Final Report

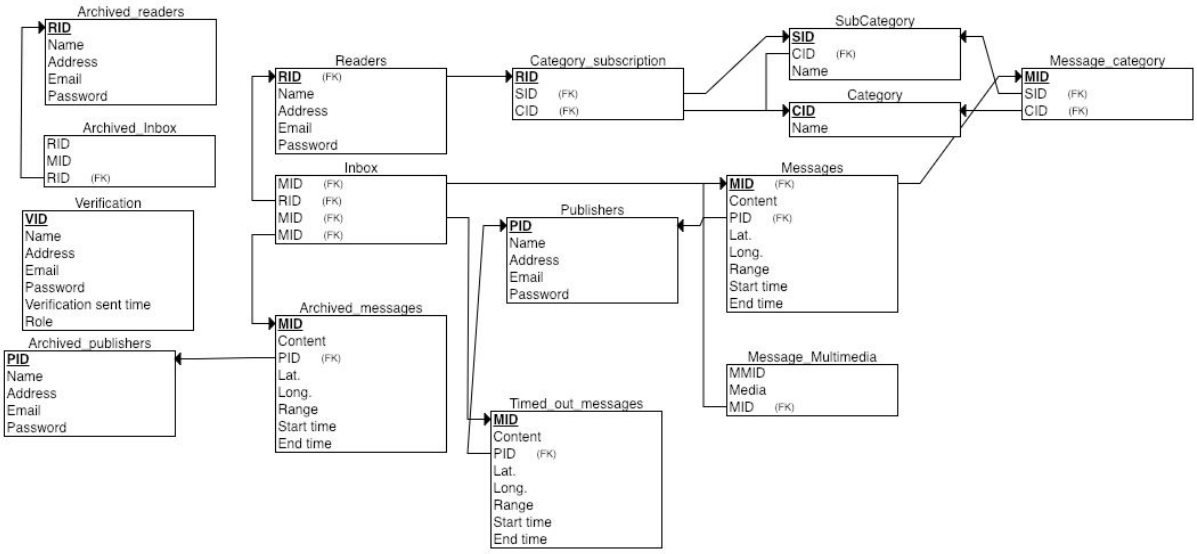
Lucas Ference
Alisha Asnani
Yanran Qian

December 11, 2018

Problems Encountered:

- Categories were a serious issue during this phase. When a publisher creates a message, they can choose any amount of categories/subcategories to be associated with it. We were unsure what the best way to handle this was, and also handle the scenario where a publisher would want to create new categories. We did not want the publisher creating a message with categories that were not already existing in the database. Because of this, we separated the functionality of creating categories/subcategories and actually selecting them for the message. This ensures that all categories selected for the message are already in the database, and it still allows a publisher to create new categories on the fly.
- Another issue we encountered was the ability to let the application connect to the web-server, even when they were located on different networks. Our solution to this was to use a software called ngrok, which creates a public tunnel to a localhost. This allows anyone to connect to the server from anywhere by accessing a URL provided by ngrok.
- Timing actions on the server was a serious consideration especially when handling message timeouts and verification expirations. We found a package called “node-schedule”, which allows the scheduling of functions to execute at a certain time. This made our lives much easier by scheduling a message removal once the expiration time had been reached. This was also useful for removing old/invalid verifications from the table, saving a large amount of space in the process.
- Image storage was an issue and sending an image to the server was rather problematic in terms of how much data our server could take in at once. Our solution was to convert the image to base64 before sending to the server and storing as a string in the database. When it is sent back to the application, it will be sent as a string and converted locally on the app back into an image before being displayed.

Updated Relational Model:



Some quick notes about schema update:

- The inbox table represents all messages received by a user. This table is there to ensure a user does not receive the same message twice and to track their interactions. When the reader is archived, their profile and interactions are archived, hence the archived inbox table and archived readers table.
- The message multimedia table is meant to associate a message with all of it's multimedia. It can always be accessed, even if a message is archived or timed out. It holds a messages ID and a base64 encoding of a sound or image file.

Assumptions:

- When a user queries for archived messages, we assume they are actually querying for archived messages AND timed out message (since they are stored separately in our database).
- No two categories can have the same name.
- No two subcategories under the same category can have the same name.
- No two users can have the same name (since name is treated as a username).
- If a message is associated with the whole category, any reader subscribed to any subcategories of that category will receive the message.

- If a reader/publisher selects any subcategories under the main category, even if the category was selected, we assume that they DO NOT want to be subscribed to the whole category and rather just the subcategories selected.
- If a user selects the category, but none of the subcategories, we assume they want to be subscribed to the entire category.
- Every time a user changes their subscriptions, the most recently selected subscriptions are the readers subscriptions, and any previous subscriptions are replaced.
- If a reader selects no subscriptions, they will receive no messages.
- A message must be associated with some categories (i.e. cannot create a message with no categories).
- When selecting a location to query for messages, we will assume they will click a location, if not they will be given stories in the default location of Ghana (latitude: 0 , longitude: 0).
- A publisher will always enter a messages range in kilometers.
- When a publisher creates a message, they must fill out the required fields (every field is required except image) or the message will be denied from the server.
- The readers inbox will only have the currently active messages for the categories and subcategories that the reader is currently subscribed to.

Users Manual:

Upon opening up the application, the user will be prompted to sign up or log in.

Signing Up:

When a user signs up, they will type in their name, address, email and password. They will also select if they are registering as a reader or a publisher. Afterwards, they will be brought to a “Congrats” page while an email will be sent to the email address they entered in the previous page. The reader will have to click a verification link provided in the email. If they do not do so within 30 minutes, the link will expire and they will not be registered. On the other hand, if they click the link in time, they will be registered in the system!

Logging In:

After the user clicks “continue”, or after if they select “login” on the home page, a login page will show up. The user will be prompted to enter their username and password.

Publisher:

If the user is registered as a publisher, they will be led to a page with all of the active messages’ titles that they have published. However, the messages won’t show up unless they click the button “show all my messages”. When they click on a title, they will be able to view the content of the specific message.

Querying Messages:

By clicking on the “search for messages” button, the publisher will be able to query messages. They select location from map by either triple tapping or searching in the search bar. Then, they can select certain categories and subcategories, and/or specify the time interval. Finally, they can select if they want to query for active or archived messages. Messages will now show up on the table. When they click on the message, the message contents will show up.

Publishing New Messages:

When the publisher clicks “publish new message”, a map will show up, where the publisher can select the location of the center of the message coverage. Then they press set_loc button and a page will show up where they can enter the message’s subject, content, the range in kilometers, and pick start time and end times. The “next” button will save the information and let the publisher select which category/categories and subcategory/subcategories they want the message to be under. The selected categories and subcategories will appear green while the unselected will remain blue. At this point the can choose to Add a new Category or subcategory if they click the Add New button. After they’ve selected all the categories and subcategories, they can press “publish” and the message will be published! Note that if they don’t select any categories or subcategories, they will not be able to publish.

Adding New Categories:

On the select category page, there will be an “add new” button. This will lead to a page where they can add new categories or subcategories. Note that when they add a new category under a category, they have to ensure that the category does not already exists or it will not be added and the request will be ignored. A label will show up saying Category already exists.

Note that when they add a new subcategory under a category, they have to ensure that the category already exists or it will not be added and the request will be ignored.

Reader:

When a user logs in as a reader, their home page will be all the messages in their inbox. That is, all active messages under the categories they are subscribed to and valid for the reader's current location. Again, they won't be able to see those messages unless they click "show my messages" button. And again, when they click on a message title, they will be able to view the content of the specific message.

Add Subscription:

There is an "add subscription" button for readers. When they click on that, they will be lead to the list of categories and subcategories. Selected categories and subcategories will appear green. After they've decided on their interest, they can click "subscribe" and their preferences will be saved to our database! They will have to select all the categories that they want to be subscribed to. Including those they were already subscribed to. If the reader does not pick a category or subcategory that they previously subscribed to, they will be unsubscribe to that category or subcategory.
Note: if the reader selects a category and one or more of its subcategories, they will only be subscribed to the subcategories.

Requesting Messages From a Different Location:

After a reader has clicked "receive messages from a different location", a map view will show up. The reader can search for a location, and a pin of the searched location will show up on the map. When the user clicks "confirm location", a list of active messages in that location will appear. And again, when they click on a message title, they will be able to view the content of the specific message. These queried messages will not be added to the users inbox!

Querying Messages:

By clicking on the "find messages" button on the upper left corner, the reader will be able to query messages. They can select certain categories and subcategories, and/or specify the time interval. Finally, they can select if they want to query for active or archived messages. A list of messages in that location will appear. And again, when they click on a message title, they will be able to view the content of the specific message.

Deleting Account:

On the home page for both readers and publishers, there's a button called "delete account". By clicking that, a pop up window will ask the user to confirm if they want to delete the account. If they choose "yes", the account will no longer be able to login. Note: at this point the account will be moved from the readers/publishers table to the archived_readers/archived_publishers table.

Room For Improvement:

- If a user has the app closed and running in the background, let them be notified when new messages are added to their inbox. This functionality is currently not implemented.
- Implement logout functionality.
- Allow publishing and receiving messages with more multimedia such as audio files and videos.
- Add legitimate authorization for user via OAuth.

Testing Efforts:

- A user cannot have the same name as another user. Ensures that there is not already a user in the process of being verified with the same username.
- Adding a large amount of messages will not cause a timeout from scheduling their removal.
- User that is not verified cannot login.
- If a user unsubscribes from everything, their messages will be empty as they are no longer subscribed to those messages.
- If you attempt to verify after 30 minutes, you will be unrecognized and denied. The data will also be removed from the verification table to save space.

Phrase 1 Report:

1. A description of the scope of the project and the assumptions that we made
    a. We assumed that this project is to create a basic SNS server which is backed by a mySQL database. The server will connect the database to the application by sending messages from the database. The web server will check the which users fit the requirements of the message (category and location) and push it out as notifications to those users.
    b. It order to do this, we assumed that the project has three parts, an mobile application, a web server, and a SQL database.
    c. Most of the interaction with the user will be at the application level like registering, picking categories and sending out messages.
    d. SQL database will storage all the messages, interactions, and user information.
    e. And the web server will connect the application and SQL database by performing the necessary queries, sending out the messages, and storing message and user data.
2. A description of the technical/conceptual issues encountered in this phase and justification for the solutions adopted
    a. The biggest problem we have faced technical is not having the same hardware. Since not all of us used the same OS and phones, we had to discuss which software we were going to make (iphone vs android, and for which version of iphone) and how to break it up.
    b. Also, technically, no one in our group had really make anything like any parts of this project, so we all had to learn how to do their parts from scratch. But, I assume that is expected that we learn it.
    c. Conceptual issues we faced was just trying to figure out what would be the best places to have specific interacts. We didn't know whether we should make the web server more interactive and have the publishers in put messages from there.
3. the list of tasks and subtasks in your implementation. These should include the ones specified above appropriately modified to fit your implementation.

App:

- Publishers:
    - Publish
        - name of the publisher
        - time of entering
        - category or categories
        - time duration (start and end times)
        - specify the center of the location
        - the extend (i.e. range) that the message will be forwarded.

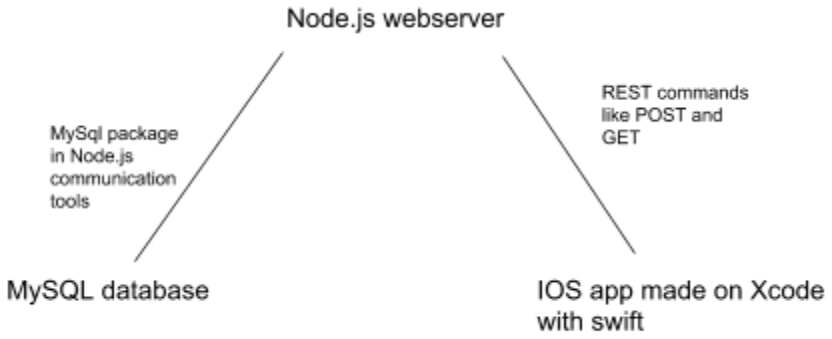
                - (user) save all the GPS locations of the readers so that it can be used to calculate number of users in extent
        - Check credentials of the publisher to make sure that he/she is registered and verified
        - Display and Enter New Categories
            - Display is same as readers
            - Have options at all end of list of category and subcategory to add new
        - Have a map showing the number of people got sent the message and their location.
- Reader:
    - Receive notification
        - Have the application accept incoming information from the web server.
            - If it is a message (tell using meta-data), send it up as a notification
            - If it is other info, new categories, do appropriate changes on app.
    - Send phone location automatically
        - Re-sent location every few minutes
        - Send command to search for the active messages in location
            - (node.js) search all of active messages and forward the ones that are within the range of user's location.
        - If the users moves into new location, ask if user wants to receive this location-specific notifications.
    - allow having location sent by the reader (on map)
        - Look for messages in that specific area and forward it to application
    - Let readers query messages within a time-period or in a specific category.
        - A page-just for inputting the time-period/category and returning all results.
    - Display Categories
        - Categories will have subcategories.
        - System will only take the currently clicked categories
            - User can click the whole category or click on arrow to expand and click the subcategories
            - User can unclick to stop receiving
        - Subscribe to the Categories
- Register readers and publishers
    - 2-phase verification
        - Reader/publisher puts in information(name, address, e-mail, password,)
            - Information is stored in database with unverified
            - The time asked for verification is stored
        - Reader/publisher is sent a URL to his email with verification
        - When link is clicked and sys.time - timeAskedVerification < 30 minutes, the system changes person to verified.
- De-register
    - user/publisher is removed from active table
    - The system moves user/publisher's information to a different de-registered table.

- All data associated with the deregistered user will be moved into a de-registered table as well.

Web Server/Database (exclusively)
**there are most web/database stuff under app because app triggers it

- Archival of Inactive Messages
  - Move messages that are no longer active into a new table. Done at the end of a messages life (need to pick time)
- Time-interval and Content Query
  - Let people on app be able to query for messages within a time-interval and/or category.
    - If time interval is not considered active, the query should only be on archived table and vice-versa.

4. implementation and demo
   a. For our architectural design of the system, we have three parts. We have a node.js web server connected to connected to a mySQL database and a iphone application written in swift on Xcode.



   b. The map API we used and its implementation. Demo of it by:
      i. specifying a location, say Baltimore, and display it on the map, and
      ii. specifying a point on the map obtain the GPS coordinates of the point. This can be done on a laptop or a mobile device.
      iii. Describe the implementation of the web server and demo its access.
         1. Show POST requests working, and information posted is relayed.
         2. Show basic GET requests to the server (“Hello World”).

Phrase 2 Report:

Corrections from TA:

The TA did not recommend anything for our next part. He said he was pleased with our current implementation and was impressed with the plans we have made for the future.

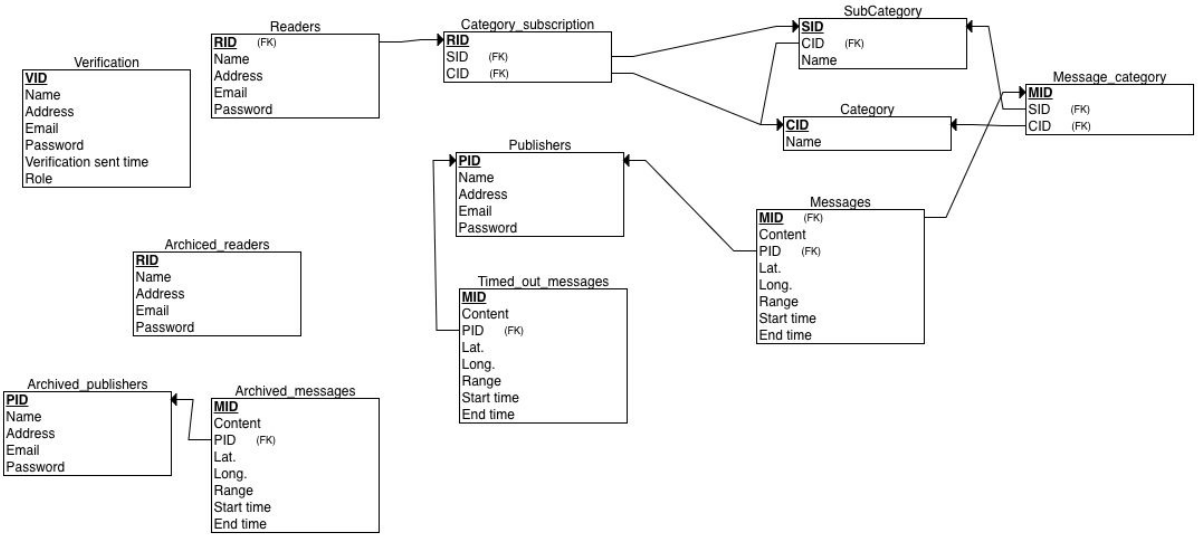
Problems encountered:

1) We encountered an issue when organizing categories and subcategories. A user can subscribe to many subcategories without subscribing to the main category. We decided the best solution was to associate a reader with many subcategories and process based on that rather than categories. A default value will be used for the subcategory in the CATEGORY_SUBSCRIPTION table to identify a situation where a user is subscribed to the entire category, rather than just certain subcategories.

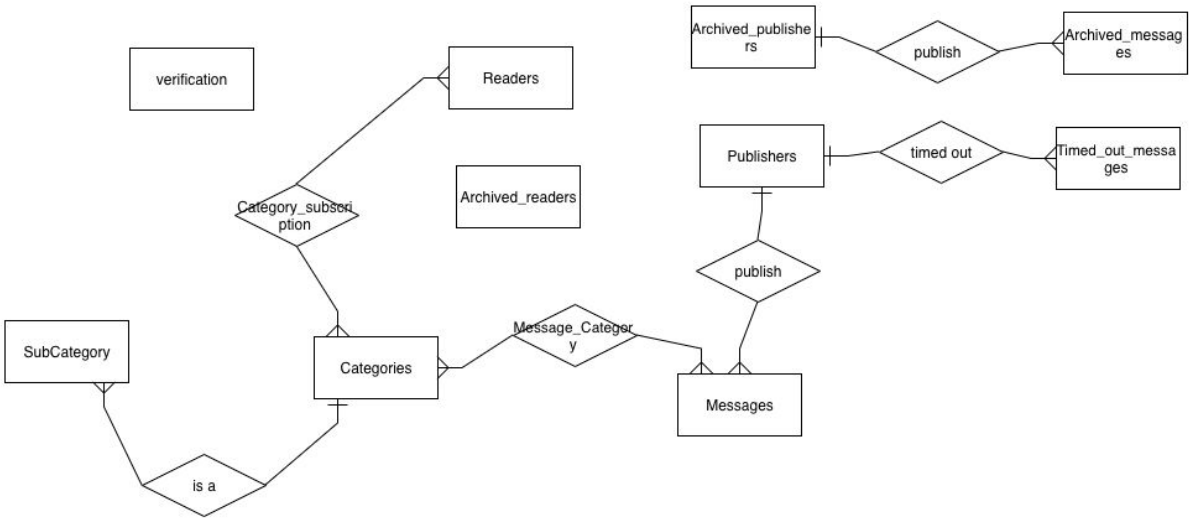
2) Storing coordinates also causes some issues. We are mostly concerned with space and if it is possible to store latitude and longitude in the same attribute, rather than two. This is mostly a concern to try to conserve space.

3) We chose to separate inactive messages into two tables. We divided them into TIMED_OUT_MESSAGES and ARCHIVED_MESSAGES. Current publishers that have messages that are just timed-out will find their old messages in TIMED_OUT_MESSAGES. When a publisher is actually deleted, all of their current messages and timed-out messages will be moved into ARCHIVED_MESSAGES. We feel that this is important because finding a publisher of an archived message would involve figuring out if they were active or not, then searching that table. This is rather slow, and we decided taking up more space would be valuable.

Relational Scheme:


Verification
VID
Name
Address
Email
Password
Verification sent time
Role
Readers
RID (FK)
Name
Address
Email
Password
Category_subscription
RID
SID (FK)
CID (FK)
SubCategory
SID
CID (FK)
Name
Category
CID
Name
Message_category
MID
SID (FK)
CID (FK)
Publishers
PID
Name
Address
Email
Password
Messages
MID (FK)
Content
PID (FK)
Lat.
Long.
Range
Start time
End time
Archiced_readers
RID
Name
Address
Email
Password
Timed_out_messages
MID
Content
PID (FK)
Lat.
Long.
Range
Start time
End time
Archived_publishers
PID
Name
Address
Email
Password
Archived_messages
MID
Content
PID (FK)
Lat.
Long.
Range
Start time
End time

ER Scheme:


verification
Readers
Archived_publishers
publish
Archived_messages
Publishers
timed out
Timed_out_messages
Archived_readers
Category_subscription
publish
SubCategory
Categories
Message_Category
Messages
is a