres to all requirements.
(4) Conditionally redirect the user based upon their inputs.
   (a) If the information is valid, the user should be diverted to the website's homepage.
   (b) If the information is invalid, the user should be diverted to the account creation screen for revisions.
(5) Populate the database with the user's information.

REQ-1: Accurately transfer user inputs into the database once validation has taken place.
REQ-2: Precisely handle the error handling of invalid account creation field attributes.

3.2 Login

4.1.1 Description and Priority

- Sign-in is characterized by the following. First, the user will provide their username and password to login into their account on the website. A user's credentials must remain hidden so that bad actors do not masquerade as them online. Furthermore, a user should correctly enter their information into the appropriate prompts within at least 20 tries. Any attempts exceeding that threshold lock the user out of their account. At which point, the only respite would be to utilize the "Forgot Username/Password" feature. After a given session, the user could then potentially logout to safeguard any alterations sustained on a shared device for example.
- Similar to an account's creation, a user must first establish a session with the application before the social media-like functionalities can be performed. In that regard, it should possess a priority that mirrors that of account creation or a designation of High.

4.1.2 Stimulus/Response Sequences

- Access the website's main page. You are presented with various prompts from "Create Account" to "Login" to "Forgot Username/Password." Click the 'Login' button on the website's initial loading screen.
- Input the required fields, which in this case are the user's username and password.
- Next, guarantee that the inputs are valid. If the inputs are invalid, an error should appear and the user will be redirected back to the sign-in screen. If the inputs are validated,

pressing the submit button should sign-in to the user's account and redirect the user to his or her timeline.

- In case the user forgot either their username or password, click the 'Forgot Username/Password' link on the initial loading screen. If the user's credentials have been rejected within the permissible limit of re-attempts (20 tries), then the system can auto-generate a replacement username or password to replace the forgotten credential.

4.1.3 Functional Requirements

(1) Present and construct a form for the sign-in screen including a textbox for the user's credentials or username and password. Additionally, create a link to the password retrieval service titled 'Forgot Username/Password.'
(2) Next, fashion a button for submission, that, when pressed, will validate the user's inputs, ensuring that the provided information matches an existing entry/row in the system's database.
(3) Conditionally redirect the user based upon their inputs:
    (a) If the information is valid, the user should be signed-in to their account and be able to view their timeline.
    (b) If the information is invalid, the user should be diverted to the sign-in screen for a limited number of revisions/attempts (of which 20 is the limit).

REQ-1: Correctly match a user's credentials to that of an existing member within the database
REQ-2: Precisely handle the error handling of invalid login attempts.

3.3 Access Timeline

4.1.1 Description and Priority

- Timeline acquisition and access involve the following. A user must interact with his or her account that has been made public to the masses. To customize an account, the alterations can be made: changing the accent color selected, modifying the profile picture used, and altering the background banner image displayed. To edit a user's biographic information located in their 'About' section, the web page owner can provide a brief description of themselves, their objective(s), their total number of likes from all posts, the tally of the current number of friends, contact information, and any other linked pertinent accounts. Finally, the user can remove their friends if they have fallen out of fellowship with one another.
- The priority of this function is moderate to low as profile customization can be beneficial for the user to satisfy their creative urges, but it is not essential or required in the grand scheme of things.

4.1.2 Stimulus/Response Sequences

- Establish a two-way connection with the system's database server if one has not already been formed.
- Ensure that the following has occurred:
    - the user of the current session matches that username made available in the web application's central hub.
    - that every editable portion of the website can be altered as necessary

- that the articles and sections of the user web page can be modified, perhaps, by executing a well-defined, automated workflow

4.1.3 Functional Requirements

(1) Establish a session with the application from a prior login/sign-in event.
(2) Fashion and deploy a multitude of wickets to facilitate the alteration of various user webpage assets.
(3) Display the webpage in its established configuration, accounting for updates when changes have occurred.

REQ-1: Permit changes to an account's cosmetics, while maintaining reliability and correctness.

REQ-2: Remember aesthetic changes and re-institute them when a session is closed and created.

3.4 Compose, Edit, and Post Statuses on Timeline

4.1.1 Description and Priority

- Timeline creation and alteration are composed of the following. First, a user must generate statuses containing either text or media such as images or video. Next, a user must possess the ability to modify statuses once they have been posted. Then, a user should be able to release posts into a wider community arena with the intent of fostering user interactions. Finally, a user must reserve the right to terminate a status.
- While not a core feature, this functionality is significant nonetheless; therefore, it will receive priority of Moderate.

4.1.2 Stimulus/Response Sequences

- Establish a two-way connection with the system's database server if one has not already been formed.
- Ensure that the following has occurred:
  - that the user of the current session matches the username made available in the web application's central hub
  - that every editable portion of the website can be altered as necessary
  - that a post can be created, changed, released, or deleted, perhaps, creating a temporary status for testing purposes

4.1.3 Functional Requirements

(1) Establish a two-way connection with the system's database server if one has not already been formed
(2) Establish a session with the application from a prior login/sign-in event
(3) Fashion and deploy a multitude of wickets to facilitate the alteration of various users' webpage assets
(4) Display the webpage in its established configuration, accounting for updates when changes have occurred

REQ-1: Maintain the timeline database effectively and efficiently.

REQ-2: Reduce redundancy among relations within the database.

3.5 Send and Accept Friend Requests and Connect with Friends

4.1.1 Description and Priority

- Friend requests are included to achieve the following: to broaden the user's horizons by chatting with a large and diverse set of people, transmit friend requests to unique and entertaining individuals, and intercept and respond favorably or unfavorably to friend requests.
- The handling of friend requests has a High to Moderate priority since it is the crux of the social media experience. Granted, it is not the most feature, since it is dependent on another function's successful completion.

4.1.2 Stimulus/Response Sequences

- Establish a two-way connection with the system's database server if one has not already been formed.
- Ensure that the following has occurred:
    - that the user of the current session matches the username made available in the web application's central hub
    - that every editable portion of the website can be altered as necessary
    - that a friend request message can be sent and handled correctly, perhaps by sending a dummy request to various users

4.1.3 Functional Requirements

(1) Establish a two-way connection with the system's database server if one has not already been formed
(2) Establish a session with the application from a prior login/sign-in event
(3) Fashion and deploy a multitude of wickets to facilitate the alteration of various users' webpage assets
(4) Display the webpage in its established configuration, accounting for updates when changes have occurred

REQ-1: Ensure that the correct recipients receive their appropriate friend requests.

REQ-2: Once a relationship is established, the linking of accounts reveals otherwise obscured or hidden user timeline content.

3.6 Like, Comment, and Share Status/Timelines

4.1.1 Description and Priority

- Status maintenance involves the following: incrementing a counter that represents the likability or the general reception of a given user's post within their timeline, engaging users by responding to their timeline posts, and forwarding compelling posts to a user's friend group.

- While conducive to an expressive environment, this particular functionality received a Moderate priority. The reason is that users could still upload content in a self-contained bubble, devoid of interesting discourse. Such a social media platform would be boring but operable nonetheless.

4.1.2 Stimulus/Response Sequences

- Establish a two-way connection with the system's database server if one has not already been formed.
- Ensure that the following has occurred:
    - that the user of the current session matches the username made available in the web application's central hub
    - that every editable portion of the website can be altered as necessary
    - that a given post can be:
        - modified in terms of its like by pressing a "Like" button
        - engaged with by linking the responses of foreign users to a well-rounded timeline database table/relvar
        - transmitted to other users by including their unique user ID to a "shared_with" attribute of a timeline database
        - the process could be tested by uploading a temporary timeline post with the intent of running automated audits to check the validity or repeatability of the function

4.1.3 Functional Requirements

(1) Establish a two-way connection with the system's database server if one has not already been formed
(2) Establish a session with the application from a prior login/sign-in event
(3) Fashion and deploy a multitude of wickets to facilitate the alteration of various users' webpage assets
(4) Display the webpage in its established configuration, accounting for updates when changes have occurred

REQ-1: User comments are easily distinguishable from the original post.
REQ-2: Shared content presents the metadata of the original poster.

4. Other Nonfunctional Requirements

4.1 Performance Requirements

For all intents and purposes, the most crucial requirement is functionality; that is, does the web application function as intended based upon the stakeholder's requirements? Given the limited scope of the product and the technical underpinnings of certain features, our team prioritizes usability over the comfort of use. In that regard, an unoptimized program that completes all the tasks we have set out for it is a successful code, regardless of how slow or inefficient it may be.

4.2 Safety Requirements

Again, as previously articulated, workability does not immediately translate to safety. With an online application, the concept of safety can be housed within the umbrella of security outside of instances of slander, sexual harassment, stalking, or physical assault. The aforementioned circumstances stem from the user's comfortability with sharing (or perhaps oversharing) highly personal aspects of their daily life. With this in mind, we, as developers, cannot stand over the user and provide them with guidance on what he or she can and cannot post. In that way, all safety concerns could be addressed with policies that users must consent to during the creation of their accounts. Overall, it is one thing to create a tool, but it is another thing entirely to supervise the use of the tool once it has been placed into the hands of consumers. Rather, the development team is not completely absolved of all wrongdoings perpetrated by our systems/software, but we recognize the extent to which we can police our creation. In that way, reading the fine print will be our primary mechanism in place for user safety.

4.3 Security Requirements

Security encompasses not only the composition of workable code but code formulated in such a way with minimal program-breaking bugs. A bug could (A) facilitate SQL injection, compromising our user's credentials and other metadata, or (B) permit the leakage of assets stored in the cloud, like our users' images and videos.

Regarding SQL injection, all SQL queries can be written as canned transactions with the program feeding query inputs from a predefined tuple/list into a query skeleton outlining the action to be performed. Disallowing the user to directly interface with the creation of queries works toward thwarting SQL injection.

Concerning the leakage of user data, our software could implement a cryptographic hash encrypting passwords with algorithms such as BCRYPT, SCRYPT, ARGON2, or PBKDF2. Using algorithms such as MD5 or SHA would do little in the way of safeguarding data as the implementations are circumstantially easy to decrypt.

To check for robot/automated accounts, the account creation subroutine could send out a confirmation email to the web address provided during the registration process.

In regards to cloud storage, there is little that can be done by way of preservation or protection. Our team does not own the servers, nor can we prevent some individuals with administrator privileges from arbitrarily terminating our external storage. Better yet, our group cannot verify that the vendor has placed sufficient safeguards in place to circumvent unauthorized access. For all intents and purposes, the security protocol regarding the external storage of files is that there is no policy; this is a calculated risk we are incurring for the use of free and open source software.

Finally, there is the informal threat of cross-infectability (or the compromising of a shared password among multiple platforms) and the voluntary disclosure of credentials (either via the mouth, paper, or data). To address this attack vector, we could entertain the use of two-factor authentication, but–like with enforcing encryption–doing so would add a non-trivial amount of overhead to the project. Similarly, we could enact an upper limit on the number of times an incorrect password can be fed into the login prompt. To provide sufficient protection and avoid the situation of locking out the user, an upper limit of 20 attempts would be sufficient.

In short, securing our code is a considerable undertaking that should not be ignored, but will likely wane due to time constraints. Again, functional code is not always impregnable, and unassailable code is not always useful.

4.4 Software Quality Attributes

For any social media platform, aesthetics (how the web page looks), reliability (the degree to which functions accurately perform their workload), and correctness (everything is in its place and operates as it should) are most important.

The first attribute is somewhat self-explanatory. An unappealing application will deter users; a disorienting layout will distract users; a poor choice of colors will disengage users; the size of the text will irritate users. Striking a balance or maintaining synergy amidst the web application's assets is of the utmost importance for users as they will likely view them more often than developers.

Reliability is similarly defined. The application should perform to its specifications under all circumstances without fault. Variance is undesirable, and will likely leave a bad taste in the mouths of our users if detected. For instance, posting an image should always result in a new status entry appearing on a user's timeline. When a user suspects that a function is not performing adequately, correctness is also evoked.

Correctness delves more into the presentation of data rather than the repeatability of functions. For example, when uploading a textual status, its content should match whatever was entered in the textbook word-for-word. Anything less not only breaches correctness but also reliability as the process cannot be duplicated properly nor can it handle input/output to an acceptable degree.

5. Other Requirements

To reduce redundancy within our database, we will seek to normalize our tables and other constructs placing them in Boyce-Codd Normal form (BCNF) where applicable.

In terms of legality, we could research what goes into formal, binding user agreements to absolve our company of any infractions committed over or utilizing our software or systems.

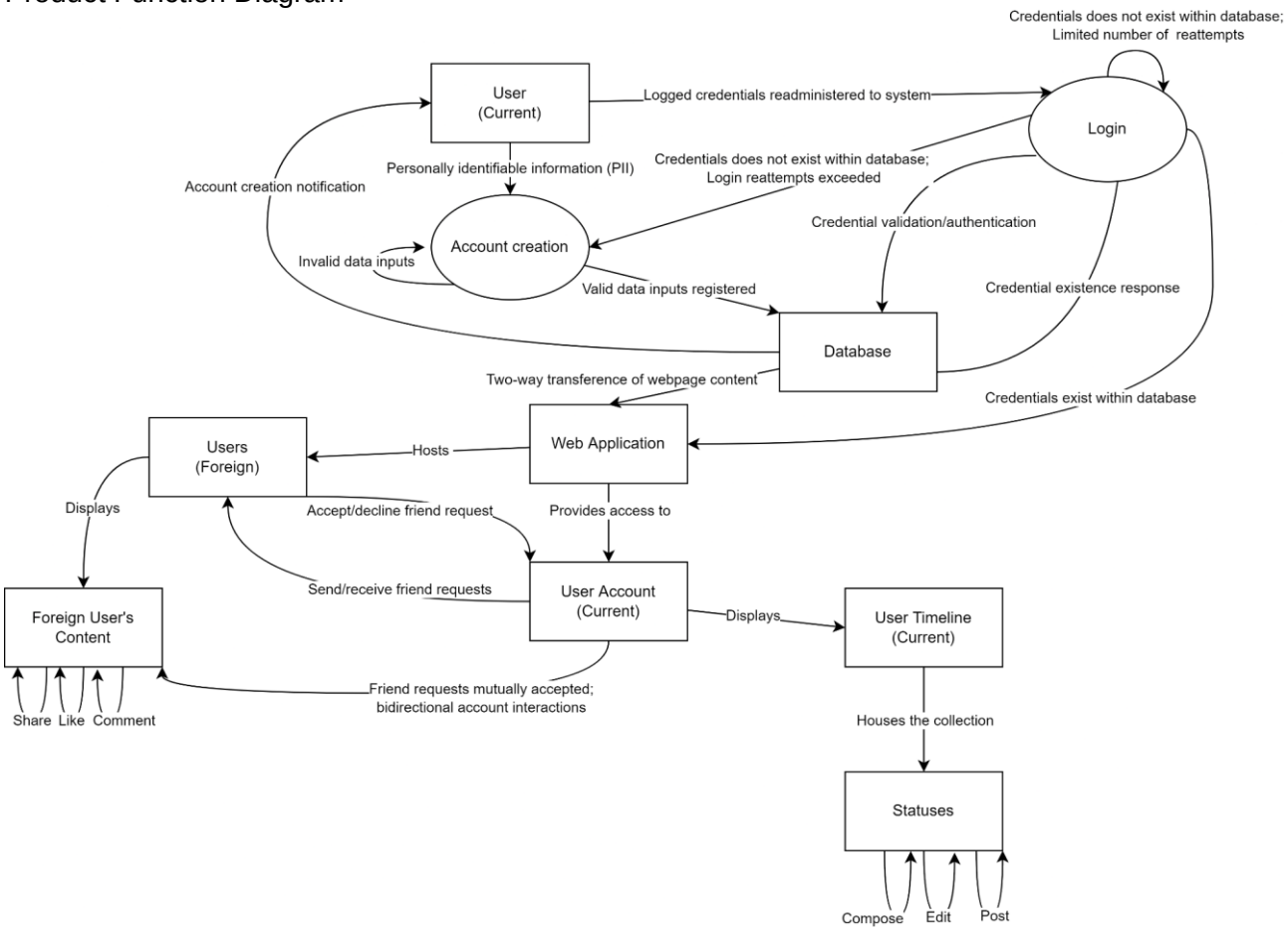
Appendix A: Glossary

A. Boyce-Codd Normal Form: a normalized database table or relvar that has been purged of redundancy in terms of foreign, alternate, and composite keys (or linkages to other tables or relvars)
B. SQL Injection: the introduction of malicious SQL queries into a database, often with the intent of performing a generic "SELECT * FROM TABLE" query to retrieve the contents of an otherwise secure table
C. Cloud Object Storage Service (OSS): external storage (or server) hosted by a third-party primarily for web-based computing
D. BCRYPT, SCRYPT, ARGON2, and PBKDF2: encryption algorithms that are relatively "slow" to crack or decrypt
E. MD5 or SHA: encryption algorithms that are relatively "fast' to crack or decrypt
F. Amazon Simple Storage Service (S3): a cloud object storage service provided by Amazon for a fee
G. Cloudinary: a free cloud object storage service alternative
H. Hand-in-Hand: "with hands clasped; in close relation" (Merriam-Webster Dictionary, n.d.)
I. MySpace: an obsolete and antiquated social media platform usurped by Facebook
J. Facebook: a mainstream social media platform; also referred to as Meta

K. canned SQL transaction: a predefined SQL query whose behavior has been tested and observed that takes in attributes from a list or tuple; one of many preventative measures for mitigating SQL injection

Appendix B: Analysis Models

Product Function Diagram



Appendix C: To Be Determined List

NA