# 

# Slide Presentation Script

Pigeon is an encrypted messaging service where you can securely connect with your friends. You can connect with other users in chat groups called “Nests” or individually through private messaging. Your message content is encrypted so that only authorized users can read what you have to say. The name Pigeon was chosen as a reference to courier pigeons who were used to relay messages over long distances since as early as ancient Egypt.

## What needs to be completed?

As we are coming into the middle point of the semester, I’m wanting to take a moment to go through the remaining MVP features yet to be implemented into the project. Currently, there are three tasks left to be implemented:

1. Group Creation
2. Image/File Sharing
3. Shoulder Surfing Protection

### Group Creation

The idea for group creation is that a user creates a chat group called a “Nest” and can invite other users to join their Nest after searching for them by their username or selecting them from their friends list. The creator of a Nest can choose a custom name to give their Nest as well.

One idea we had for the Nest creation at the beginning of the project was the idea of protecting the Nest with a password set by the user upon Nest creation. The problem with this idea is that there is not a good way to relay the group password to a user you invited, except for through a direct message or something similar. We are now thinking that the use of invitations, rather than a search feature to find public/private groups, serves as a replacement for this password protection as uninvited users wouldn’t be able to get into the group in the first place.

### Image/File Sharing

The next feature yet to be implemented is image and file sharing. Pigeon file sharing would allow a user to attach an image or text document to their messages. These files would be uploaded to the server as a Base64 string, encrypted using the Nest’s encryption key, and stored in the database to be retrieved later.

Although there is some backend framework for this already in place, there are some problems we ran into during design. A file string could be sent to the database, encrypted, and stored, but we ran into issues with the Oracle PL/SQL parameter size limits when attempting to retrieve and decrypt the stored data. Our current proposed fix is to convert the stored data into a BLOB format, rather than the RAW format it is currently stored in.

### Shoulder Surfing Protection

The last feature to be implemented is Shoulder Surfing Protection. The planned implementation of this was that a user would only see ciphertext on the page rather than the decrypted plaintext, and they would have to click on a message to decrypt it and read it.

The problem we have run into with the design of this feature is that it seems to be incompatible with the real-time messaging aspect of Pigeon. When the page retrieves the messages for the Nest the user is looking at, it rebuilds the panel containing the messages for that group. This would mean that whenever the page updates, anything that the user has decrypted since the last update would be re-encrypted again, and they would have to click the message again to see it. Currently, the page polls the database for messages every 3 seconds, and this would effectively nullify the manual decryption and make it very hard to read messages.

# State of the Team

At this point in the project, we’re starting to really solidify habits in our teamwork, both positive and negative. I think it’s important that we take a second to reflect on what we’ve done well up to this point, and what we still need to improve on.

## What have we done well?

Our biggest strength so far in this project has been our development skill. Two specific areas that we have done very well in are our security measures and object-oriented design.

### Strict Security Measures

Since we started putting our first program concepts into actual code, we have been actively developing with data security in mind. We implemented data validation through assertion procedures in the database to ensure that no invalid data was being processed in our database. We also developed a secure schema for storing user information like passwords and authentication tokens. We’ve also successfully implemented AES-256 symmetric data encryption for message data to ensure that no unauthorized users can access message data.

### Object-Oriented Design

From the start of the project, we built our codebase to be adaptable and responsive to rapid change throughout our development lifecycle. We separated our database functionality into packages based on their functional purposes, developed an extendable messaging framework, and created both a database access class and user class for the website to interact with.

## What have we not done well?

Two areas we struggled with early on in this project were our lack of formal team meetings and our lack of communication through our software tools like Microsoft Teams and Microsoft Planner. We have recently formed response plans to help us improve in these areas.

### Lack of Formal Meetings

To respond to the lack of formal meetings, I have scheduled individual meetings with each of our team roles. These meetings occur weekly and are meant to discuss relevant information such as the state of our acceptance tests, security concerns, database design issues, and so on.

### Lack of Communication through Software Tools

To help us communicate more through our tools like Teams and Planner, we have planned to write and share documentation from our in-person meetings and are encouraging each other to ask more questions and seek feedback from each other within Teams. For Planner, we are planning to formalize the reviews of our task statuses to ensure that everything is up to date with the information required.