CEN4010 – Principles of Software Engineering.

Fall 2021

Relay

Group 10 – Emanon.

Team members

Andrew Sexton [asexton2020@fau.edu](mailto:asexton2020@fau.edu) (Team Lead), Maxon Corvil [mcorvil2018@fau.edu](mailto:mcorvil2018@fau.edu),

Ashley Davis [davisashley2020@fau.edu](mailto:davisashley2020@fau.edu), Patrick Messina [pmessina2017@fau.edu](mailto:pmessina2017@fau.edu),

Michael Niebauer [mniebaue@fau.edu](mailto:mniebaue@fau.edu)

Milestone 3 – Project Proposal.

October 25, 2021

Revision Dates

|  |  |  |
| --- | --- | --- |
| Revision | Date | Changes |
| 1 | 10/17/2021 | Initial changes based on instructor feedback, initial skills risk assessment and functionality priorities added, |
| 2 | 10/24/2021 | Updates to skills risks adding detail. Updates to use cases and functional requirements based on lessons learned while developing prototype |
| 3 | 10/24/2021 | Added UML diagrams |
| 4 | 10/25/2021 | Updated data definitions |
| 5 | 10/25/2021 | Added more info to high-level functional requirements and database organization. Minor consistency edits throughout document. Added page listing features that are ready for testing with information on not yet implemented features . |

**Features and Functionalities for Testing Vertical Prototype**

* **Account Creation** – All error checking is in place except for verifying that the user entered a working email.
* **Logging in** – Logging in is implemented using username and password.
* **Create Post** – Currently partially implemented, prototype allows users to create posts with a title, text-only content, and select one of three privacy levels. All other related features NYI.
* **View posts** – Prototype provides two methods to view posts: a community page that currently only displays community posts, and a feed page that displays posts by friends and best friends only. Privacy restrictions are in place.
* **Chatrooms** – Not yet implemented.
* **Profiles** – User can view their own profile, which displays their posts, friends, received friend requests, and sent friend requests. The profile’s list of friends displays the friendship level and allows a user to unfriend any of their friends. Users may also view other users’ profiles if they are logged in.
* **Friend Requests** –When viewing another user’s profile, a user may send a friend request, which the receiving user may accept as a Friend, accept as a Best Friend, or reject. Currently there is no way to adjust between Friend and Best Friend levels after the request is accepted.
* **Search** – Users may currently search for other users by entering a search term into the search bar.

1. **Executive Summary**

Keeping in touch with friends and family can be a difficult thing, especially when people are no longer able to regularly meet to socialize in person. This is an issue that has become extremely apparent since the start of quarantines during the Covid-19 pandemic. Many social media sites that existed prior to the pandemic provide some method for staying connected to friends and family, but most of them fail to come close to emulating the experience of in person socialization.

Relay is a new social media site that that intends to improve upon existing platforms by creating higher quality connections between its users. Relay is a tool for users to keep in touch and maintain a strong connection with their family and friends whenever in person meetups are not possible, such as during lockdowns or when friends move long distances. It is intended to be a one stop site for users to share details about their day-to-day lives with those they care about.

Relay will provide users with the expected features of a social media site, posts, a friends list, image, and video sharing. Users will also be given enough control over their posts so that they can feel comfortable sharing any detail of their lives and knowing that only the intended recipients will see the news. Additionally, Relay lets users have dynamic meetups where they can socialize in real time, coming closer to a real, in-person meetup than other social media sites. Unlike some other social media sites that have become tools for advertisers, Relay is focused on connections between people, not businesses, and so accounts are restricted to only real people. This is one of the many ways that Relay hopes to help users maintain a deep connection with those they care about.

1. **Competitive Analysis**

In this section Relay is compared to some of its competitors on the following metrics: the types of posts available to each site’s users, privacy options for posts, the implementation of a live chat feature, implementation of an activity, ease of use, and account security. Each metric will be rated on a 5-point scale where 1 is bad, 2 is subpar, 3 is decent, 4 is good, and 5 is extremely good. Five competitors were chosen based on their potential for connecting users.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Relay** | **Twitter** | **Tumblr** | **Facebook** | **Snapchat** | **Instagram** |
| **Post Types** | **4** | **2** | **5** | **5** | **3** | **3** |
| **Post Privacy** | **5** | **3** | **1** | **5** | **5** | **4** |
| **Live Chatrooms** | **4** | **3** | **2** | **4** | **5** | **4** |
| **Activity Status** | **5** | **1** | **2** | **4** | **3** | **2** |
| **Ease of Use** | **5** | **4** | **2** | **1** | **5** | **4** |
| **Account Security** | **3** | **4** | **4** | **5** | **3** | **4** |

**Relay (4.33)** <https://lamp.cse.fau.edu/~cen4010_fa21_g10/>

Relay will have default posts of 200 characters with an option to create longer posts with a 200 character teaser that will appear when seen on a feed. Posts will also support images and video content. Relay will allow users to specify the privacy level of each of their posts individually, restricting whether each post can be seen by anyone, only their friends, or only a subset of their friends. Relay will also have live chatrooms which provide a space for users to socialize with their friends in real time. Relay will also have activity statuses that will let a user’s friends know when they are online and available, busy, or offline. Relay will have a straightforward design that makes it easy for users to make posts, see their friends’ posts, or join chatrooms. Relay plans to implement an optional 2-Factor Authentication (2FA) for users to secure their accounts.

**Twitter (2.83)** <https://twitter.com/>

Twitter does not provide posts longer than 280 characters by default, but it does let users make multiple posts at once that are linked in a thread. Posts may also include images and video. Twitter has a binary, account wide setting for post privacy where all posts by an account are either public or restricted to their followers. Twitter does not have live chatrooms, but their direct messages/group conversation feature is similar as it provides private conversations between groups of users. Twitter does not have built in method for tracking who is online, even for followers. Users can still estimate who is active by checking timestamps on recent posts. Twitter has a well-designed interface with a post feed and simple search bar. Twitter has 2FA and the ability to require a phone number or email address to reset a password.

**Tumblr (2.67)** <https://www.tumblr.com/>

Tumblr’s text posts do not have a character limit, and may include images, video, audio, quotes, and Chats which resemble a script for a movie or play. Tumblr posts are viewable by anyone using the website, the exception is private posts which are only viewable by their creator. Tumblr has a direct messaging feature that allows two users to have a private conversation, but does not have group chats. Tumblr has also has a simple green dot to indicate when a user is active. Making posts and searching for content are straightforward, however, long posts are not shortened when they appear in a feed, which can be an annoyance. Tumblr also provides 2FA and will send emails when there are logins on new devices.

**Facebook (4)** <https://www.facebook.com/>

Facebook posts have a roughly 60,000 character limit and truncates long posts to roughly 500 characters on a user’s feed. Posts can contain a variety of media including images, video, and audio. Facebook has a strong post privacy feature that allows users to specify whether a post is public or friends-only and specify a subset of friends to include or exclude. Facebook allows users to have private group chats where they can share messages and media or have conversations over voice or video. Facebook shares users’ activity status with their friends by default but gives users the option of manually managing which friends can see their status or hiding it entirely. Facebook is not easy for new users learn how to navigate and use. It has many features that clutter the screen and make navigating the site difficult for new users. Facebook has many security features including 2FA and login alerts for new devices, it also has optional encryption for emails from Facebook.

**Snapchat** **(4)** <https://www.snapchat.com/>

Snapchat is a phone app for sharing images and video that are deleted after being viewed. Each video is up to 10 seconds long, but videos can be grouped up to one minute in length. Users can select which friends are sent their posts on a case-by-case basis, or post to their Story which can be configured so that anyone can see, or just friends. Snapchat also lets users privately message each other in groups of up to 32 users, supporting text, video, and voice. Snapchat does not let users see which of their friends are currently using the app by default, but in group chats users can see when the other members are typing. Additionally, users can see whether their friends have viewed their posts or not. Snapchat has a UI similar to a camera app with a navigation bar along the bottom for accessing its features. Snapchat also has 2FA for accounts.

**Instagram (3.5)** <https://www.instagram.com/>

Instagram allows users to post images and video, with an optional caption of up to 2,200 characters. There is no post-by-post privacy control for default posts except for making an account private so that only followers can see the account’s posts and new followers must be approved. For more privacy control, users can create a Story post which lets them choose between sending the Story as a direct message to one or more users, posting it to their Story, or posting it to their Close Friends list. Followers can be manually removed from the Story to hide it from them and must be added to the Close Friends list to see Close Friends Stories. Instagram also has group chats where two or more users can have a conversation via text, voice, or video. By default, Instagram also allows users to see the activity status of their followers, or people who they receive a message from. If users can turn this feature off, their activity is hidden from everyone, but they can no longer see the activity of anyone else. Instagram is simple to use for both making posts and searching for content. Some features, like the chatroom, are only accessible in specific places on the app. Instagram supports optional 2FA and maintains a list of official emails sent from Instagram going back 14 days for each account.

**Planned Advantages**

Relay’s goal is to be more focused on connecting its users than any other social media site. Many other sites make it possible for users to have conversations with each other, but few emphasize users actively talking to each other at the same time. Relay will feature live chatrooms that only allow users who are currently online to participate in, strengthening the connection between users as they know other participants are active in the conversation. Relay will also aim to make it easier for users to share details about their life than other sites by allowing a wide variety of posts. We also aim to give users enough control over their posts so that they feel comfortable sharing news with their friends and family.

1. **Data Definitions**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Meaning | Usage | Description |
| User | Actor | Use Case scenarios | General term for person navigating the site |
| Account | Data | Use case scenario | Registration information that is stored in the website database in connection to a member of the site |
| Registration   * Email      * Username * Password * First name * Last name | User service  Data  Data  Data  Data  Data | Use case scenario  Use case scenario  Use case scenario  Security  Use case scenario  Use case scenario | Allows a user to sign up for an account to the site  Email of user to add security measures to the account  User generated name that is used to represent the member’s account  User generated password to access account. At least 8 characters long contains 1+ numbers, 1+ uppercase letters, and 1+ lowercase letters.  The first name of the user creating an account  The last name of the user creating an account |
| Profile | Data | Use case scenario | Contains information on a user such as name and email, including a short bio describing the user |
| Member | Actor | Use case scenario | User with a registered account |
| Non-member | Actor | Use case scenario | User who does not have an account registered with the site |
| Activity status | Feature | Member status | Indicates whether a user is online, busy, or offline |
| Login | User Interface | Use case scenario | Allows members to sign into their account. Requires user credentials (e.g., username and password) |
| Log out | User Interface | Use case scenario | Exit out of user account |
| Sent request | Feature | User service | A request to another member seeking to become friends |
| Received request   * Accept      * Reject | Feature  Privacy feature  Privacy feature | User service  User service  User service | A request sent by another member seeking to become friends  User approves the invitation  allowing both members to become friends  The member requesting friendship is declined. Neither member is added to the other user’s friends list |
| Friend | Actor | Use case scenario | Another member that has previously accepted a Friend Request from the member being discussed |
| Best Friend | Actor | Use case scenario | A Friend that has had their Friendship Level increased beyond the default of Friend |
| Non-friend | Actor | Use case scenario | Another member that is not a Friend of the member being discussed |
| Friends List | User interface | Use case scenario | A list of the all the member’s mutually accepted friend requests |
| Friendship level | Feature | Use case scenario | A rank among friends to determine privacy level of shared content. By default, the level is set to friend. |
| Friend | Actor | Use case scenario | A list of the all the member’s mutually accepted friend requests |
| Best Friend | Actor | Use case scenario | A Friend that has had their Friendship Level increased beyond the default of Friend |
| Non-friend | Actor | Use case scenario | Another member that is not a Friend of the member being discussed |
| Community | User interface | Use case scenario | Posts shared publicly to any user of the site with no restriction, including non-friends |
| Website | User Interface | User interface | Front-end display for user interaction |
| Relay | Domain | Use case scenario | Represents the website and all associated web pages |
| Home page | User Interface | Default page | The first page the user encounters. By default, is the community page. Allows a user to view community posts, log in, or register for an account |
| Feed | User interface | Use case scenario | Contains posts of followed friends and best friends. Allows members to interact with the posts |
| Post | User interface | Use case scenario | Section of the website that allows users to create short or long posts. Contains option to select who can view the content- best friends, friends, or community |
| Long post | Data | User generated content | User post containing 200 characters or more. May contain image or video |
| Short post | Data | User generated content | User post containing less than 200 characters. May contain image or video |
| Video | Data | Feature | Allows a user to upload a video to their editing post. Formats include .mp4 and .mov |
| Images | Data | Feature | Allows a user to upload photos to their editing post. Formats include .png and .jpg |
| Like | Data | User service | Allows members to react positively to posts with the click of a button |
| Comment | Data | User service | Reply to user posts |
| Chatroom | Feature | Live message room | Private messages between two or more friends that allow a real time connection |
| Filter | Data | User service | User can sort posts based on relevancy and date |
| Delete | Data | User service | Remove data such as friends, comments, or posts |
| Search | User interface | User service | User can look up posts and members related to the keywords inserted |
| System | Hardware platform and services | Use case scenario | Front-end design, back-end services, MySQL database, and all code |
| Friends Only | Security feature | Privacy | Privacy setting for shared content. Minimal level of access to shared content. Non-friends can only see one’s name and profile photo. All other content is hidden |
| Best Friends Only | Security feature | Privacy | Privacy setting for shared content. Only Best Friends can view the content |
| Community | Security feature | Privacy | Privacy setting for shared content. Content is shared to any user on the site with no restriction |
| Follower | Feature | User service | A member who has sent a pending Friend Request and will see the recipient’s community posts in their feed |

1. **Overview, Scenarios, and Use Cases**

**Use Case – Homepage:**

The user goes to the home page and wants to see global posts and has not logged on or created an account.

1. **Description:** Use case describes the first thing that the user sees when going to the website.

2. **Actors**:

2.1 Existing user that is not logged in or new user

2.2 System

3. **Preconditions**:

3.1 User has an active internet connection

4. **Primary Flow of Events**:

4.1. User arrives on web page.

4.2. User has access to community posts

4.3. User can either create an account or login from this page

5. **Alternate Flows**

5.1 **If the user attempts to interact with posts**

If user tries to interact with posts, such as by commenting or liking the post. The user will be forced to either login or create an account.

1. Website notifies user that they need an account to proceed

2. Return user back to previous post

**Use Case – Sign Up:**

When a user goes to the homepage, they will be shown community posts, but they will need to login or create an account if they want to interact with posts or create a post. If they do not have an account, they can click on sign up to create a new account.

1. **Description:** Use case describes the process of the user creating an account.

2. **Actors**:

2.1 New user

2.2 System

3. **Preconditions**:

3.1 User has an active internet connection.

3.2 User has an email not connected to another account.

4. **Primary Flow of Events**:

4.1 User arrives on web page.

4.2 User clicks on create account.

4.3 User enters a username, email, password, and first and last name.

5. **Alternate Flows**

5.1 **If the user attempts to use an email or username that is already used**

If user tries to use an existing email or username in step 4.3

1. Website notifies user that the email or username Is already being used

2. The user is given the option to change email or username.

**Use Case – Login:**

When a user goes to the homepage, they will be shown community posts, but the user will need to login or create an account if they want to interact with post or create a post. If they have an account, they can just sign on with their credentials.

1. **Description:** Use case describes the process of a user logging into an account.

2. **Actors**:

2.1 Existing user

2.2 System

3. **Preconditions**:

3.1 User has an active internet connection.

3.2 User has an existing username connected to an account.

3.3 User isn’t already logged in

4. **Primary Flow of Events**:

4.1 User arrives on web page.

4.2 User clicks on sign in

4.3 User enters a valid username and password

4.4 User is taken to their personal feed

5. **Alternate Flows**

5.1 **If the entered credentials aren’t found on the database**

In this situation, the user entered credentials that aren’t found on the database.

1. Website notifies user that one or both of the fields need to be changed

2. Gives user the option to reenter the credentials or to create an account.

**Use Case – Create Short Post:**

After a member has already verified their account, they get the ability to create posts, when creating a post, the member can select post type (long or short if under 200 characters). Additionally, the member can add images or videos. Finally, members get the option of who can view each post. Posts can be either community, friends only, or best friends only.

1. **Description:** Use case describes the process of the member creating a post, customizing it and selecting who can view it.

2. **Actors**:

2.1 Member

2.2 System

3. **Preconditions**:

3.1 Member has an active internet connection.

3.2 Member has already logged in

4. **Primary Flow of Events**:

4.1 Member is already logged on

4.2 Member clicks on create new post

4.3 Member customizes post using 200 characters or less

4.4 Member selects the privacy settings for this post

5. **Alternate Flows**

5.1 **If the post is over 200 characters**

In this situation, the member enters over 200 characters. Website notifies the member that the post has become a long post and they can either send the post as is, delete it, or change it into a long post

1. If this is a long post, the member has the option to create a teaser for the post that follows the short post restrictions. There is no cap for the length of the long post.
2. If the member wants to post as is then the post will be sent as a short post
3. If member deletes the post, then nothing will be saved or posted

5.2 **If the member cancels the post**

In this situation, the member opts to not create a post at all. Website asks for confirmation to delete the post permanently or store the post as a draft.

1. If member deletes the post, then they will be returned to dashboard
2. If the member saves to drafts , then it will be saved in the database but not viewable by anyone.

**Use Case – Create Long Post:**

After a member has already verified their account, they get the ability to create posts, when creating a post, the member can select post type (long or short if under 200 characters). This case covers when the member selects “Long Post”. Additionally, the member can add images or videos. Finally, members get the option of who can view each post. Posts can be either community, friends only, or best friends only.

1. **Description:** Use case describes the process of the member creating a Long post, customizing it and selecting who can view it.

2. **Actors**:

2.1 Member

2.2 System

3. **Preconditions**:

3.1 Member has an active internet connection.

3.2 Member has already logged in

3.3 Member has surpassed 200 characters or user has already opted for a long post

4. **Primary Flow of Events**:

4.1 Member is already logged on

4.2 Member clicks on create a new post and surpasses 200 characters or opts to make it longer

4.3 The member will be given two text fields to customize,

4.3a One field is optional and allows the member to enter up to 200 characters. This will be the teaser for this post and will appear in other users’ feeds

4.3b The other field has no limit and contains the main content of the long post

4.4 Member selects the privacy settings for this post

4.5 The member finishes customizing their post by submitting it

5. **Alternate Flows**

5.1 **If the member submits without creating a custom teaser.**

In this situation, the member opts to not create a teaser for their long post. Website notifies the member that the teaser will be automatically made from the first 200 characters. Members may either accept or change the teaser.

1. If the member accepts, then the system will automatically use first 200 characters of the main content as the teaser
2. If the member declines, then they can create their own teaser

5.2 **If the user cancels the post**

In this situation, the member opts to not create a post at all. Website asks for confirmation to delete the post permanently or store the post as a draft.

1. If member deletes the post, then they will be returned to dashboard
2. If the member saves to drafts , then it will be saved in the database but not viewable by anyone.

**Use Case – Search:**

After a member has already logged into their account, they get the ability to search. The search is one search and will return other members and posts with the query inside of the post.

1. **Description:** The search function allows the member to find other members by name and posts based on content. One search will return both types of result on the same page and the member can interact with the results.

2. **Actors**:

2.1 Member

2.2 System

3. **Preconditions**:

3.1 Member has an active internet connection.

3.2 Member has already logged in

3.3 Member has a query in mind

4. **Primary Flow of Events**:

4.1 Member is already logged on

4.2 Member enters search terms and sends them in

4.3 The search returns the results, and the member can interact with them

5. **Alternate Flows**

5.1 **If the member search returns no results**

In this situation, the member’s search returned no results, so there is nothing for them to interact with.

1. The member is shown a “no results” screen
2. The member can either search again or return to their personal feed.

**Use Case – Send friend request/follow:**

After a member has already logged into their account, they get the ability to send friend requests to other members. Sending a friend request will immediately make the sender a follower of the recipient and allow the follower’s personal feed to contain the recipient’s community posts. If the request is accepted, both members will become friends of the other and give them both the ability to interact with the other member.

1. **Description:** Sending a friend request can be done from either the search screen or from the community post screen and it adds the member who was sent a friend request’s community posts to the sender’s personal feed.

2. **Actors**:

2.1 Member

2.2 Another member

2.3 System

3. **Preconditions**:

3.1 Member has an active internet connection.

3.2 Member has already logged in

3.3 Member has found another member

4. **Primary Flow of Events**:

4.1 Member is already logged on

4.2 Member finds another member they want to become friends with

4.3 Member sends a friend request, temporarily becoming a follower

4.3 The followed user’s posts get added to the follower’s personal feed.

5. **Alternate Flows**

5.1 **If the user is already a friend**

In this situation, the would-be recipient of the friend request is already a friend. If this happens then the system alerts the sender of this, and they can either remove that person as a friend or cancel.

1. The member is alerted by the system to tell them that they are already friends

**Use Case – Like/comment on a post:**

When a post is on a feed the member has the option to like or add a comment. This requires that the member is already logged into their account. When they like the count will go up with an animation and if they comment then their comment will appear under the post.

1. **Description:** This feature will create the feeling of community as it gives people the ability to interact with other’s posts. They will also have the option to react with preselected emojis.

2. **Actors**:

2.1 Member

2.2 Another member’s post

2.3 System

3. **Preconditions**:

3.1 Member has an active internet connection.

3.2 Member has already logged in

3.3 Member found a post they want to like or comment on

4. **Primary Flow of Events**:

4.1 Member is already logged on

4.2 Member finds another member’s post

4.3 Member likes on the post go up by 1 if the member liked it and the comment appears as the newest comment if the member commented.

5. **Alternate Flows**

5.1 **If the member deletes a comment or unlikes the post**

In this situation, the member either deletes their comment or unlikes the post.

1. The member is alerted by the system questioning if they are sure
2. If the member proceeds, then the total likes will go down by 1, or the comment under the post is removed

**Use Case – Chat :**

Users have the ability to chat with other users. This requires that the user is already logged into an existing account and both users are currently online. All chats are saved on the database.

1. **Description:** This feature will create the feeling of community as it gives people the ability to message each other. Chats will take place in real time but for users’ security only friends can message each other.

2. **Actors**:

2.1 Existing user A

2.2 Another existing user B

2.3 System

3. **Preconditions**:

3.1 Both users have an active internet connection.

3.2 Both users are logged in

3.3 User initiating the chat is friends with the user they want to message

4. **Primary Flow of Events**:

4.1 User A is already logged on

4.2 User A is already friends with user B

4.3 User A sends a chatroom request to user B, along with a short message.

4.4 Chatroom request appears on user B’s screen, and user B accepts the request.

4.5 A message box appears on both users’ screens, allowing them to send messages to each other. Sent messages will appear as a stack with newest being closest to the bottom.

5. **Alternate Flows**

5.1 **If the user tries to start a chat with someone who is not their friend**

In this situation, the user attempts to chat with someone who is not on their friend list.

1. The user is alerted by the system with a message that states they need to be friends first.
2. The message also gives them the ability to send a friend request or cancel.

5.2 **If user B declines the chat request**

In this situation, user B declines the request and may send an optional message to user A

1. User B declines user A’s chat request and may optionally enter a message to send to user A explaining why.
2. User A receives a notification that user B declined their chat request, along with any message sent by user B
3. **Initial List of High-level Functional Requirements**
4. **Create Account**
   1. Users should be able to create an account by supplying a Username, Email, Password, FirstName and LastName to the system. This information will be stored and used for future logins and search results. Users will not be able to create an account if any of the Username, Email, Password, Re-enter Password, FirstName, or LastName fields are not filled out. Account creation will be denied if the provided username or email is in use by another account, the Password and Re-enter Password fields have different contents, or the password does not meet minimum requirements of 8 characters, and at least one of each of the following: uppercase letter, lowercase letter, number.
   2. **Response Sequence**
      1. User enters a Username
      2. User enters their Email
      3. User enters a Password
      4. User re-enters their Password
      5. User enters their FirstName and LastName
      6. System checks if the Email and Username are used by another account
      7. System verifies that both Password fields match and meet the minimum requirements
      8. System stores Username, Email, FirstName, and LastName
      9. System will alert the user that the account was created
      10. System will redirect the user to the main page
   3. **Priority: 1**
5. **Create Post**
   1. Users will be able to submit posts to the database when they are logged into an account. Users must provide text content and optional media content that includes images or video. Posts longer than 200 characters will provide the user with an optional Teaser field where they can enter up to 200 characters. The Teaser will appear in other users’ feeds or when searching for posts. Users may also create Teasers at any time by selecting an optional Long Post checkbox. Users may also adjust the privacy level of the post between three settings: **Community** – any user may see this post; this is the default setting. **Friends-Only** – only users that are Friends may see this post. **Best Friends Only** – only users that are Best Friends may see this post. When the user submits the post, it will be saved to the database.
   2. **Response Sequence**
      1. User opts to create a post
      2. User enters text into the Content field
         1. If the Content field exceeds 200 characters, or user selects the optional Long Post checkbox, the user is provided with an additional Teaser field
         2. User enters up to 200 characters in Teaser field
      3. User provides an optional Image or Video with their post
      4. User adjusts the post’s privacy setting
      5. User finalizes and submits their post
      6. System stores the post in the database and notifies the user’s Followers and Friends
   3. **Priority: 1**
6. **Browse Posts and Members**
   1. **Search:** The system will provide users with a search field where they can enter search criteria. The system will return a list of members and posts that match the search criteria. Users will be able to click on one of these results to view it in more detail.
   2. **Response Sequence – Search**
      1. User is logged into an account
      2. User enters search criteria into the search field and sends the criteria to the system
      3. The system returns a list of users and posts matching the search criteria
      4. The system provides a way for users to view a result, search again, or cancel the search.
   3. **Feed:** The system will also provide users with a selection of recent posts from their friends that the user can use to browse content and interact with these posts from.
   4. **Response Sequence – Feed**
      1. User is logged into an account and on their home page.
      2. The system automatically displays a selection of recent posts created by the user’s friends and displays the posts to the user.
      3. The user may interact with the posts by liking, commenting, or expanding the post.
   5. **Priority: 1 – For both Searching and Feeds**
7. **Add Other Users as a Friend**
   1. The system will provide users with a method of sending friend requests to other users, as well as a method of accepting or rejecting any requests that a user has received.
   2. **Response Sequence**
      1. One user (the sender) finds another user (the recipient) and send them a friend request
      2. The system notifies the recipient that they have a friend request and gives them a method of accepting or rejecting the request.
      3. Recipient accepts or rejects the request from the sender
         * 1. If the recipient accepts the request, the system notifies the sender and makes both users friends.
           2. If the recipient rejects the request, the system notifies the sender.
   3. **Priority: 1**
8. **Interact With a Post**
   1. **Comment on a Post:** Each post will have a button members can click on to write a comment in response to the post.
   2. **Response Sequence – Comment**
      1. A member clicks the Comment button on a post
      2. The system provides the member with a text field that allows them to type a comment on the original post
      3. When the member submits this comment, the system links it to the original post, allowing other users to see it.
   3. **Like a Post:** Each post will have a button members can click on to indicate to other users that they enjoyed the post. Members may click this button a second time to remove their like.
   4. **Response Sequence – Like**
      1. A member clicks the Like button on another user’s post
      2. The system increments the count of likes on the post, and notifies the member that they liked the post successfully
      3. If the member clicks on the Like button again in the future, the system decrements the count of likes and notifies the member that they unliked the post successfully.
   5. **Priority: 2 – For both Commenting and Liking**
9. **Chatrooms**
   1. The system will provide a method for two or more users that are friends, and currently online, to create a private chatroom where they can send text messages to the other participants in real time. When all participants leave the chatroom, all messages exchanged will no longer be available to the participants
   2. **Response Sequence**
      1. One member sends a chatroom request to one or more friends
      2. At least one friend accepts the chatroom request
      3. The system places the member and all accepting friends into a private chatroom. The system provides all participants with an interface to share messages, send chatroom invites to additional friends, and leave the chatroom
      4. Participants share messages to the chatroom
      5. The system temporarily stores these messages
      6. When all participants leave the chatroom, it is closed, and the system deletes the messages.
   3. **Priority: 2**
10. **Manage Friends**
    1. The system will provide members with a method of managing the Friendship Level of each of their friends. Members can adjust the level of each friend individually between two levels: **Friends** are able to see the member’s Friends Only posts and community posts. Friends may also send and receive chatroom invites. **Best Friends** can see all posts and send and receive chatroom invites.
    2. **Response Sequence**
       1. A member clicks on a button, opening their list of friends
       2. The system provides a button on each friend that allows the member to manage the Friendship Level of that friend
       3. The member opens clicks on the button, opening a dropdown menu with the Friendship Levels as options
       4. The member selects one of the Friendship Levels
       5. The system updates the Friendship Level and adjusts the friend’s access to posts and chatrooms based on the change.
    3. **Priority: 2**
11. **List of Non-functional Requirements**

* **Performance Requirements**
  + **Expected Load:** The system will be designed to support up to 30 concurrent users.
  + **Response Time:** The maximum response time allowed to the system for transactions will depend on both the number of concurrent users, and the type of transaction. At expected load of 5-10 users, all transactions should take no more than 50 milliseconds. At maximum capacity of 30 concurrent users, creating a new account, logging in, and creating a post should take no more than 500 milliseconds, searching for posts should take no more than 250 milliseconds, both sending and receiving chatroom messages should take 125 milliseconds for a total delay of 250 milliseconds between when a user sends a message, and the other participants receive it. Beyond 30 users, response times are not guaranteed.
  + **Reliability:** The final system will allow no more than 30 minutes mean downtime per month. This will be used for fixing unexpected issues and making updates that cannot be done while the system is functional.
* **Usability Requirements**
  + **Ease of use:** The system should be designed to be intuitive and require a few minutes at most for users to become familiar with its features.
  + **Browser Compatibility:** The system will function on a minimum of Google Chrome and Mozilla Firefox.
* **Privacy & Security Requirements:** 
  + **Privacy:** The system will comply with all privacy regulations in the locations it operates in.
  + **Login System:** Accounts will require an email and password to create and log in to. On account creation, users will be required to re-enter their password for confirmation. Accounts may also be secured with two-factor authentication that links to the user’s email.
  + **System Access:** All members of the development team will be able to make changes to the database, front-end code, and back-end code.
* **Storage** 
  + The system will be stored on our LAMP server on lamp.cse.fau.edu. Both the database and site files will be stored here.
  + Backups will be stored on our project’s Github repository to avoid data loss in case of an issue with the LAMP server.
  + The system should not use more than 85% of the available space to leave room for bug fixes.
* **Availability Requirements**
  + The system should be available whenever the LAMP server is available, minus any additional downtime. Downtime will be limited to an average of 30 minutes per month.
* **Testing**
  + **Exception Handling:** The system will have exception handling that will display explanations of the exception to the user. Users will be able to either try another input or redirected away from the page that had the exception.
  + **Functional Requirements:** Tests will be devised and conducted for all functional and non-functional requirements.

1. **High-level System Architecture and Database Organization**
   1. **High level Architecture**
      1. LAMP Server - <https://lamp.cse.fau.edu/~cen4010_fa21_g10/index.html> The FAU lamp server will host both our team site and our project site.
      2. Hyper Text Mark-up Language - will be used to display the application in the web browser.
      3. Cascading Style Sheets - will be used to style the HTML.
      4. JavaScript - will be the language used for client-side functionality.
      5. Bootstrap - will be used for additional styling web page structure
      6. PHP – Will be used for server-side programming
      7. MySQL – will be used for the database to maintain user posts/images/user accounts.
      8. Visual Studio Code - will serve as the primary code editor.
      9. Notepad++ - will be used to hold code snippets or command line commands.
      10. Photoshop – will be used to edit or create images as needed for the webpage
      11. GitHub – will be used to maintain code and version control
      12. Jira- will be used to track development as the project progresses.
      13. Whatsapp – will serve as the primary communication tool for the team.
      14. Webex – will be used to host meeting.
      15. Web browsers – various web browsers including Google Chrome, Mozilla Firefox, and Microsoft Edge, will be used throughout the development of the project.
   2. **Database Organization**
      1. **Relay\_users table**
         1. Id: Automatically assigned unique identifier
         2. userName: String containing the user’s username
         3. passd: String containing a representation of the user’s password
         4. firstName: String containing the user’s first name
         5. lastName: String containing the user’s last name
         6. email: String containing the user’s email
         7. bio: String containing the user’s bio
      2. **Posts table**
         1. id: Automatically assigned unique identifier
         2. usrName: String containing the post author’s username
         3. title: String containing the post’s title
         4. teaser: String containing the post’s teaser
         5. content: String containing the post’s text content
         6. postLevel: Int representing the post’s privacy setting
         7. likes: Int containing a count of the number of times the post was liked
         8. postDate: Date containing the date the post was created
      3. **Relationship\_table**
         1. Id: Automatically assigned unique identifier
         2. recipient\_name: String containing the username of the user who received the friend request
         3. requestor\_name: String containing the username of the user who sent the original friend request
         4. request\_status: int that represents the status of the friend request
         5. rec\_to\_req\_relationship\_level: int representing the receiver’s friendship level setting for the requestor
         6. req\_to\_rec\_relationship\_level: int representing the requestor’s friendship level setting for the receiver
      4. **Chatrooms table**
         1. room\_id: Automatically assigned unique identifier
         2. status: int representing the chatroom’s state
      5. **Chatroom\_messages table**
         1. Id: Automatically assigned unique identifier
         2. room\_id: identifies the chatroom this message was sent in
         3. userId: String containing the user’s unique id
         4. content: String containing the message’s text content
         5. date: Date containing the date the message was sent
      6. **Chatroom\_participants table**
         1. Id: Automatically assigned unique identifier
         2. room\_id: identifies the chatroom the user was in
         3. userId: String containing the user’s unique id
         4. status: int representing the user’s state in the chatroom
   3. **Media Storage** – Media files will be kept in the LAMP server’s file system. Users will be able to upload .jpg, .png, .mp4, and .mov files that can be displayed in their posts.
   4. **Search and Filter –** Users will be able to search for posts and other users with a simple search algorithm that looks for the user’s search terms in usernames, firstNames, lastNames, post titles, teasers, and post content. The returned posts can then be filtered in ascending or descending order by postDate. Returned users can be sorted in ascending or descending order by username, firstName, or lastName
   5. **APIs –** We are not planning to implement any APIs at this time.
   6. **Significant non-trivial algorithms** – Populating user’s personal feed with relevant content will be done by assigning each post with weights that favor recency, higher friendship levels, and like count. There will be some randomness involved, but posts with higher weight will be more likely to be displayed earlier in a feed.
2. **High-Level UML Diagrams**
   1. **Class Diagram**

**Diagram

Description automatically generated**

* 1. **Use Case Diagrams**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

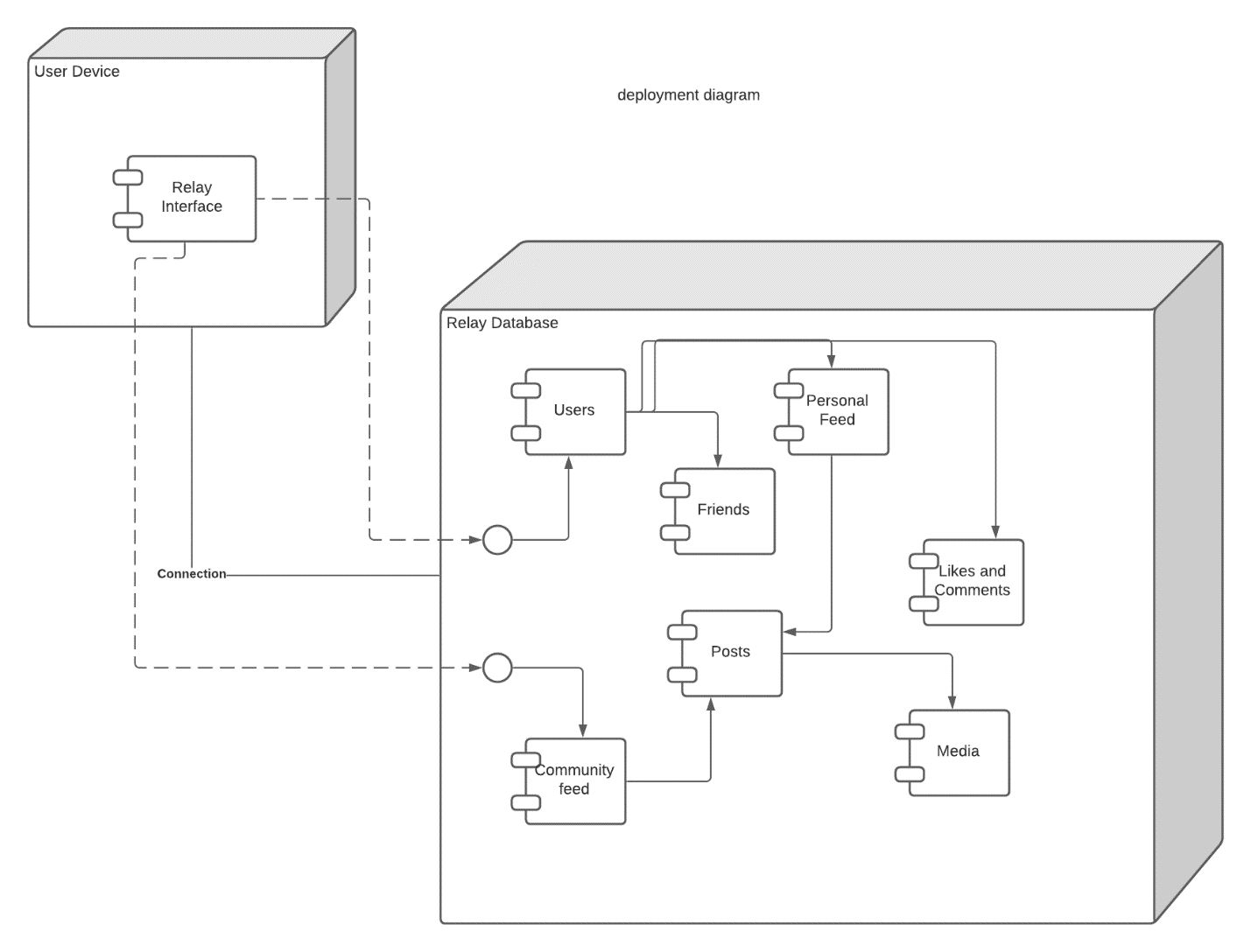
Description automatically generated**

* 1. **Component Diagram**

**Diagram, schematic

Description automatically generated**

* 1. **Deployment diagram**

****

1. **Key Risks**
   1. **Skills Risks –** At this time, our team believes we have the necessary skills for this project.
   2. **Schedule Risks –** Currently we feel that our project has been making good progress, and we believe we can achieve our goals during this semester.
   3. **Technical Risks –** One feature that may prove difficult to implement is the Live Chatrooms. This feature may be a challenge because we are unsure how to guarantee the real-time communication required by the feature at this time. We plan to deal with this issue by selecting an implementation method and have already begun researching possible implementations.
   4. **Teamwork Risks –** Our team has been working together and communicating well through our group chat and meetings. We do not foresee any issues at this time.
   5. **Legal/Content Risks –** We do not anticipate any issues with obtaining licenses and copyrights for the software we use during development. However, there may be issues with users uploading copyrighted media to our site. We should be able to address this issue through moderation and by making users agree to a terms of service where they promise to only upload media that they have all intellectual property rights to.