Requirements	Planning phase	After completion
Register a new account	✅	✅
Login to account	✅	✅
Recover password	✅	X
Edit account information	✅	✅
Join Servers	✅	✅
Invite to Servers	✅	✅
Create Servers	✅	✅
Chat within servers	✅	✅
Users can leave servers	✅	✅
Send a direct message	✅	✅
Reply to other's messages	✅	X
Mention other users	✅	✅
Add friends	✅	✅
Remove Friend	✅	✅
Admins are able to remove users from	✅	X

servers		
Accept friend request	✅	✅
Admins are able to delete a server		✅
Bad request page		✅
Users cant acces unauthorized pages		✅
Delete account		✅
Track session		✅
Navigation		✅
Landing page		✅
Send friend request		✅
Decline friend request		✅
Cancel sent friend request		✅
Log out of account		✅
Live chatting		✅
Leave server		✅

Walk through link: https://youtu.be/3Vuop_0VwfU

General Development:

1. **What did your team build? Is it feature complete and running?**

We built a Discord Clone. Yeah, it is feature complete and running. There are some missing elements from the initial design due to time constraints.

2. **How many of your initial requirements that your team set out to deliver did you actually deliver (a checklist/table would help to summarize)? Were you able to deliver everything or are there things missing? Did your initial requirements sufficiently capture the details needed for the project?**

(refer to table at the top for the user requirements)

There are some initial requirements missing to successfully describe the overall system, but these were added during the development phase

3. **What is the architecture of the system? What are the key components? What design patterns did you use in the implementation? This needs to be sufficiently detailed so that the reader will have an understanding of what was built and what components were used/created. You will need to reflect on what you planned to build vs what you built.**

The architecture that we utilized for this project was a two-tier client-server architecture, where the database, MySQL, and the server, Apache, were all run on one server, where the application logic and design, HTML5, CSS3, JavaScript, jQuery where implemented on the server side. Additionally, our project utilized the Model-View-Controller structural design pattern (MVC) by separating any viewable/interactable page into the views directory, which would represent the View, any file requiring access to the database would be separated into its own distinct class, which would represent the Model, and the any file that would facilitate the communication between the classes and interface would be separated into their own distinct files, which would represent the Controller. Additionally, we used the Observer design pattern by implementing AJAX, which stands for Asynchronous JavaScript And XML, into our JavaScript code that when the subject, which would be anything interactable, such as sending a message, friend request, and etc, would experience a change, or was interacted with, would trigger an AJAX response to update the database, as well as update the interface to reflect the change, immediately.

4. **What degree and level of re-use was the team able to achieve and why?**

We have 2 levels of reuse that we can find in our project: component reuse and object reuse. In terms of component reuse, we can resuse the homepage of our project, as well as the login and signup. What refers to the object reuse, almost all of the classes that we have can be reused including accountInfoGetter, InfoChanger, PasswordChanger, LoginFormGetter, LoginFormValidation, SignUpFormGetter, SignUpFormInserter, SignUpFormValidator, dbConnection, DirectMessageSender, DirectMessageGetter, FriendRemover, FriendRequestCanceller, FriendRequestChecker, FriendRequestHandler, FriendRequestSender, FriendRequestGetter, FriendListGetter, PotentialFriendGetter, ServerLeaver, ServerMessageGetter, ServerMessageSender.

5. **How many tasks are left in the backlog?**

   Only three (recover the password, reply to messages and delete users from the server)

CI/CD:

1. **What testing strategies did you implement? Comment on their degree of automation and the tools used. Would you (as a team) deal with testing differently in the future? Make sure to ensure that your testing report is updated to reflect what's actually been done.**

   Unit testing and operation testing. Nothing was automated. In the future we would have implemented automated testing especially on larger projects where CD/CI was required.

2. **How did your branching workflow work for the team? Were you successful in properly reviewing the code before merging as a team?**

   The branching architecture as well as creating a kanban board worked great for the team, as we were able to check the tasks that were in progress and we could also check for the progress that was being done by out teammates, as the branches were easily accessible in the kanban board. Everytime someone made a pull request, another teammate went over the code and merged if everything was alright

3. **How would your project be deployed? Is it docker ready and tested? Provide a brief description of the level of dockerization you have implemented and what would be required to deploy.**

   Because for the development we used XAMPP, if we were to deploy the project, we would run xampp on a cloud server, making use of the mySQL and apache services that are already included with it.

Reflections (Comment on the following items as a team):

1. **How did your project management work for the team? What was the hardest thing and what would you do the same/differently the next time you plan to complete a project like this?**

The kanban board was very useful to divide up tasks and keep track of the progress for each task, it also helps to know what someone is already working on. The kanban board also helped visualize how far along we were and add the priority of certain tasks

2. **Do you feel that your initial requirements were sufficiently detailed for this project? Which requirements did you miss or overlook?**

Generally, the requirement details that we had were sufficient for completing the core tasks, However, some of the requirements arose during the development process (the ones that do not have a green icon near them under the planning phase column). These include delete account, track session, navigation, send/cancel/decline friend request, live chatting, leave server, etc.

3. **What did you miss in your initial planning for the project (beyond just the requirements)?**

We miscalculated the time needed to both learn new material as well as complete various tasks. Also, some tasks were bigger than what we expected, so even if it said to be small in the kanban board, the task would end up being very large. Also, there were errors in the task priotitization as some “Critical” tasks could not be done until a “middle” importance task was completed

4. **What process did you use (ie Scrum, Kanaban..), how was it managed, and what was observed?**

We used the Kanban Process and it was managed via the projects page in GitHub. Most of the tasks were created from the initial requirements and there were attributes added to each. Each task had a priority attribute, a size attribute and a description. We used these to decide which tasks had to be done before and the description was used to have easy access to the branch that was being used for that task.

5. **As a team, did you encounter issues with different team members developing with different IDEs? In the future, would the team change anything in regard to the uniformity of development environments?**

The IDE that we initially used was VScode. However, throughout the development process we switched to phpstorm, which provided far more benefits for developing in php. So generally, the IDE used by different team members was the same so it helped as to avoid conflicts or any unnecessary hindrances.

6. **If you were to estimate the efforts required for this project again, what would you consider? (Really I am asking the team to reflect on the difference between what you thought it would take to complete the project vs what it actually took to deliver it).**

The initial efforts that we estimated to put into the project were nowhere near the actual efforts that we put in. Firstly, we needed a decent amount of time to get familiar with php and web development in general. Furthermore, there were certain tasks that required far more time to be completed (including debugging and refactoring) than aaa initially estimated.

7. **What did your team do that you feel is unique or something that the team is especially proud of (was there a big learning moment that the team had in terms of gaining knowledge of a new concept/process that was implemented).**

Throughout the project, we gained a lot of knowledge in web development. We got acquainted with php, how to build web applications and have databases integrated with them, and how to use XAMPP for running and developing php web applications. Furthermore, we gained a lot of insights of how to apply various design patterns as well as how a software development process is initiated and executed. Last but not least, working collaboratively on a single project using Kanban board helped us to understand the value and benefits of working in a team.