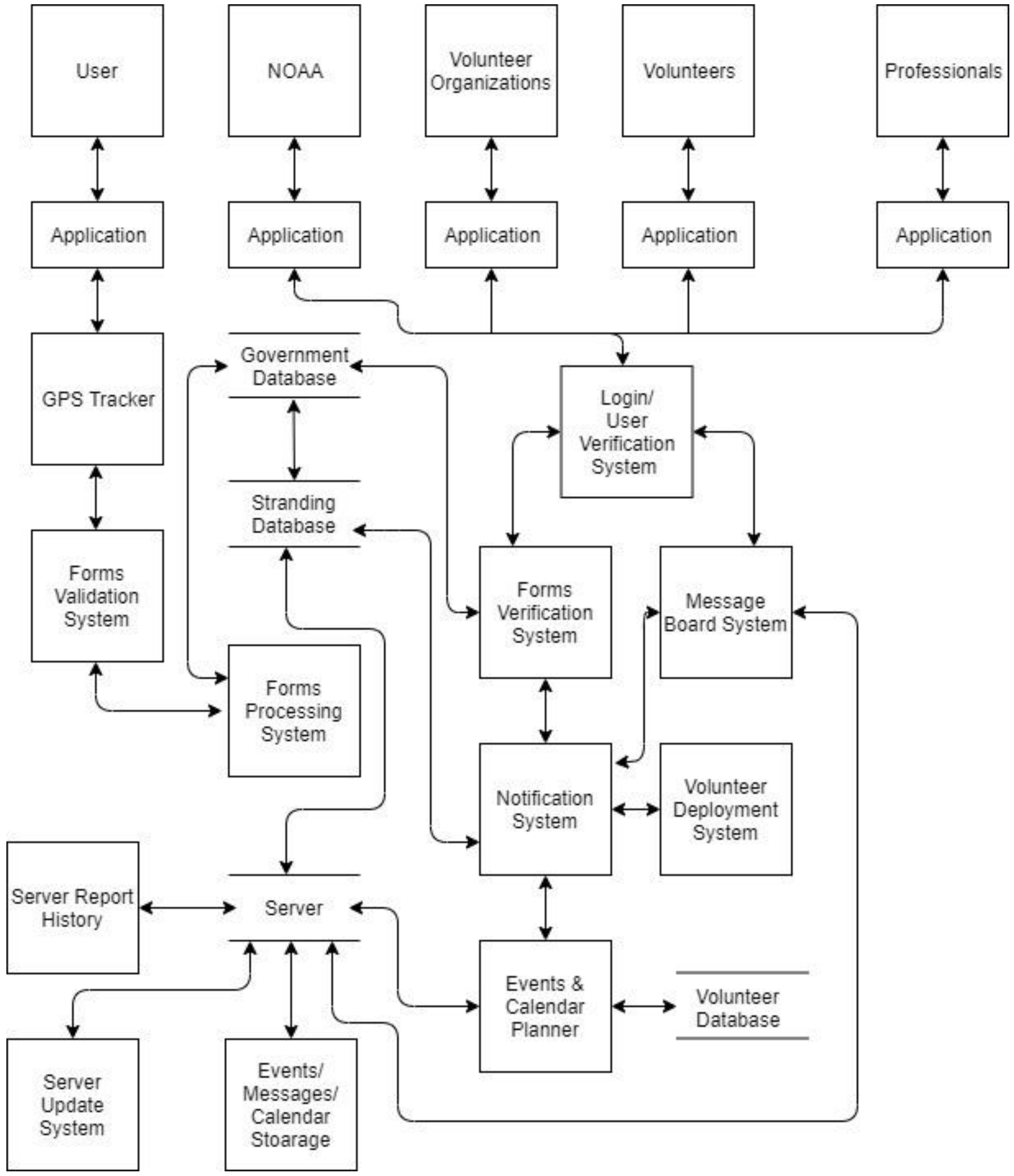
MARINE MAMMAL STRANDING RESPONSE & DATA TRACKING – ARCHITECTURE

NOVEMBER 3, 2019

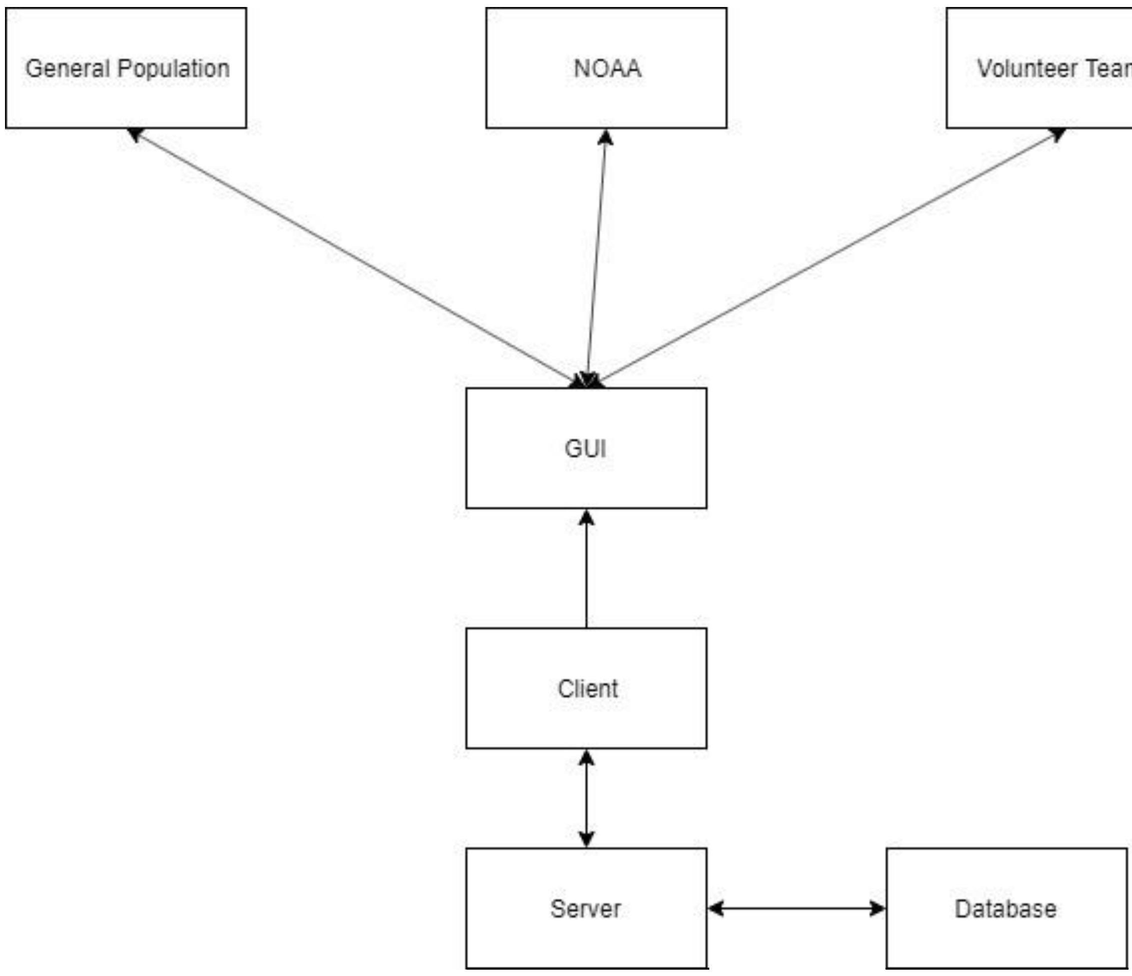
Group 27
Brittany Abad
Lauren Boone
Christopher Feth
Manda Phadke
Kunal Patadia

SYSTEM ARCHITECTURES

Architecture 1 – Publish-Subscribe Architecture



Architecture 2 – Client-Server Architecture



KEY QUALITY ATTRIBUTES

- Reliability
    - **Architecture 1**
        - The publish-subscribe system allows the users to interact with the other users and access data within a centralized system. The advantages of this is since many clients using this system operates by connecting to a network, the users will be able to store and retrieve data from the server. Additionally, verified users will be able to subscribe to events that are posted by NOAA to take part in it as well as being able to communicate with each other. If there is some internet connection, whether through wifi or a cellular network, users will be able to access these components in the application.
        - Because entering phone numbers on the forms would be required as a means of communication, and as a backup, text messages can be sent through the user's devices in case if the server went down. This will allow verified users to be able still communicate since the events were sent to their emails, where they can access that data. However, there may be some details left out since users also post on the message board, which is accessible through the application, and if the server is down there will not be way accessing this.
    - **Architecture 2**
        - As long as there is a strong internet connection and the server is up the system should be highly reliable. If there is no internet connection or the server is down for maintenance then the system loses reliability.

- Efficiency
    - **Architecture 1**
        - The publish-subscribe system would be able to scale all workloads being done with the client. Accessing key information from the databases will allow the data being sent from one official user to organizations and volunteers with speed and systematically. The volunteers are able to subscribe to an event set by NOAA, allowing them to receive notifications and communicating with co-volunteers and officials through the message board. In some cases, there can be disruption caused if there isn't strong internet connection available, but a backup process integrated with the system allows certain features such as text messages to be sent rather than a standard message from the message board.

- **Architecture 2**
  - The client-server model would be very efficient in regards to setting up the stranding website and accessing the database. The message board, website, and application will be updated in real time by the server allowing for fast communication. However, the system may lose efficiency from the need to be connected to strong internet.
  - This model is efficient for data sharing to authorized users. This means that someone using a mobile phone at a stranding event can pull the correct forms very easily.

- Integrity
  - **Architecture 1**
    - The verification system checks to see if the forms have been previously validated to remove discrepancies on reports that have been already submitted, reducing multiples submissions of the same event. The login requires verification from the user end, mainly the officials part of NOAA and volunteers, who use their organization's email address to verify their credentials.
  - **Architecture 2**
    - There is very little confidential data that is kept within the system and therefore integrity is not a large issue. However, the client-server architecture provides security by storing the data on the server allowing for better control over access.

- Usability
  - **Architecture 1**
    - The publish-subscribe model would require internet to send any updates or notifications to volunteers and first responders if any events have been updated or changed. This can potentially impede the work between the different groups, which would make it challenging to reorganize the event. However because this is server reliant users can switch to different clients with a better connection to access this.
  - **Architecture 2**
    - This client-server model requires internet that may not be available in the strandings locations. This could be a large detriment to usability. However, all the data will be stored in the server allowing for an approved user to have access to all the information they need via a client.

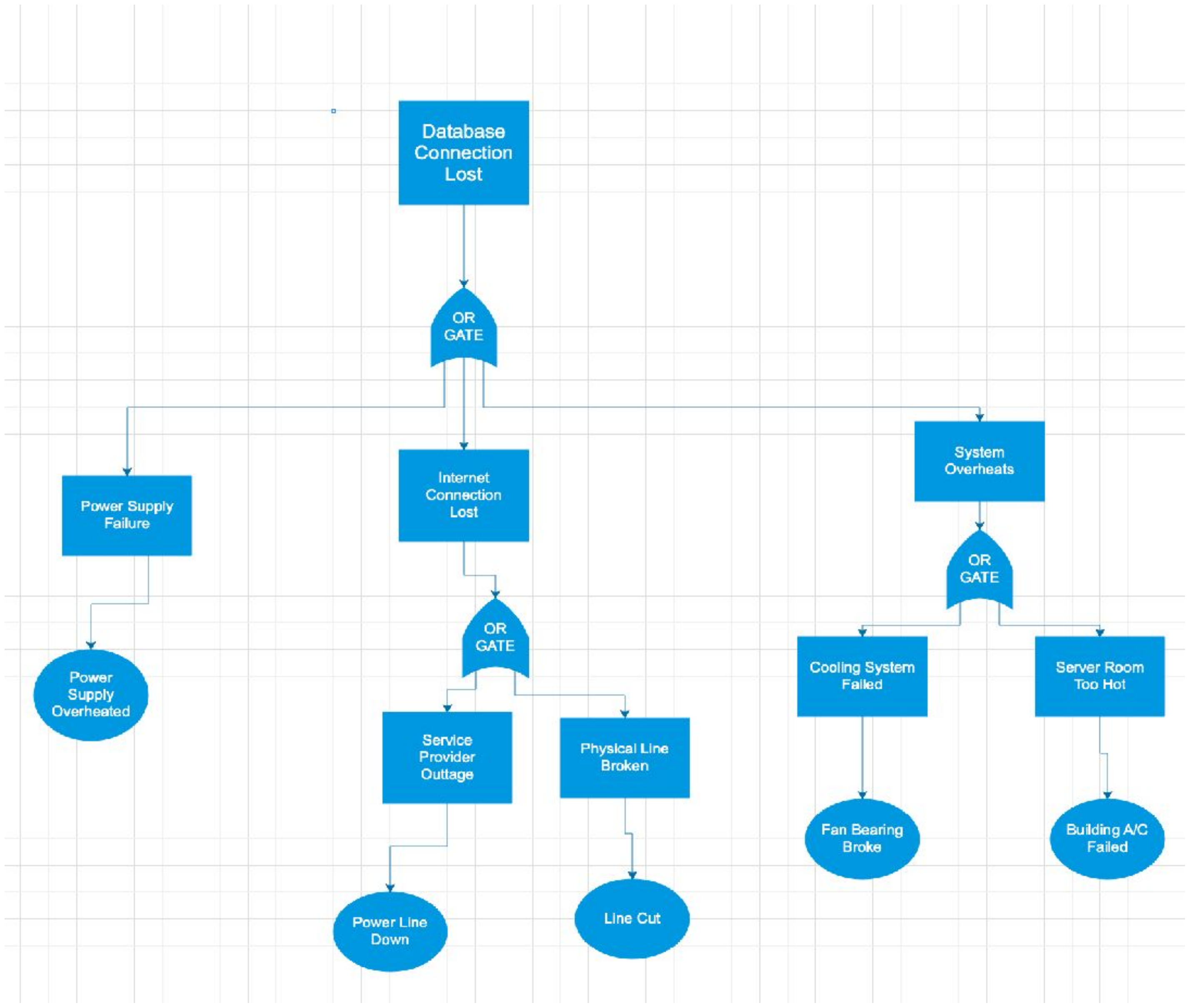
- Maintainability
  - **Architecture 1**
    - Any changes or updates made to the system would be from the server, which is beneficial since it doesn’t rely on multiple types of clients. However, the clients might experience some disruption. This would require some down time for the server for any maintenance issue which can affect retrieving some data such as sending notifications to organizations and volunteers. Since this system doesn’t depend solely on the client, multiples clients using the system might experience this disruption.
  - **Architecture 2**
    - Any updates would need to take place on the client and the server. This may require the server to go down during maintenance. If maintenance was occurring during a stranding event this would affect NOAA and the volunteer groups ability to attend to a stranding event.

- Flexibility
  - **Architecture 1**
    - Because the server stores a lot of the reports, messages, and events, it allows the users to access and communicate using a centralized system. As mentioned, if there was some issue or system maintenance required, clients would be disturbed by any processes required to obtain such data for the user.
  - **Architecture 2**
    - The system should be able to adapt to unusual conditions however the system would go down if changes or updates need to be made to comply with unusual conditions.

FAILURE MODES

Database Connection Lost

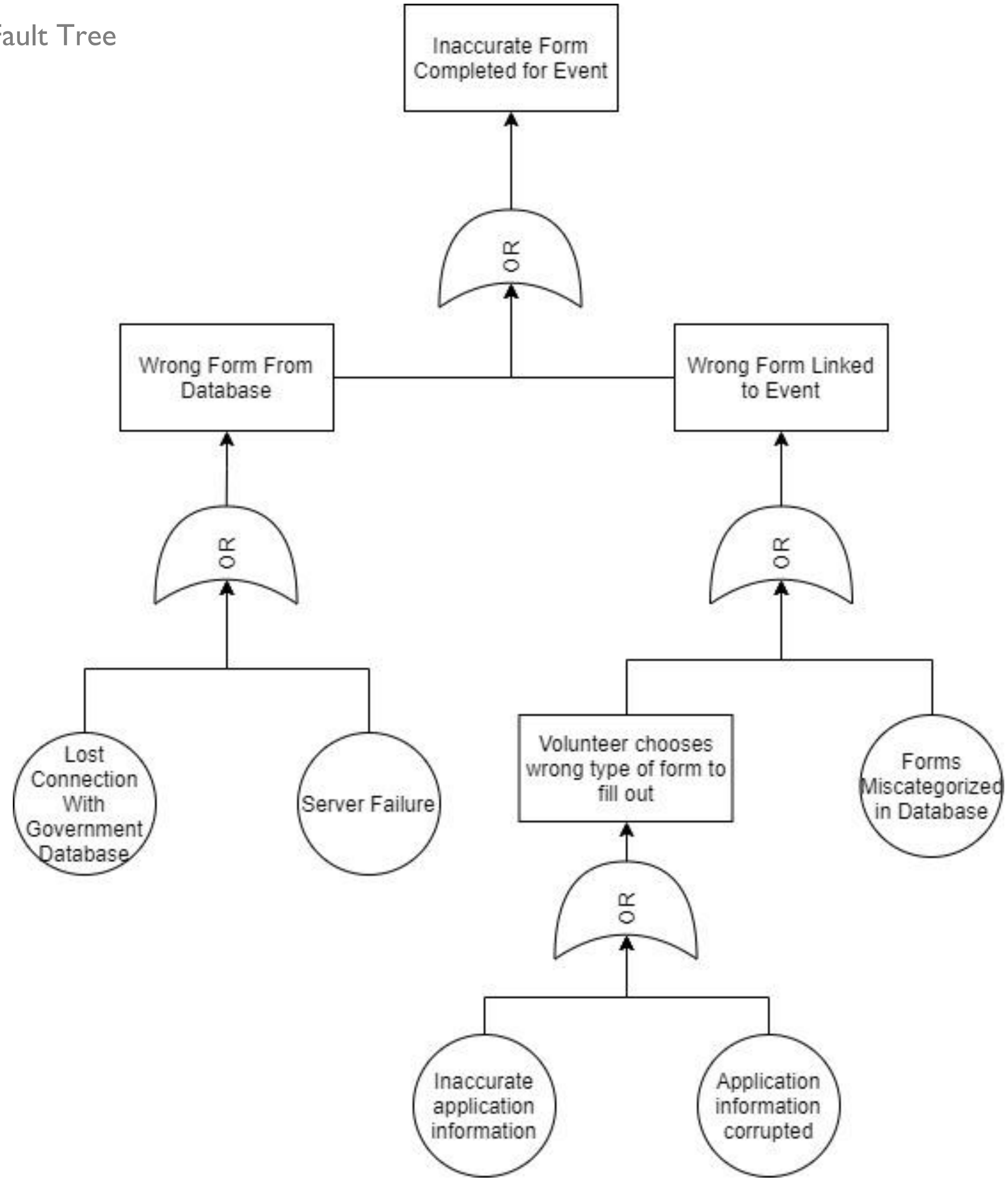
Fault Tree



The architecture more affected by the database connection lost failure mode is architecture 1 — where most of the application's functionality is dependent upon 3 database connections. In fact, the functionality that the general public user will have is entirely dependent upon databases and therefore, in the event of a database failure, the reporting system would completely fail. The Volunteers/Professionals would still be able to view active events, but the events themselves may not be current because of the user functionality failing. With architecture 2, however, the database is only a portion of the critical systems communicated with to deliver functionality alongside the server, website, application, and client. Architecture 2 is only dependent on the database as a means of storing information and not means to display it.

Inaccurate Form Completed for Event

Fault Tree

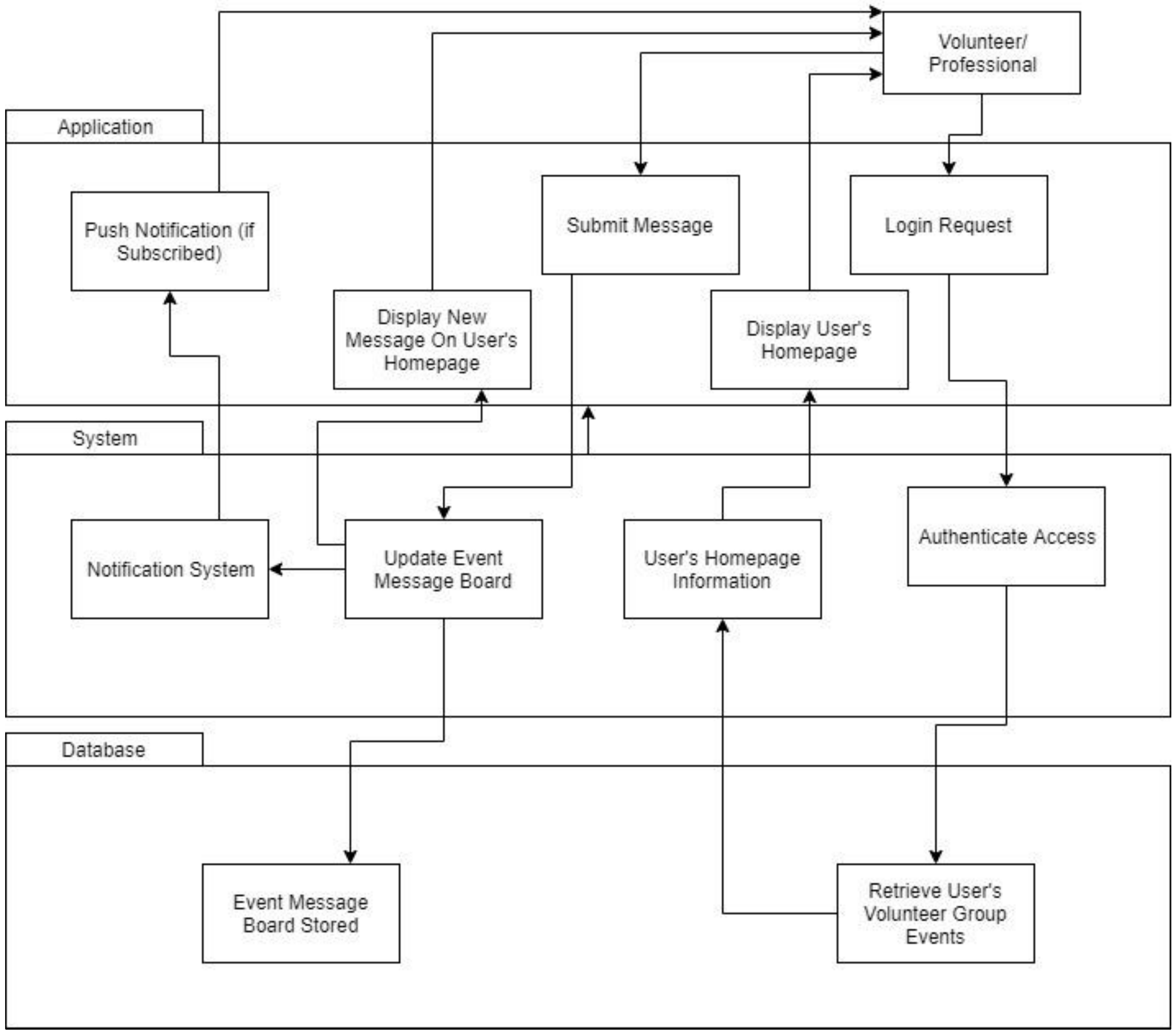

Inaccurate Form Completed for Event
OR
Wrong Form From Database
Wrong Form Linked to Event
OR
OR
Lost Connection With Government Database
Server Failure
Volunteer chooses wrong type of form to fill out
Forms Miscategorized in Database
OR
Inaccurate application information
Application information corrupted

The failure mode outlined by this chart is influenced by a server failure, loss of connection with the government database, corruption of data from the application or a mismanaged database of current forms. In the client-server architecture, the server failure, loss of connection with the government database and mismanaged database of the current forms all happen with the server in one line so if one failure happens it is likely to ripple across the system. In the case of the publish-subscribe model, the government database is only pulled from once - when the event is processed by the volunteer - so if it gets corrupted or the connection is lost, then it doesn't impact the event page that the Volunteer/Professional is interacting with since the connection with the government database is complete.

ARCHITECTURE DECOMPOSITION

Important Element 1

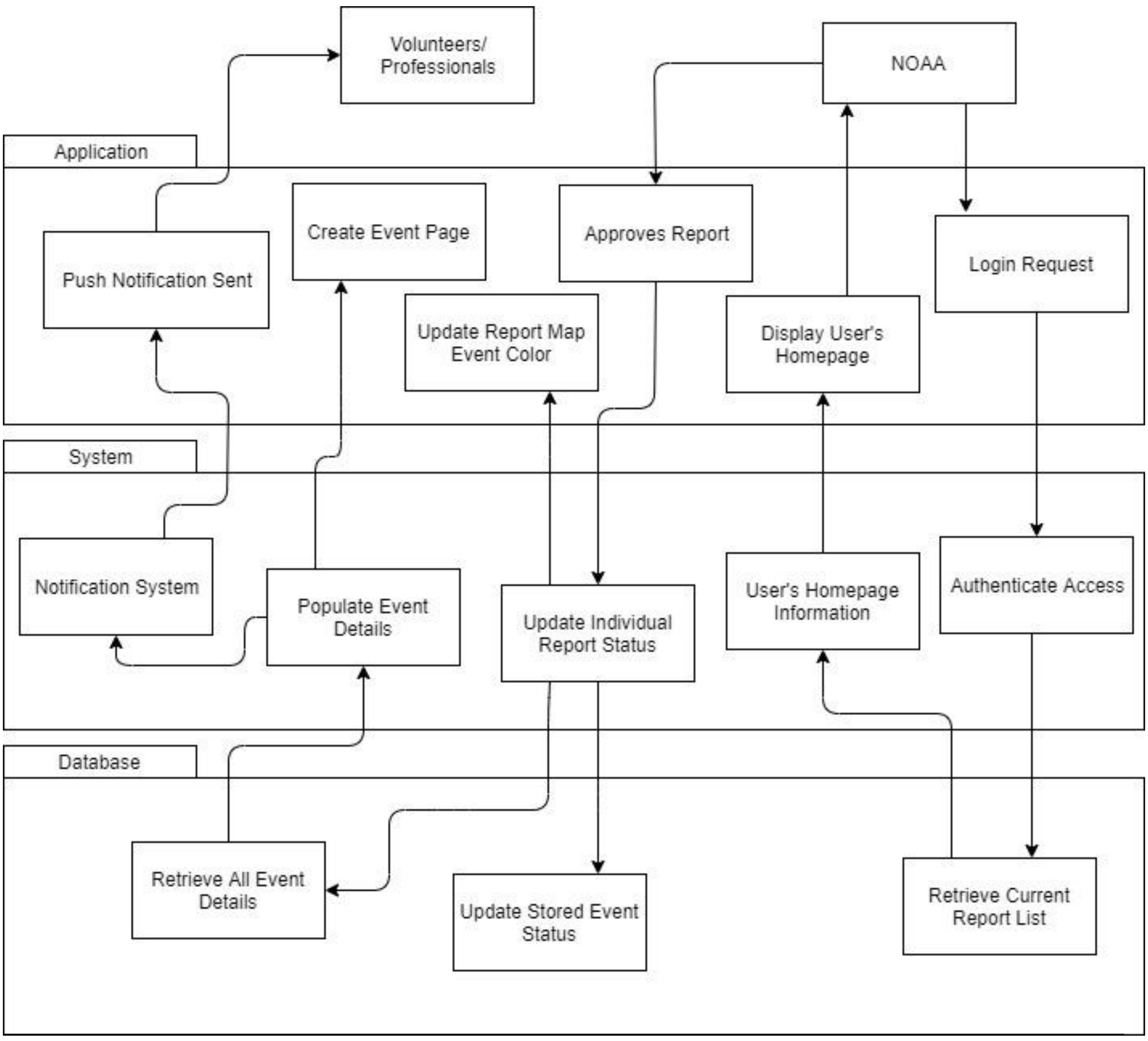
Message Board Lower-Level Dataflow Diagram



Volunteer/ Professional
Application
Push Notification (if Subscribed)
Display New Message On User's Homepage
Submit Message
Display User's Homepage
Login Request
System
Notification System
Update Event Message Board
User's Homepage Information
Authenticate Access
Database
Event Message Board Stored
Retrieve User's Volunteer Group Events

Important Element 2

NOAA Report Approval Lower-Level Dataflow Diagram


Volunteers/ Professionals
NOAA
Application
Push Notification Sent
Create Event Page
Approves Report
Login Request
Update Report Map Event Color
Display User's Homepage
System
Notification System
Populate Event Details
Update Individual Report Status
User's Homepage Information
Authenticate Access
Database
Retrieve All Event Details
Update Stored Event Status
Retrieve Current Report List

USE CASE WALKTHROUGHS

Use Case 1: General Public

- The publish-subscribe architecture allows the general public to use the client in order to send a notification of a stranding event to NOAA. The GPS location, photographs, and details are sent in a message to NOAA.

Use Case 2: NOAA

- With the chosen architecture, we have established that NOAA will be able to receive notifications of potential stranding events after the general public uses the client to submit a report.
- Once a NOAA employee has validated the public reported incident, they can then use the client to create an event page and publish a notification with the event details via text to the volunteer and professional response teams in the corresponding area.
- NOAA will also interact with the client to update the event map so that other users (volunteers/organizations) can view all event statuses.
- During an ongoing event, NOAA will receive any updates sent by volunteer/professional responders and in turn, can also send out any updates until the end of the event.
- After a lead responder has ended an event, NOAA will be able to receive a notification that a stranding event has been concluded and a corresponding form has been completed.

Use Case 3: Volunteer Responder

- The publish-subscribe architecture will support the initial notification of an event to the volunteer groups through the Volunteer Deployment System that texts the users to notify them quickly that there has been an event. Users will only get notified of events that are in the group they are subscribed to based on their volunteer organization. This will be done via the volunteer database maintaining which volunteers are subscribed to which event areas.

- The architecture will also support the continued notification of the Volunteer of event ongoings based on the message board system which will be stored for long term keeping on the server and be replicated out through the event page/planner. Based on the way the Volunteer selects to be notified of certain events happening, the volunteer deployment system will either send push notifications or text messages for continued updates.
- When a volunteer selects to sign up for a shift, this will go through the server to be stored for long term storage as well as be displayed on the Events and Calendar Planner.
- When a volunteer arrives at the stranding site and selects a form to fill out for the stranding event, they will select the form directly from the government database with the most up to date information via the Forms Verification System. This will enable the user to select the most recent version of the form.
- The managing volunteer can be subscribed to notifications of form completion so that they will get an update from the notification system that the form has been completed.
- The managing volunteer can then verify the form and end the event which will notify all the volunteers via the notification system and go to the events & calendar planner to close the event and historize the event via the server into the database.

IMPLICATIONS

As discussed in the first of the failure modes, one of the biggest faults of using the publish-subscribe architecture for our system is that most of its functions rely on the connection to the three databases. A possible solution for this is to implement an offline mode and store some data locally, on the user’s device. If volunteer and professional responders could store the most commonly used, or most recently used, forms on their device, then they could still complete that step if the connection to the government database is down.

In a similar situation, if the client does not have connection to the server, then the general public user cannot submit incident reports. With an offline mode, the information for the general public incident reports (GPS, pictures, etc.) can be stored on the user’s device and sent to the server once connection has been restored. This would make the system more efficient and flexible for the users because they could still complete a vital step in the stranding response process without having to wait for connection to be restored. However, this would also make the publish-subscribe architecture slightly less reliable, since the notifications sent to all the different types of users would be delayed, and key functions like the message board and event map would be inaccurate if responders or NOAA submitted updates.

SUMMARY

TEAM MEMBER CONTRIBUTIONS

- All – Team meeting via Google Hangouts on Wednesday, 10/30/19 and continued discussion throughout the week on Slack.
- Brittany Abad – Use Case 2 Walkthrough, Implications, Assembled and reviewed HW3 final document
- Lauren Boone – Client-Server Architecture Dataflow Diagram, Key Quality Attributes for Architecture 2, Use Case 1 Walkthrough
- Christopher Feth – Database Connection Lost Failure Mode, Message Board Decomposition
- Manda Phadke – Inaccurate Form Failure Mode, NOAA Report Approval Decomposition, Use Case 3 Walkthrough
- Kunal Patadia – Publish-Subscribe Architecture Dataflow Diagram, Key Quality Attributes for Architecture 1