

Neutron Pass

A password manager

Phase 1

Project Development

Team Members and Roles

Trevor Super - Implementing Password Hashing and other potential security measures. Project Organizer. Team Leader.

Logan Templin - Implementing Manual Network Protocols, Email and Password Login, 2FA

Alejah Tillerson - Implementing Password Generation. Scrum Master.

Kevin Tripi - UI Designer / UI Developer (Cascading Style Sheets)

Yousuf Zetuna - Database Manager

Problem Statement

It's hard to keep track of the hundreds of different services that require email and a password for sign up, especially services only used once. Computing power is getting stronger every year which means longer and more complex passwords are needed to prevent brute forcing passwords. Passwords are created to be memorable and convenient over secure.

Objectives and Impact

Our objective is to create a program that allows the user to securely store passwords in a vault behind a master password. It will also help the user create strong passwords. Potential Impact: Reduced Cybercrime. Less time spent managing passwords.

Functional Requirements

- 1 - User is required to login with an email and password.
- 2 - User is prompted to verify their login session when trying to login with two factor authentication. Code is sent to email. The code has an expiration timer of 10 minutes that starts after the user receives the six-digit code.
- 3 - Strong Password Generation. Generated passwords include uppercase characters, lowercase characters, symbols, special characters, and are at least 12+ characters long.
- 4 - Login and Master Passwords are hashed.
- 5 - Login password and master password are separate. The master password is used to access the passwords in the manager.

Non-Functional Requirements

- 1 - The system should be easy to edit, add, and delete passwords
- 2 - The system should be secure to protect stored passwords
- 3 - The system should be easy to navigate from the perspective of the user.

Target Environment

Chromium Based Browsers on computers, not including mobile devices.

Technologies and Tools

Dev Platform - GitHub & Trello

Programming Languages - HTML, CSS, JavaScript

Server Runtime - NodeJS

Database - JSON

Network Protocols

HyperText Transfer Protocol Secure (HTTPS) Port 443

Simple Mail Transfer Protocol (SMTP) Port 587

Algorithms

SHA 256 Hash

Commercial off the shelf / Open source components

<https://github.com/AndersLindman/SHA256/blob/254e29ca2fb2d0697ced7e4fea03149e210ce8b9/js/sha256.js>

<https://www.npmjs.com/package/body-parser>

<https://www.npmjs.com/package/cors>

<https://www.npmjs.com/package/express>

<https://www.npmjs.com/package/nodemailer>

<https://nodejs.org/en>

Security Techniques

2FA - adds an extra layer of security by requiring users to provide two authentication factors to access their accounts. If we have extra time we could add more ways of verifying 2FA.

User Interface

We will be using Html and css to create a user interface. The user will first login to be able to access the password manager. Then, the user will be sent to a page where they will see a list of their passwords or generate a random password. In order to access the files, there will be an additional password to prevent threats from accessing the passwords.

Scrum

Scrum Plan:

1. Sprint Planning:
 - Frequency: Weekly
 - Objective: The team collaborates to select items from the product backlog and define the sprint goal.
 - Participants: Scrum Master and Development Team
 - Outcome: Sprint backlog with tasks identified for implementation during the sprint.
2. Daily Scrum:
 - Frequency: Daily
 - Objective: Team members synchronize their activities, discuss progress, and identify any obstacles.
 - Participants: Scrum Master and Development Team
 - Duration: As needed
3. Sprint Review:
 - Frequency: Weekly
 - Objective: The team presents the completed work to the development team members, gathers feedback, and discusses any adjustments to the product backlog.
 - Participants: Scrum Master and Development Team
 - Outcome: Feedback for improvement and potential updates to the product backlog.
4. Sprint Retrospective:
 - Frequency: Bi-weekly
 - Objective: The team reflects on the sprint progress, identifies strengths and areas for improvement, and creates action items for enhancing future sprints.
 - Participants: Scrum Master and Development Team
 - Outcome: Action items for process improvement and adaptation.

Artifacts:

1. Product Backlog: Managed by the development team the product backlog prioritizes features, enhancements, and security measures for the password management system.
2. Sprint Backlog: Created collaboratively during Sprint Planning, the Sprint Backlog contains tasks selected for implementation during the current sprint.
3. Increment: At the end of each sprint, the team delivers a potentially shippable product increment, which includes completed features and enhancements.

Why Scrum?

The scrum framework allows for frequent meetings with group members and allows the team to reassess and adapt our goals as the project changes. This happens during daily meetings to ensure all members are on the same page.

Agile Project Plan

User Story	Sprint #	Assigned To	Start	Finish	Story	Sprint Ready	Priority	Status	Story Points	Assigned to Sprint
	Sprint 1		5/15/2024	5/17/2024					36	
Security, Encryption	Sprint 1	Trevor	5/15/2024	5/17/2024	Yes	Yes	Medium	Complete	8	Yes
2FA, Login, Create account	Sprint 1	Logan	5/15/2024	5/17/2024	Yes	Yes	Medium	Complete	7	Yes
Github, Trello, Scrum	Sprint 1	Alejah	5/15/2024	5/17/2024	Yes	Yes	Medium	Complete	6	Yes
UI Design, CSS	Sprint 1	Kevin	5/15/2024	5/17/2024	Yes	Yes	Medium	Complete	7	Yes
SQL	Sprint 1	Yousuf	5/15/2024	5/17/2024	Yes	Yes	Medium	Complete	8	Yes
	Sprint 2		5/14/2024	5/24/2024					15	
Use Case Diagram	Sprint 2	Alejah, Trevor	5/17/2024	5/17/2024	Yes	Yes	High	Complete	5	Yes
Use Case Specification	Sprint 2	Trevor, Yousuf	5/17/2024	5/18/2024	Yes	Yes	High	Complete	5	Yes
Domain Class Diagram	Sprint 2	Logan	5/14/2024	5/24/2024	Yes	Yes	High	Complete	5	Yes
	Sprint 3		5/25/2024	6/6/2024					21	
Design Class Diagram	Sprint 3	Logan	5/25/2024	6/6/2024	Yes	Yes	Medium	Complete	8	Yes
Design Sequence Diagrams	Sprint 3	Alejah, Trevor, Yousuf, Kevin	5/25/2024	6/6/2024	Yes	Yes	Medium	Complete	13	Yes
	Sprint 4		6/7/2024	6/14/2024					22	
Password Generator Test	Sprint 4	Alejah, Trevor, Kevin	6/7/2024	6/13/2024	Yes	Yes	Medium	Complete	8	Yes
Password Toggle Test	Sprint 4	Trevor, Logan	6/7/2024	6/14/2024	Yes	Yes	Medium	Complete	5	Yes
Database	Sprint 4	Yousuf	6/7/2024	6/14/2024	Yes	Yes	High	Complete	9	Yes

Estimation

Functional Requirements:

1. External Inputs (EI):

- User login with email and password
- Two-factor authentication code verification
- Strong password generation
- Organizing passwords in folders

Ø Total EI = 4

2. External Outputs (EO):

- Displaying organized list of passwords

Ø Total EO = 1

3. External Inquiries (EQ):

- N/A

4. Internal Logical Files (ILF):

- Stored passwords in the vault
- User login session information

Ø Total ILF = 2

5. External Interface Files (EIF):

- N/A

Non-functional Requirements:

1. User-friendliness
2. Security
3. Reliability

GSC1: Reliable Backup & Recover (1)

GSC2: Use of Data Communication (2)

GSC3: Use of Distributed Computing (1)

GSC4: Performance (2)

GSC5: Realization in heavily used configuration (2)

GSC6: On-line data entry (3)

GSC7: User Friendliness (2)

GSC8: On-line data change (3)

GSC9: Complex user interface (1)

GSC10: Complex procedures (3)

GSC11: Reuse (1)

GSC12: Ease of installation (1)

GSC13: Use at multiple sites (1)

GSC14: Adaptability and flexibility (2)

$UFP = (EI \times \text{Weight}) + (EO \times \text{Weight}) + (ILF \times \text{Weight})$

(Using Average Weight Factors)

$UFP = (4 \times 4) + (1 \times 4) + (2 \times 7) = 16 + 4 + 14 = 34$

GSC = 25

$VAF = 0.65 + (0.01 \times 25) = 0.9$

$AFP = UFP \times VAF$

$AFP = 34 \times 0.90 = 30.6$

We will assume that the performance factor is 2. $PF = 2$

$\text{Project length} = AFP / PF$

$\text{Project length} = 30.6 / 2 = 15.3$

The estimated number of days to complete the project is about 15.3 days.

Risk Analysis

Project: Neutron Pass

Risk type: Security

Priority (1 low ... 5 critical): 5

Risk factor: Project completion will depend on tests which require software component under development.

Software component delivery may be delayed

Probability: 65%

Impact: Loss of users as project promises increased security, yet could lead to a data leak.

Monitoring approach: Code Reviews

Contingency plan: Don't ship unreviewed code that contains security flaws

Estimated resources: A continuous commitment in development and longer development periods should be expected.

Risk Analysis

Project: Neutron Pass

Risk type: Security

Priority (1 low ... 5 critical): 4

Risk factor: Implementing weak encryption algorithms could compromise the security of the password manager

Probability: 50%

Impact: Project completion will be delayed for each day a proper encryption algorithm is not chosen.

Monitoring approach: Proof of concept

Contingency plan: Delay implementation until proper encryption algorithm is chosen

Estimated resources: One Week

Risk Analysis

Project: Neutron Pass

Risk type: Development

Priority (1 low ... 5 critical): 3

Risk factor: Project completion will be affected by novice experience of the team. This will require extra effort and time to learn basic web development to overcome.

Probability: 90%

Impact: Slow Development, Frustration.

Monitoring approach: Weekly team meetings calls to assess progress and time remaining until deadline.

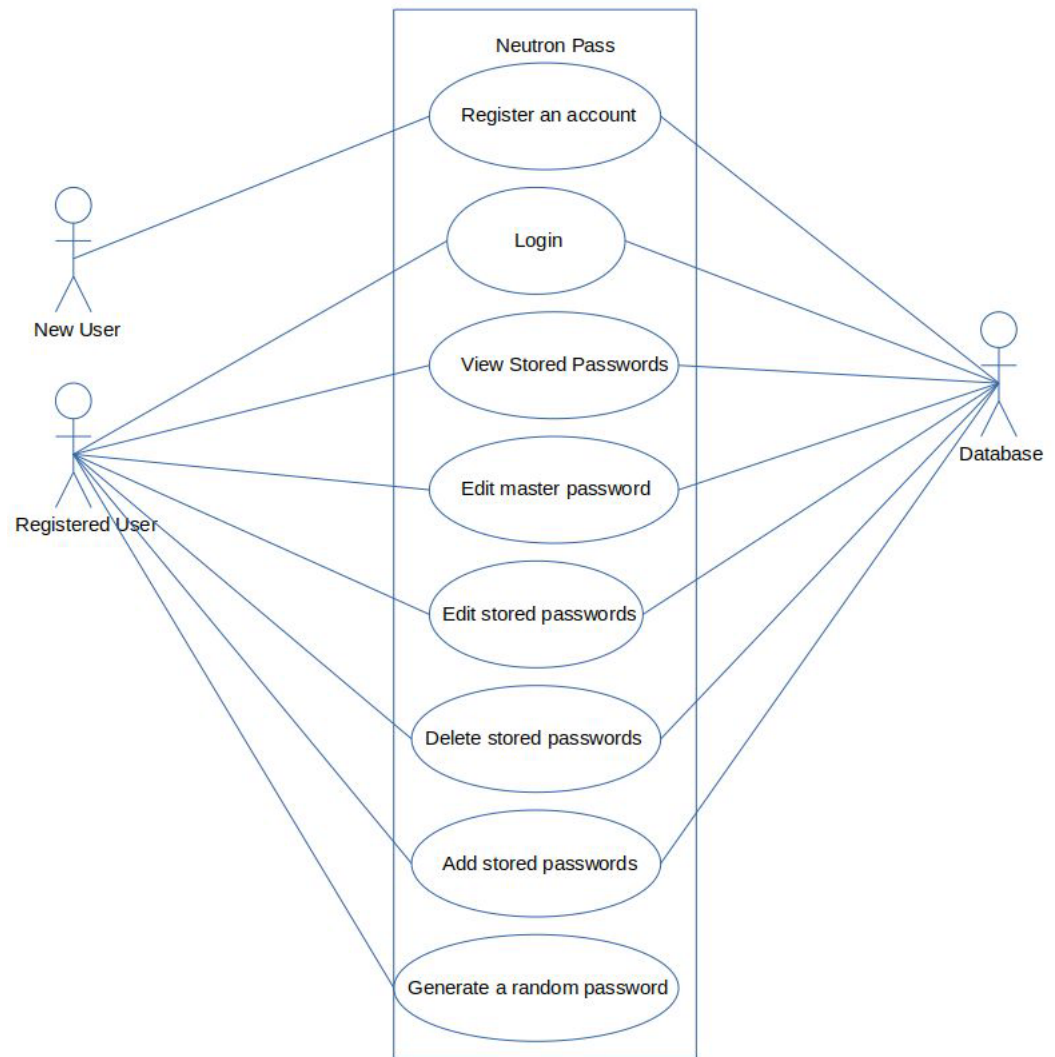
Contingency plan: Scaling back on number of features to develop

Estimated resources: At least as much time spent developing will be spent researching

Phase 2

Requirements Analysis

Use Case Diagram



Fully Dressed Use Case Specifications

Register an account

Scope: Password Management System

Level: User Goal

Primary Actor: New User

Stakeholders and Interests: User: Wants to create an account to use the password management system.

System Administrator: Wants to ensure accounts are created securely.

Preconditions: User has access to the registration page.

Success Guarantee (Postcondition): The account is successfully created and the user is logged in.

Main Success Scenario: 1. User navigates to the registration page.

2. User fills in the registration form with necessary details.

3. User submits the form.

4. System validates the input and creates the account.

5. System sends a confirmation email.

Extensions: 1a: User enters an already taken email.

2a: Account creation fails

Special Requirements: Must comply with The General Data Protection Regulation (GDPR)

Technology and Data Variation List: Supports Web Application, data in JSON format.

Frequency of Occurrences: Infrequent, typically once per user.

Miscellaneous: Open issue - verification method for email?

Login

Scope: Password Management System

Level: User Goal

Primary Actor: Registered User

Stakeholders and Interests: User: Wants secure and quick access to their account.

System Administrator: Wants to ensure authentication is secure.

Preconditions: User must have a registered account.

Success Guarantee (Postcondition): User is logged in and granted access to their data.

Main Success Scenario: 1. User navigates to the login page.

2. User enters email and password.

3. System validates credentials.

4. System sends the user a code

5. User enters the code

6. System logs the user in.

Extensions: 1a: User enters incorrect email or password.

2a: Server error occurs, preventing verification.

3a: User provides an incorrect code.

Special Requirements: Supports CAPTCHA to prevent bot attacks.

Technology and Data Variation List: Supports Web Application, Data in JSON format.

Frequency of Occurrences: Frequent, typically daily by users.

Miscellaneous: Open issue - Support for biometric login?

View Stored Passwords

Scope: Password Management System

Level: User Goal

Primary Actor: Registered User

Stakeholders and Interests: User: Wants to access their saved passwords and folders easily.

System Administrator: Wants to ensure the data is displayed securely.

Preconditions: User must be logged in.

Success Guarantee (Postcondition): Passwords can be viewed by the user.

Main Success Scenario: 1. User logs in.

2. User navigates to the password manager page.

3. User enters master password

4. System displays the user's stored passwords.

Extensions: 1a: Error occurs, preventing the list from loading.

Special Requirements: Ensure data is encrypted and decrypted securely.

Technology and Data Variation List: Web Application, Data in JSON format.

Frequency of Occurrences: Frequent, typically daily by users.

Miscellaneous: Open issue - Paginate or infinite scroll for large data sets?

Edit Master Password

Scope: Password Management System

Level: Subfunction

Primary Actor: Registered User

Stakeholders and Interests: User: Wants to change their master password.

System Administrator: Wants to ensure the process is secure.

Preconditions: User must be logged in.

Success Guarantee (Postcondition): Master password is updated.

Main Success Scenario: 1. User navigates to the password manager.

2. User enters the current master password.

3. User enters a new master password.

4. System updates the password.

Extensions: 1a: User enters an incorrect current master password.

2a: Error occurs, preventing the update.

Special Requirements: Must enforce strong password policies.

Technology and Data Variation List: Web Application, Data in JSON format.

Frequency of Occurrences: Infrequent, typically during security reviews or incidents.

Miscellaneous: Open issue - Automatic logout after password change?

Edit Stored Passwords

Scope: Password Management System

Level: Subfunction

Primary Actor: Registered User

Stakeholders and Interests: User: Wants to update or correct stored passwords.

Preconditions: User must be logged in.

Success Guarantee (Postcondition): Specified passwords are updated.

Main Success Scenario: 1. User navigates to the password manager page.

2. User enters master password

3. User selects the password to edit.

4. User updates the password details.

5. System saves the changes.

Extensions: 1a: User makes edits and saves changes.

1b: User enters invalid data.

2a: Error occurs, preventing the update.

Special Requirements: Ensure data integrity during updates.

Technology and Data Variation List: Web Application, Data supported in JSON format.

Frequency of Occurrences: Occasional, as needed by users.

Miscellaneous: Open issue - Notify user of weak updated passwords?

Delete Stored Passwords

Scope: Password Management System

Level: Subfunction

Primary Actor: Registered User

Stakeholders and Interests: User: Wants to remove obsolete or incorrect passwords.

System Administrator: Wants to ensure deletions are secure and intentional.

Preconditions: User must be logged in.

Success Guarantee (Postcondition): Specified password is deleted.

Main Success Scenario: 1. User navigates to the password manager page.

2. User enters master password

3. User selects the password to delete.

4. System prompts for confirmation.

5. User confirms deletion.

6. System deletes the password.

Extensions: 1a: User cancels the deletion at the confirmation prompt.

2a: Error occurs, preventing the deletion.

Special Requirements: Must have a confirmation step to avoid accidental deletion.

Technology and Data Variation List: Web Application, Data supported in JSON format.

Frequency of Occurrences: Infrequent, occasional cleanup by users.

Miscellaneous: Open issue - Undo feature for deletion?

Add Stored Passwords

Scope: Password Management System

Level: Subfunction

Primary Actor: Registered User

Stakeholders and Interests: User: Wants to add new passwords to the system.

System Administrator: Wants to ensure password addition is secure.

Preconditions: User must be logged in.

Success Guarantee (Postcondition): New password is added to the system.

Main Success Scenario: 1. User navigates to the password manager page.

2. User enters master password

3. User clicks add

4. User enters the password details.

5. System saves the new password.

Extensions: 1a: User enters password details and saves.

2a: Server error occurs, preventing the addition.

Special Requirements: Ensure data is encrypted and stored securely.

Technology and Data Variation List: Web Application, Data supported in JSON format.

Frequency of Occurrences: Frequent, typically as users acquire new accounts.

Miscellaneous: Open issue - Bulk addition support?

Generate a random password

Scope: Password Management System

Level: Subfunction

Primary Actor: Registered User

Stakeholders and Interests: User: Wants to generate strong, random passwords.

System Administrator: Ensures generated passwords are secure and meet complexity requirements.

Preconditions: User must be logged in.

Success Guarantee (Postcondition): Secure password is generated and presented to the user.

Main Success Scenario: 1. User navigates to the password generator section.

2. User selects the desired criteria for the password.

3. System generates the password.

4. System presents the password to the user.

Extensions: 1a: Password generation fails due to invalid criteria.

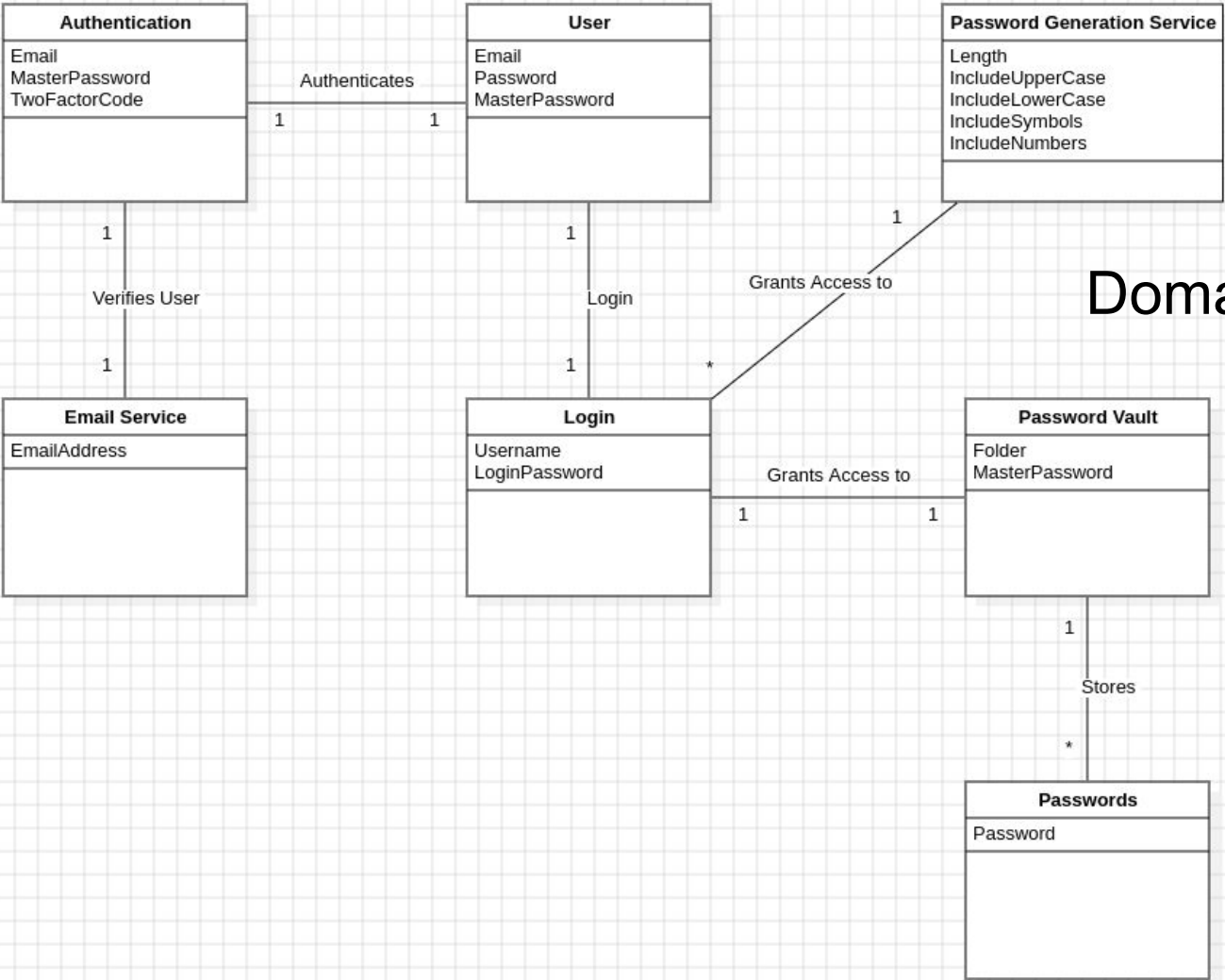
Special Requirements: Must comply with best practices for secure passwords.

Technology and Data Variation List: Web Application, Data supported in JSON format.

Frequency of Occurrences: Frequent, as needed by users.

Miscellaneous: Open issue - Support for password policies of various services?

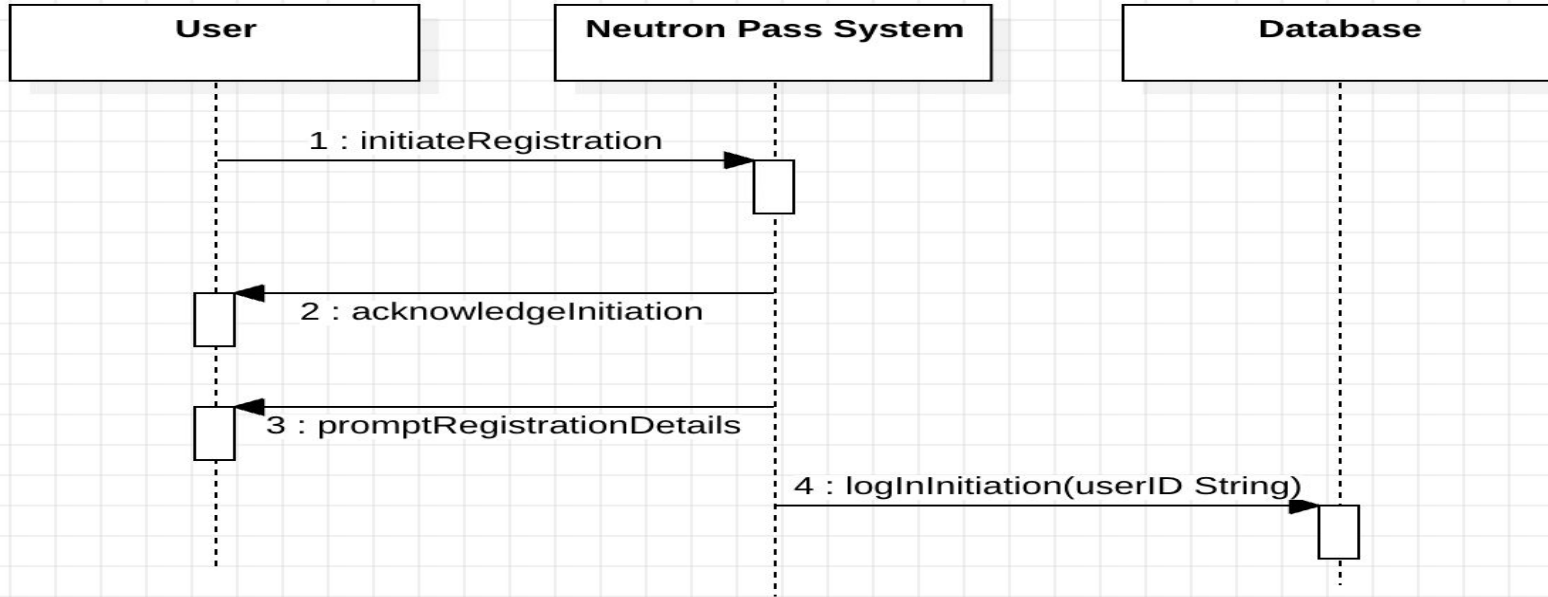
Domain Class Diagram



Phase 3

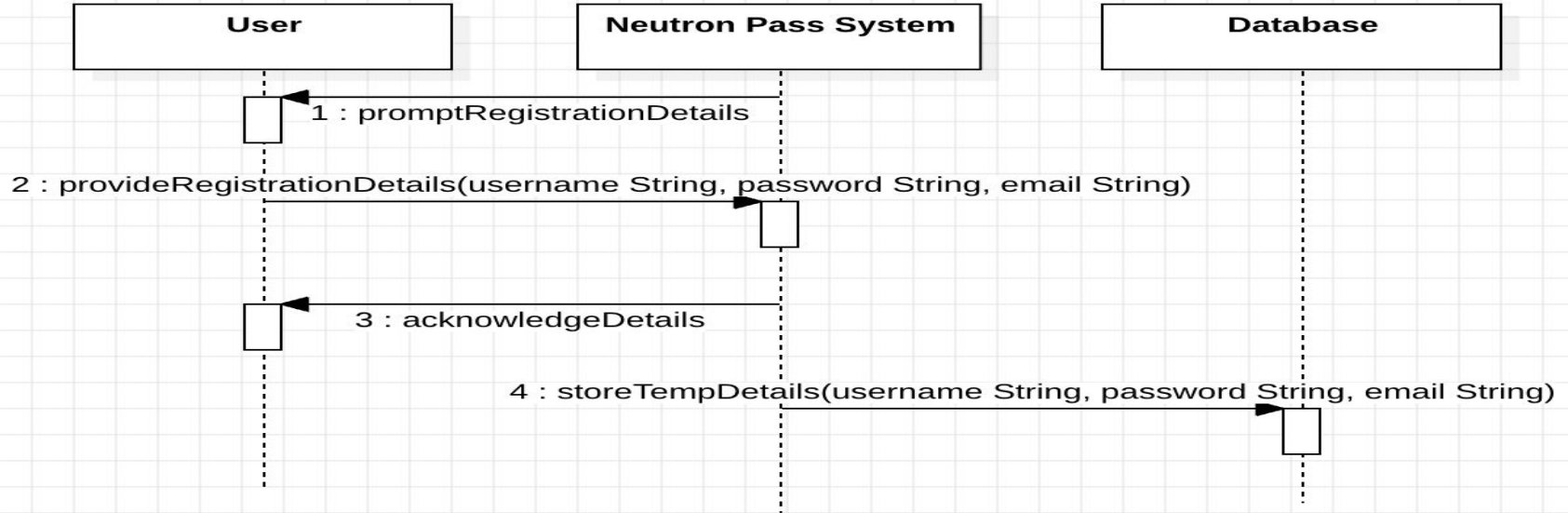
Design Modeling

Register An Account Design Sequence Diagrams



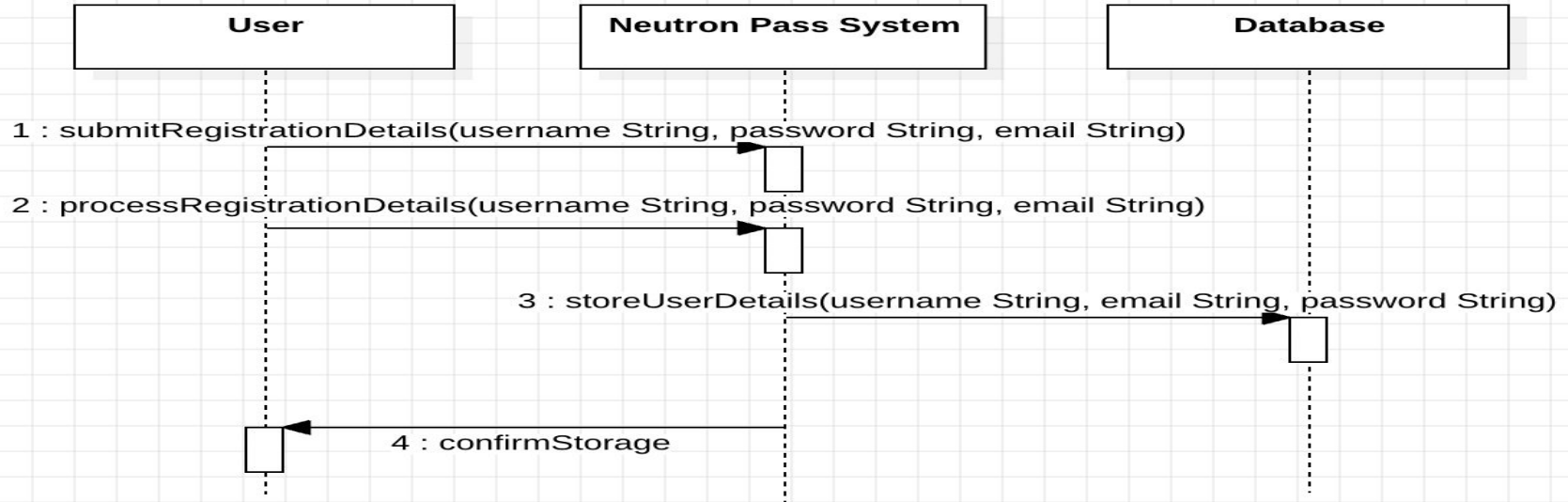
Initiate Registration

Register An Account Design Sequence Diagrams



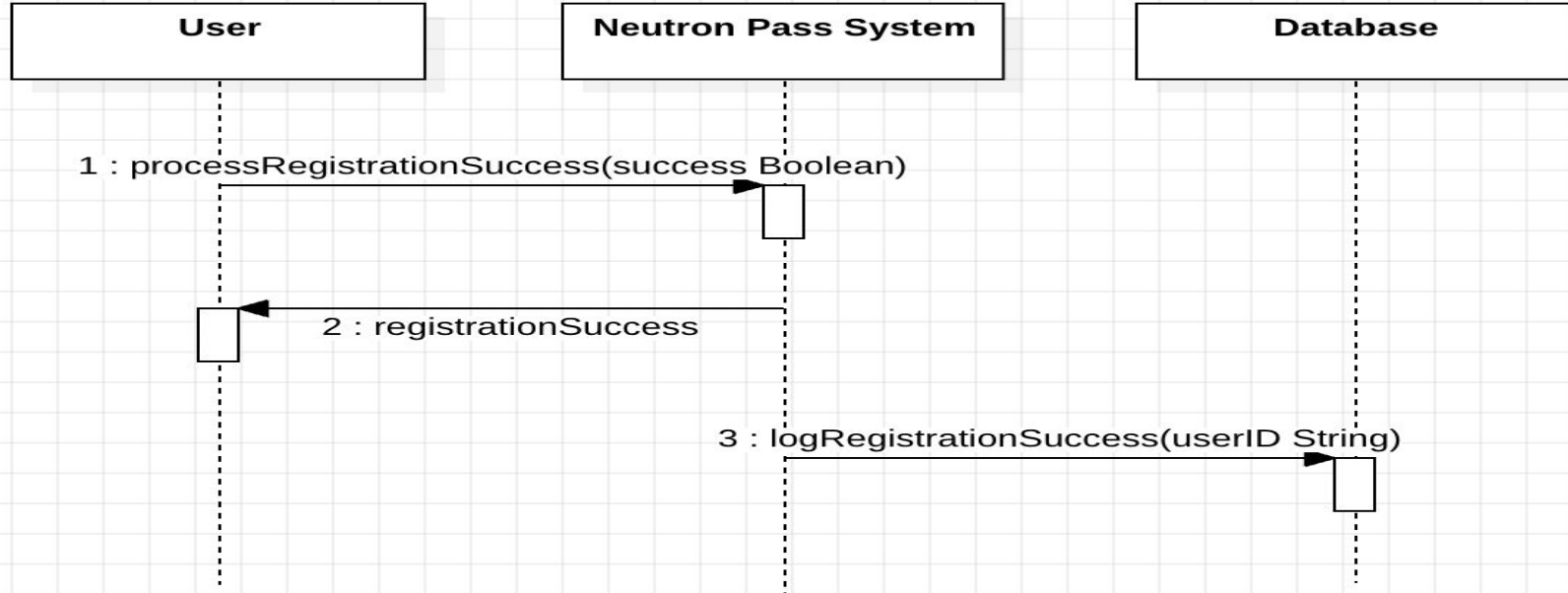
Prompt For Details

Register An Account Design Sequence Diagrams



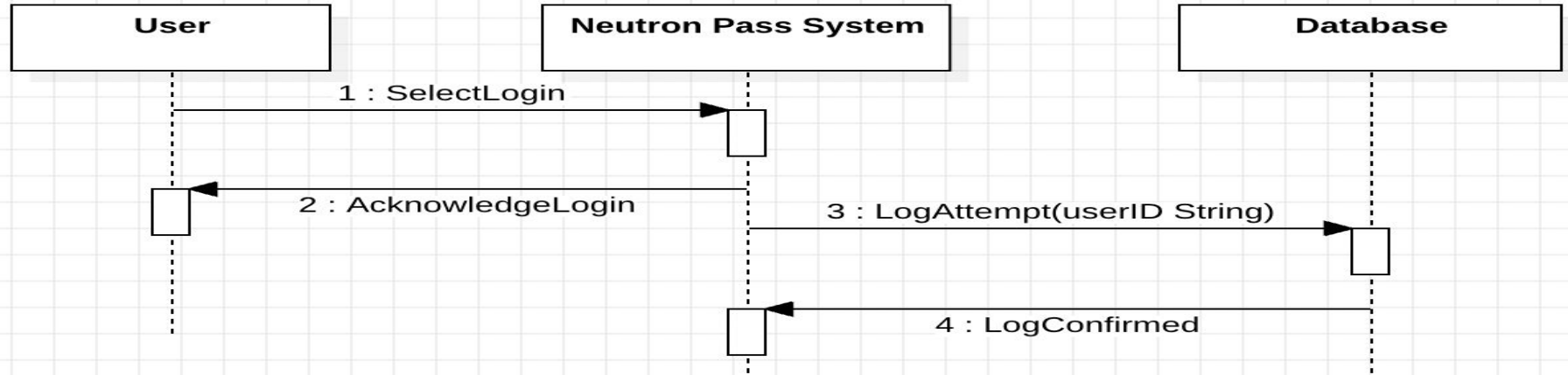
Submit Details

Register An Account Design Sequence Diagrams



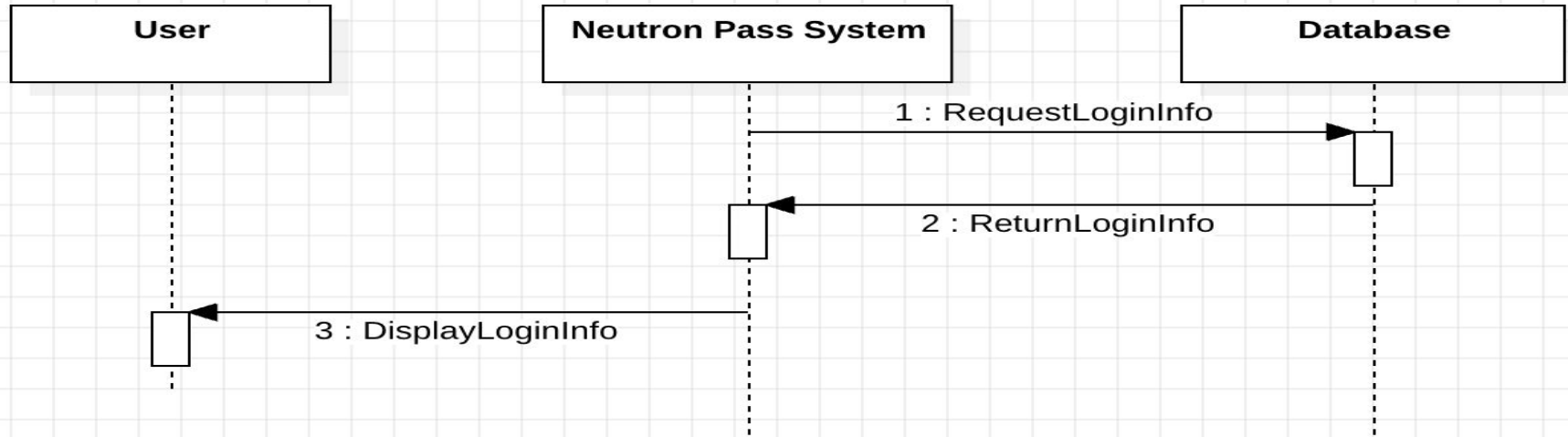
Confirm Registration Success

Login Design Sequence Diagrams



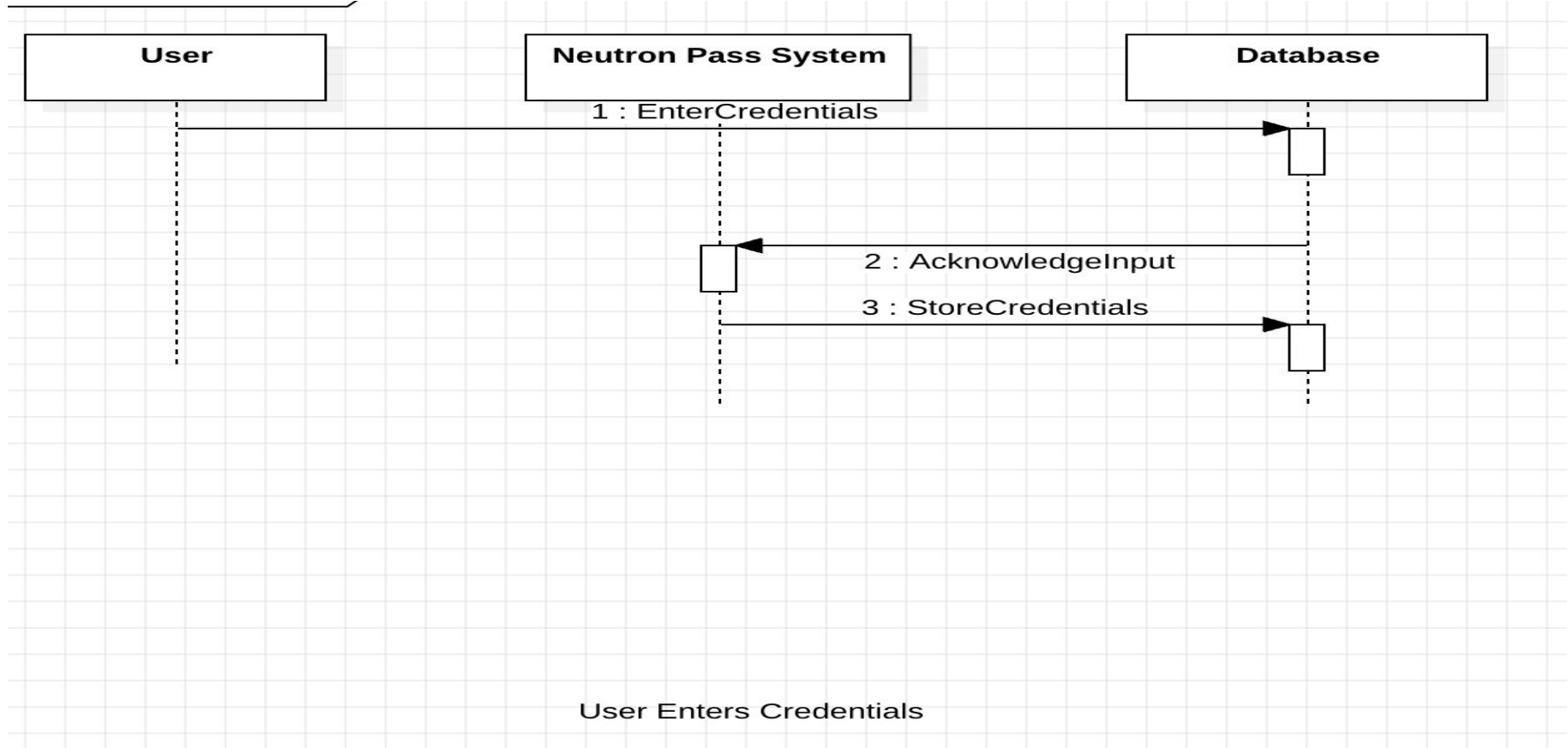
User Selects Login

Login Design Sequence Diagrams

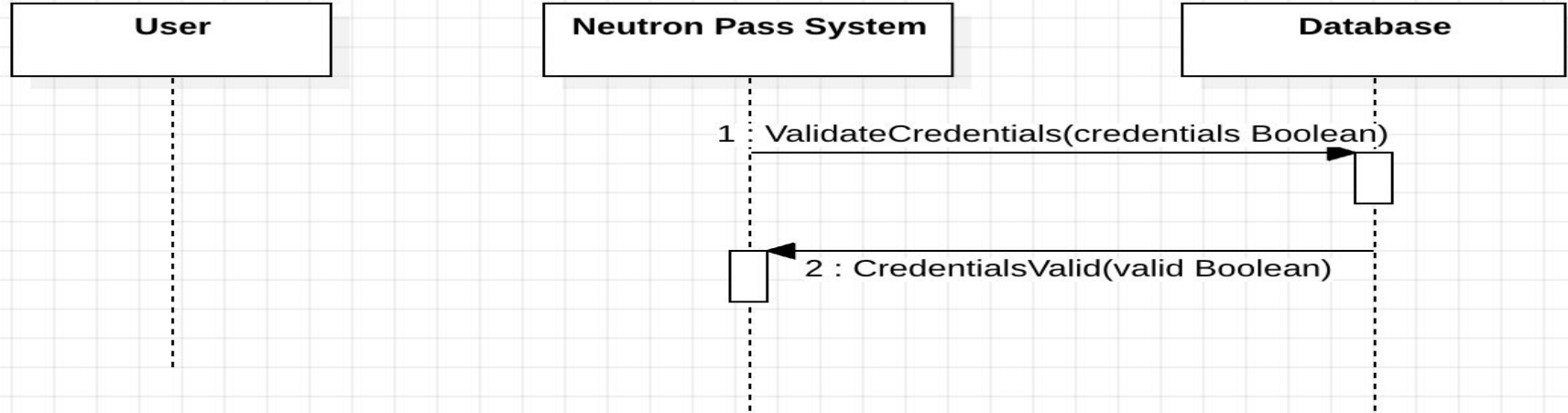


Display Login To User

Login Design Sequence Diagrams

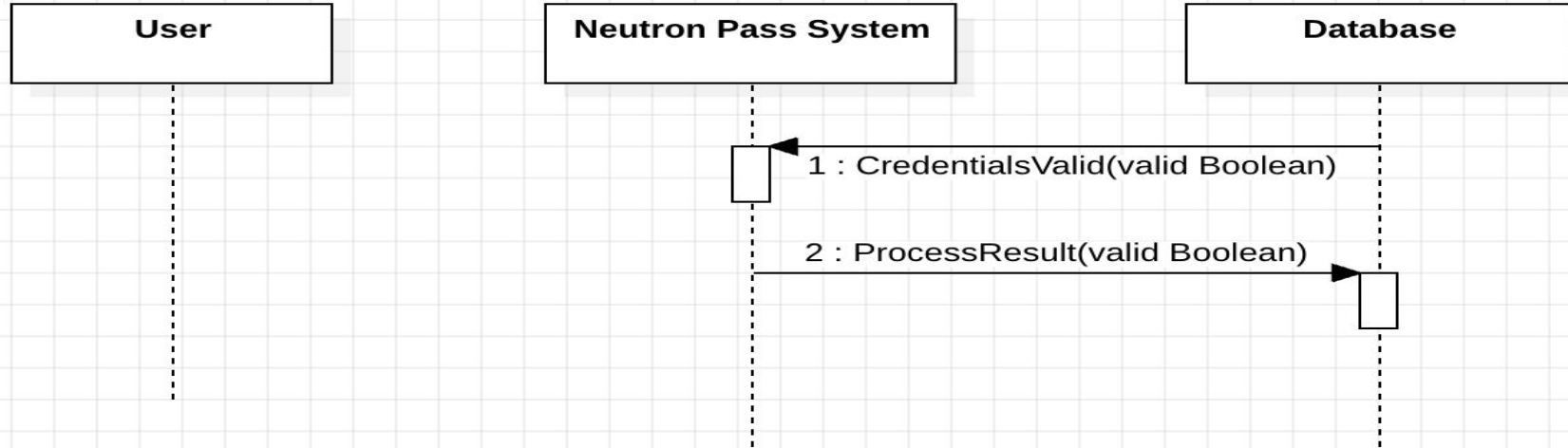


Login Design Sequence Diagrams



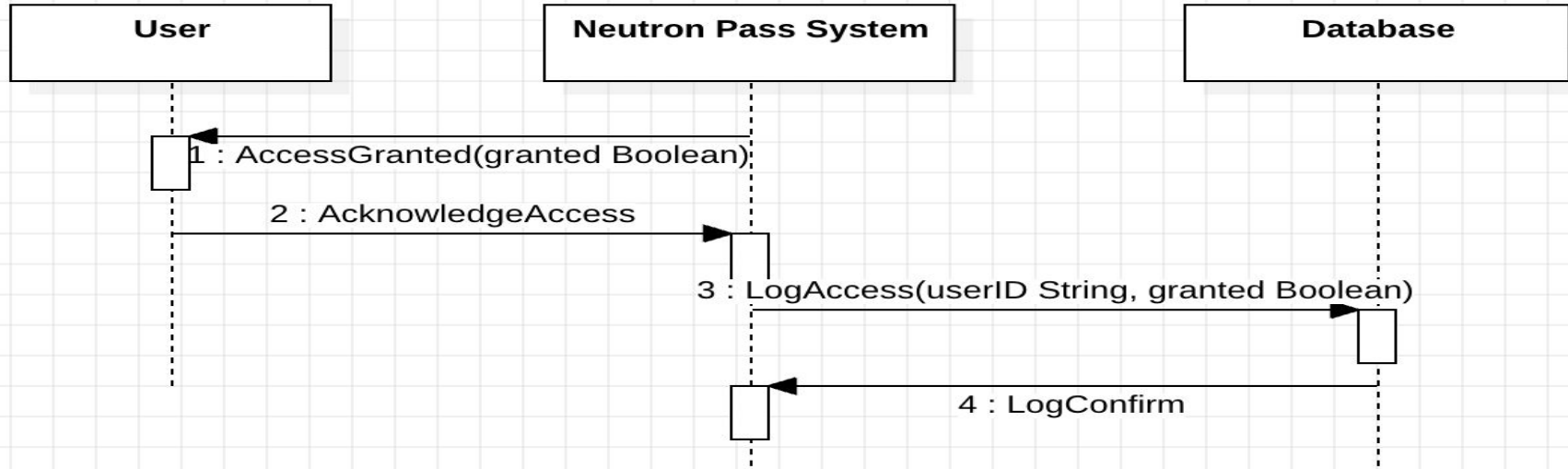
Credentials Are Validated

Login Design Sequence Diagrams



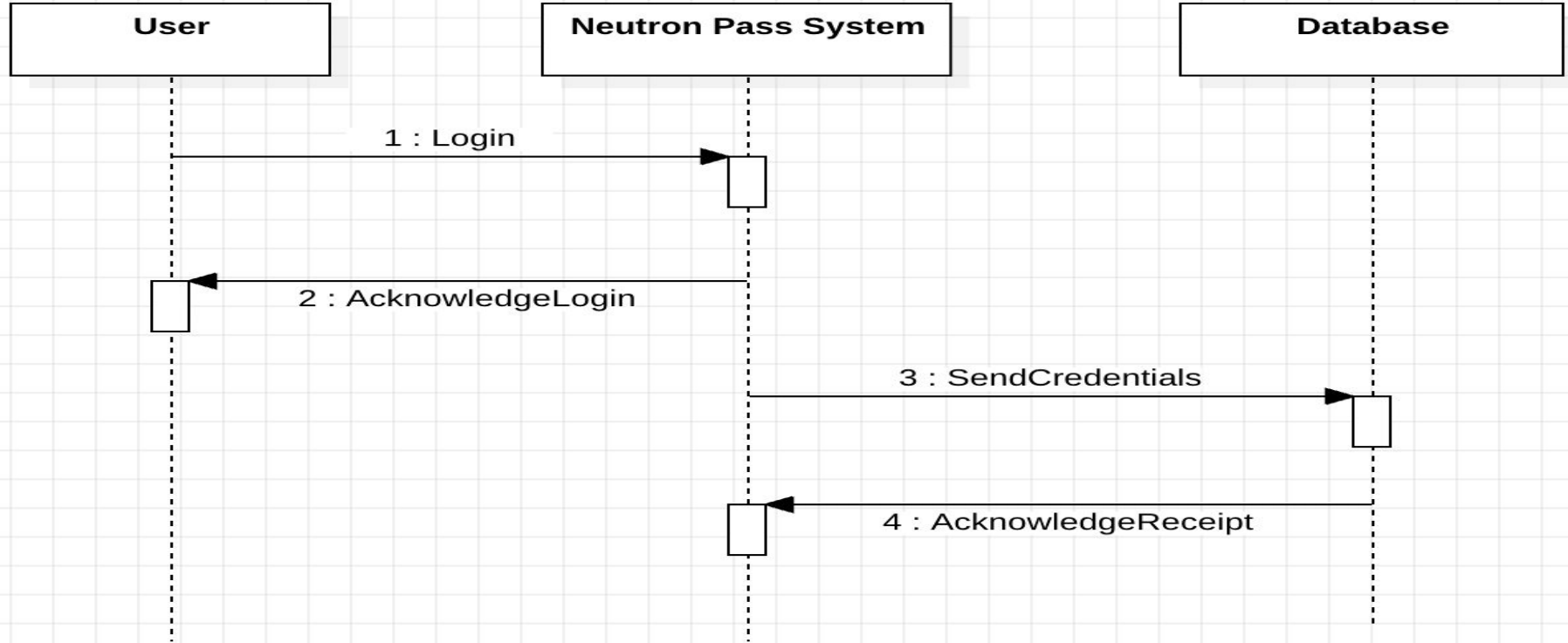
Database Validate Credentials

Login Design Sequence Diagrams



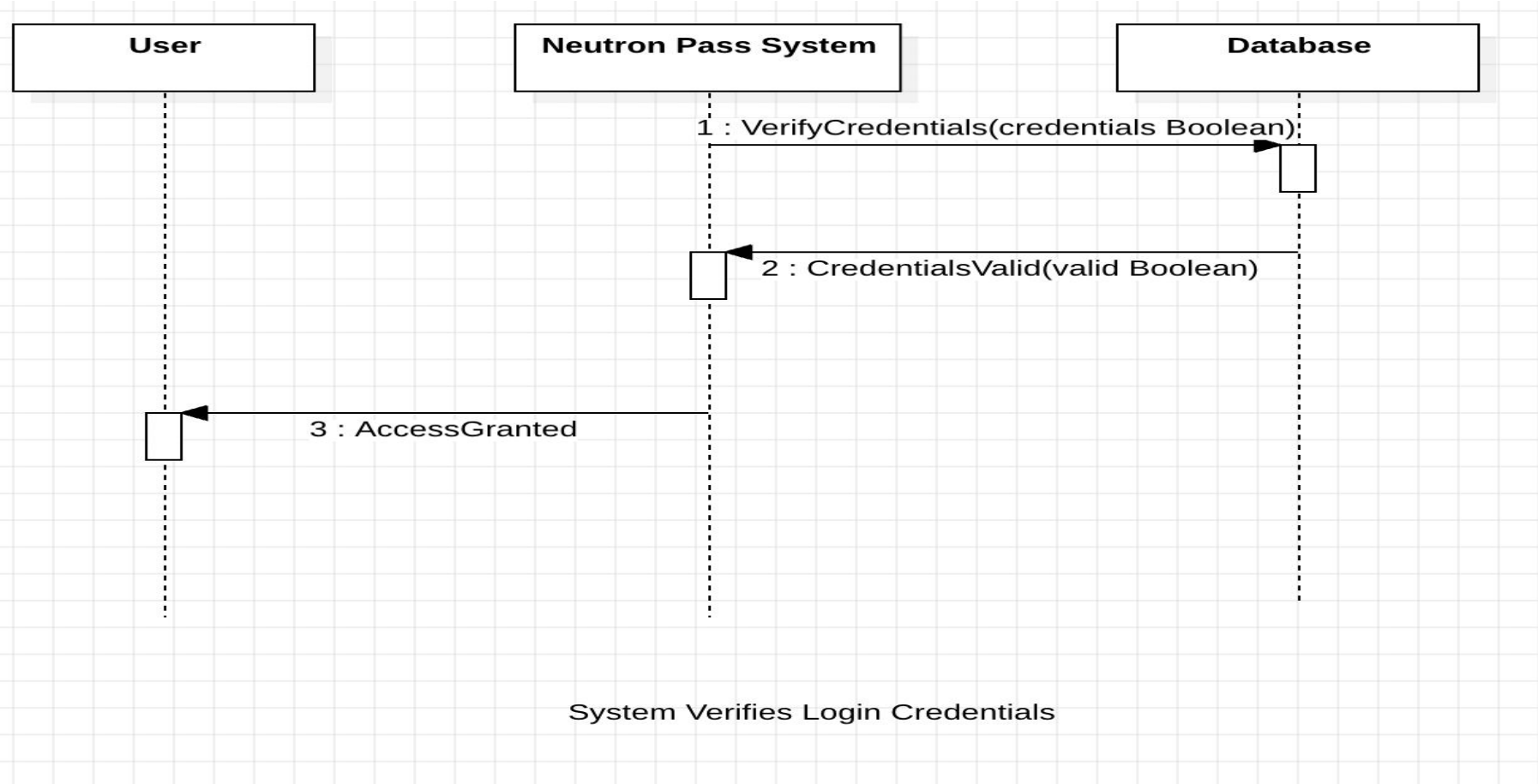
Access Is Granted To User

View Stored Passwords Design Sequence Diagrams

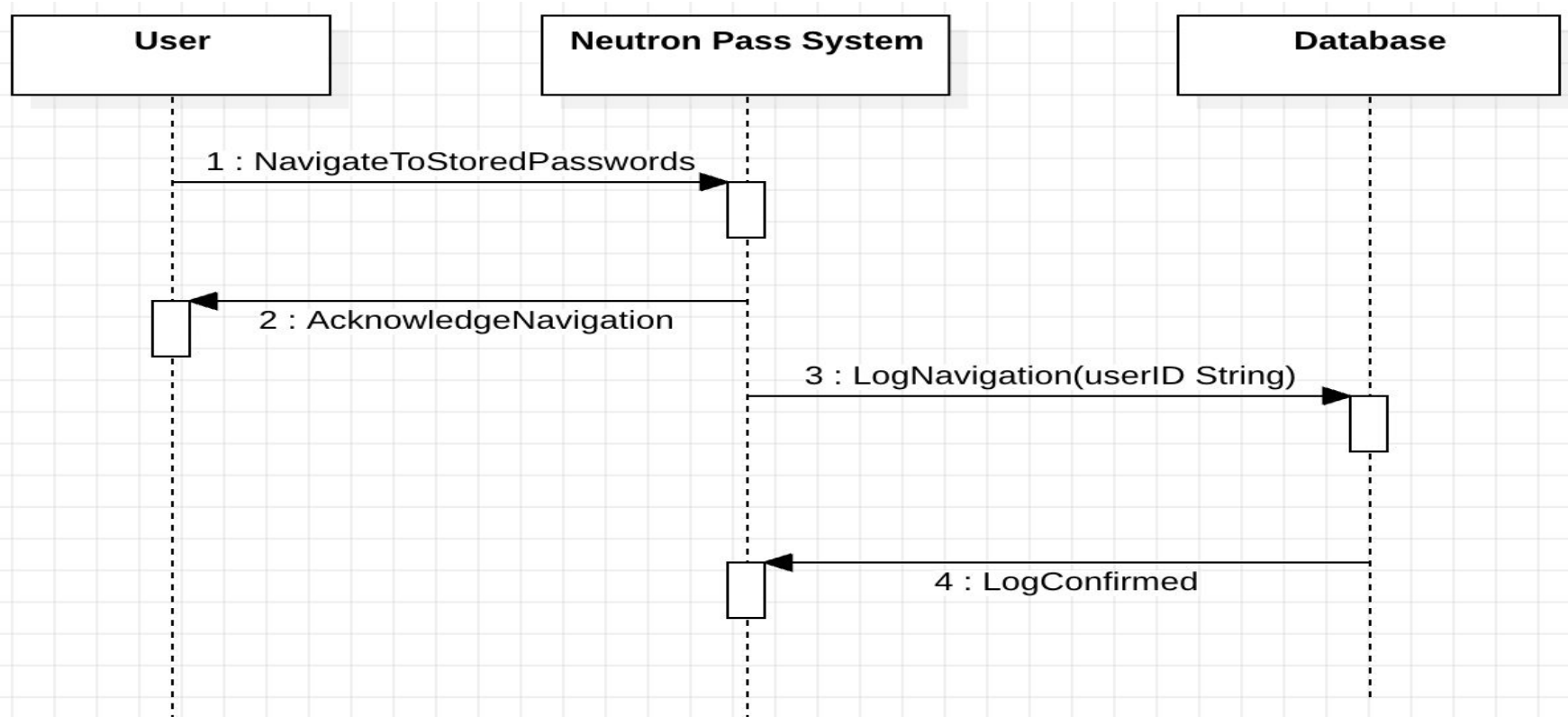


User Logs In

View Stored Passwords Design Sequence Diagrams

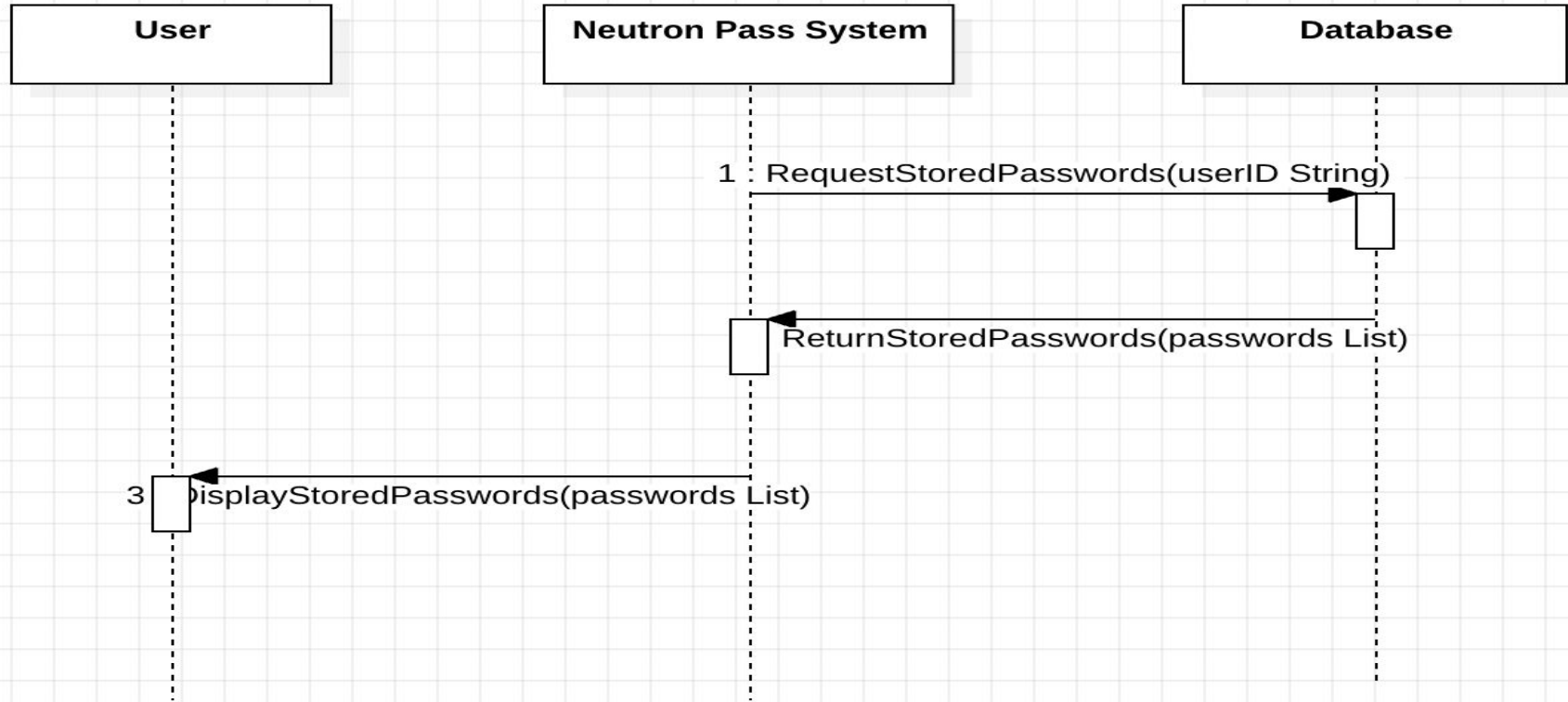


View Stored Passwords Design Sequence Diagrams



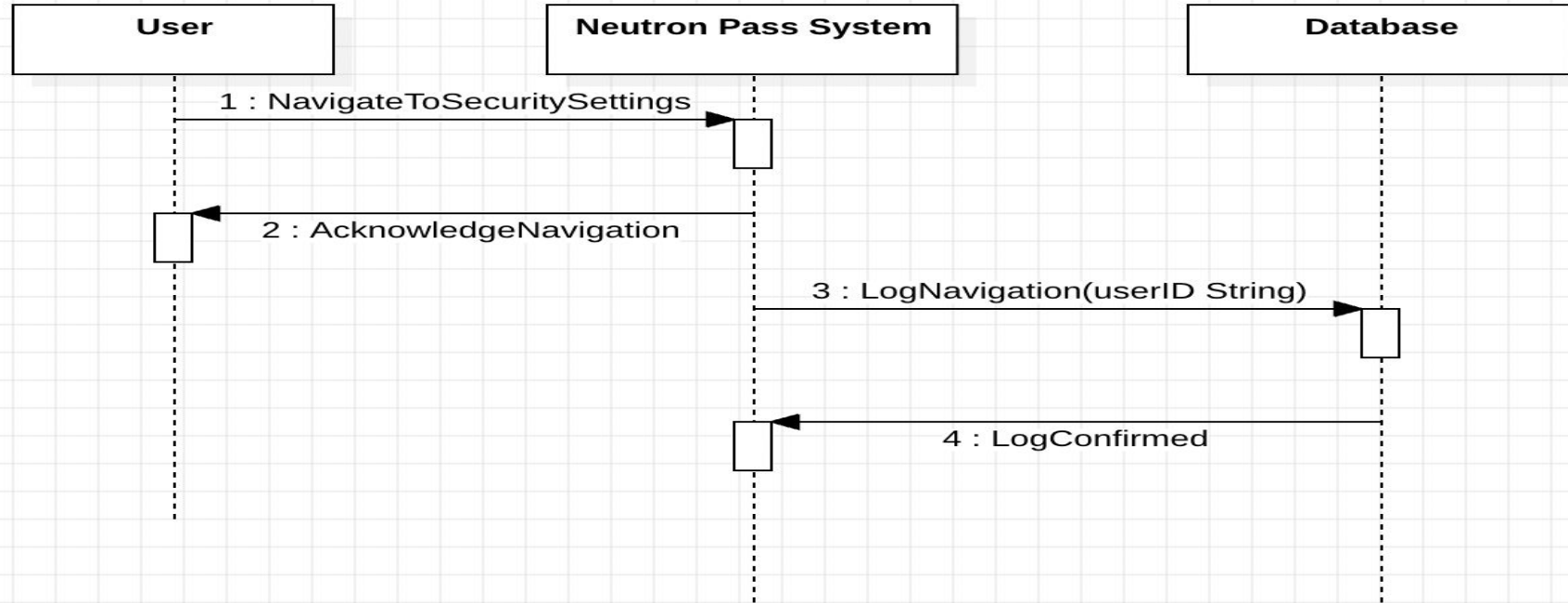
User Goes To Stored Passwords

View Stored Passwords Design Sequence Diagrams



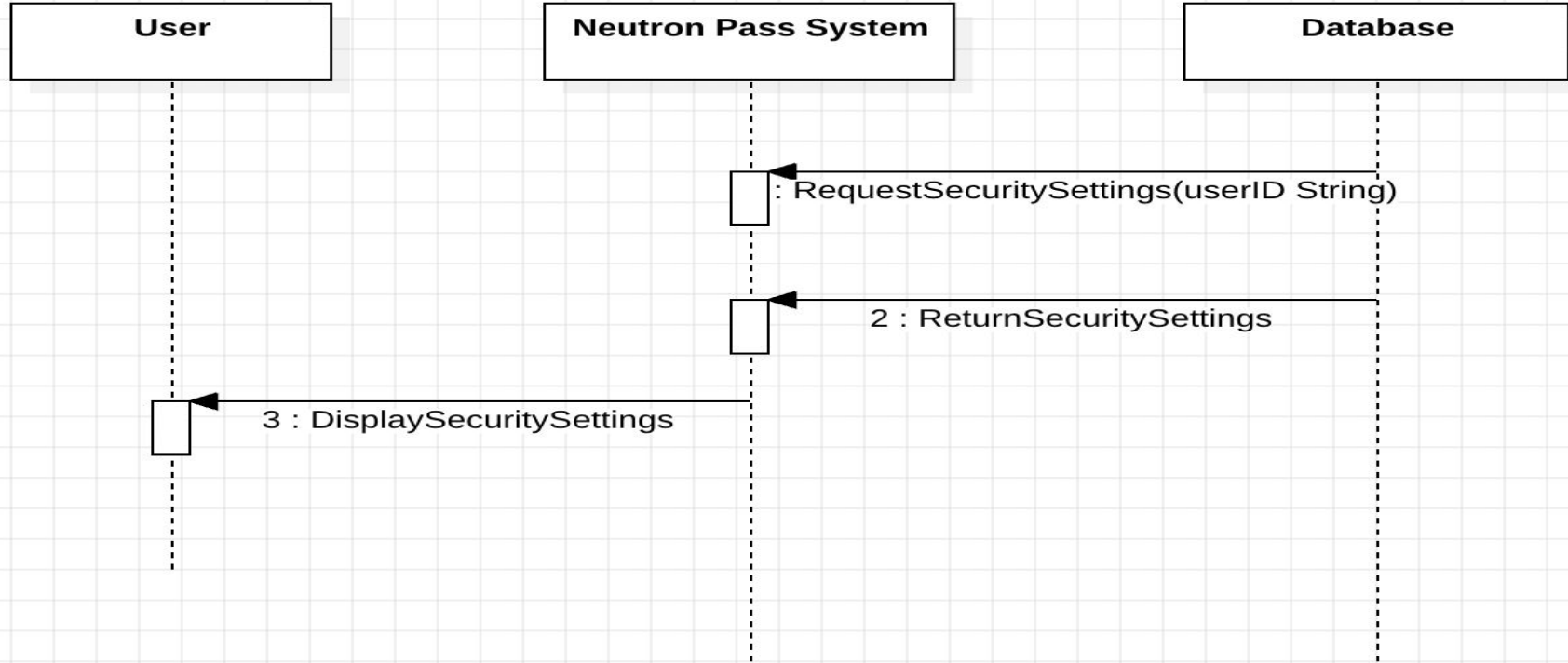
System Shows Stored Passwords

Edit Master Password Design Sequence Diagrams



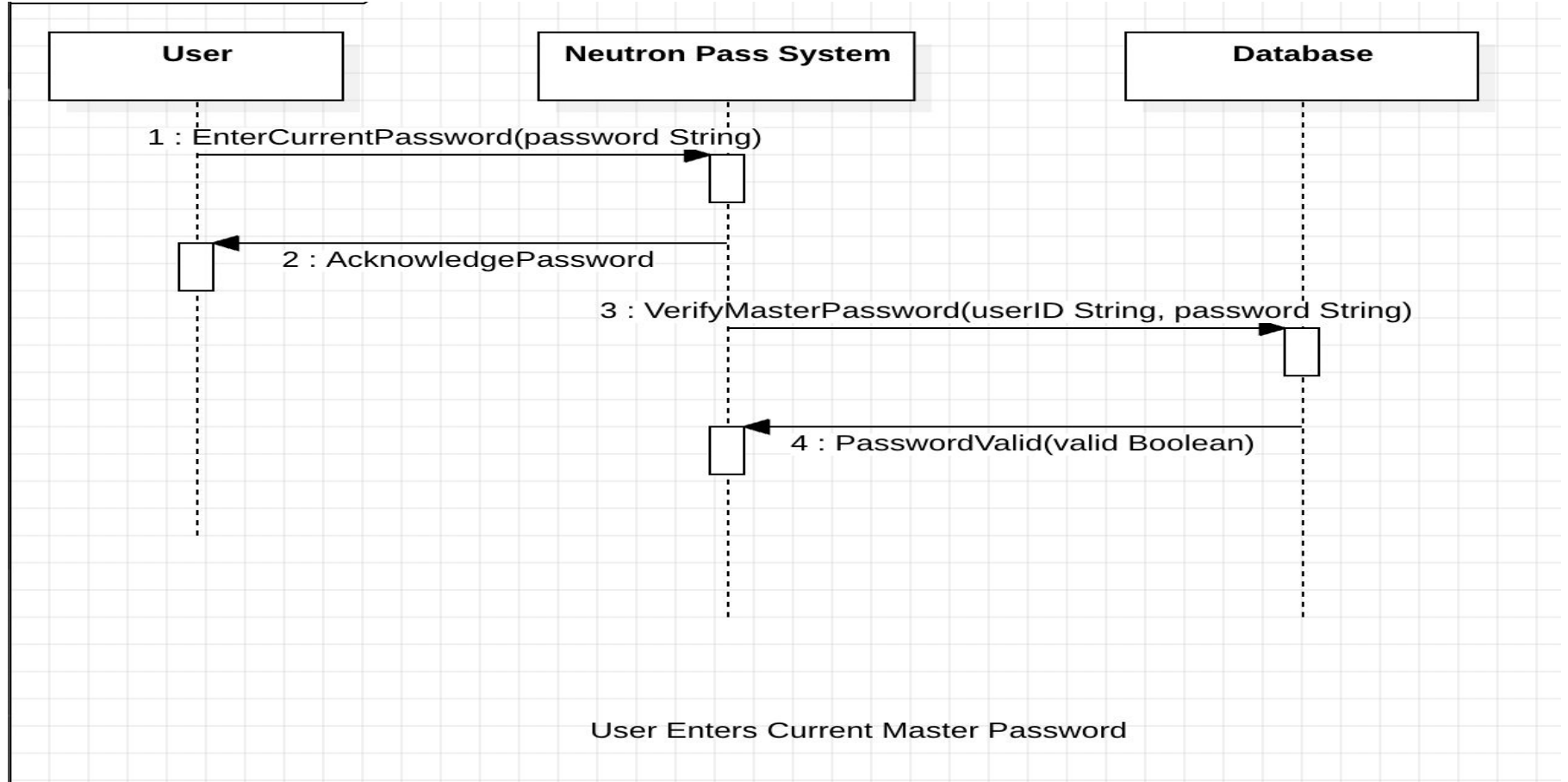
User Navigates To The Security Settings

Edit Master Password Design Sequence Diagrams

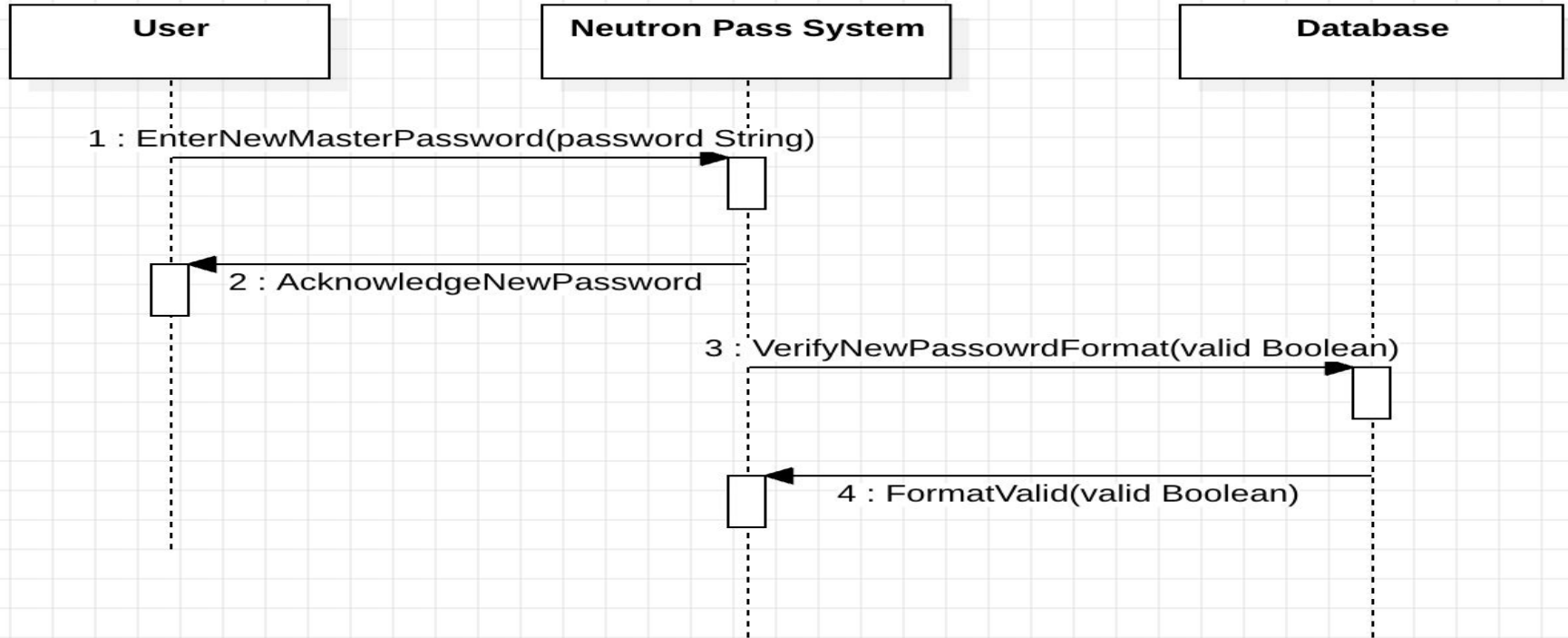


System Displays Security Settings

Edit Master Password Design Sequence Diagrams

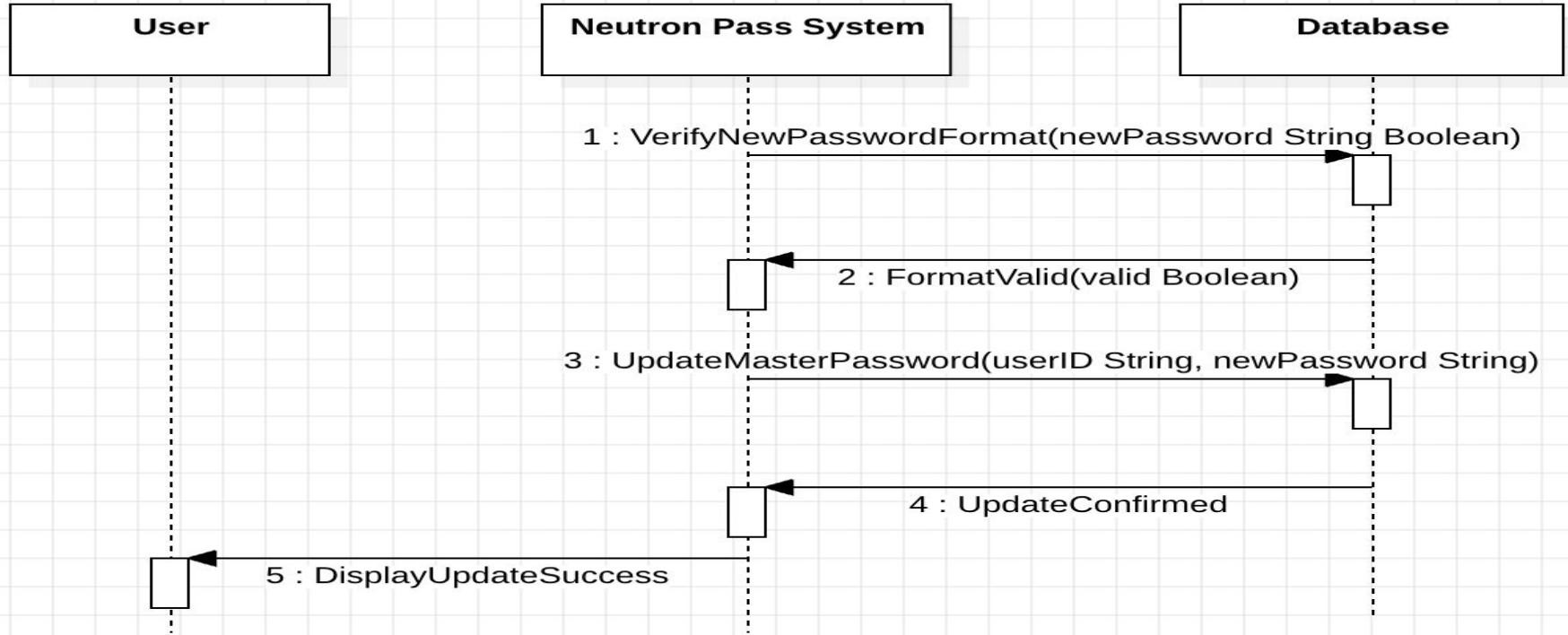


Edit Master Password Design Sequence Diagrams



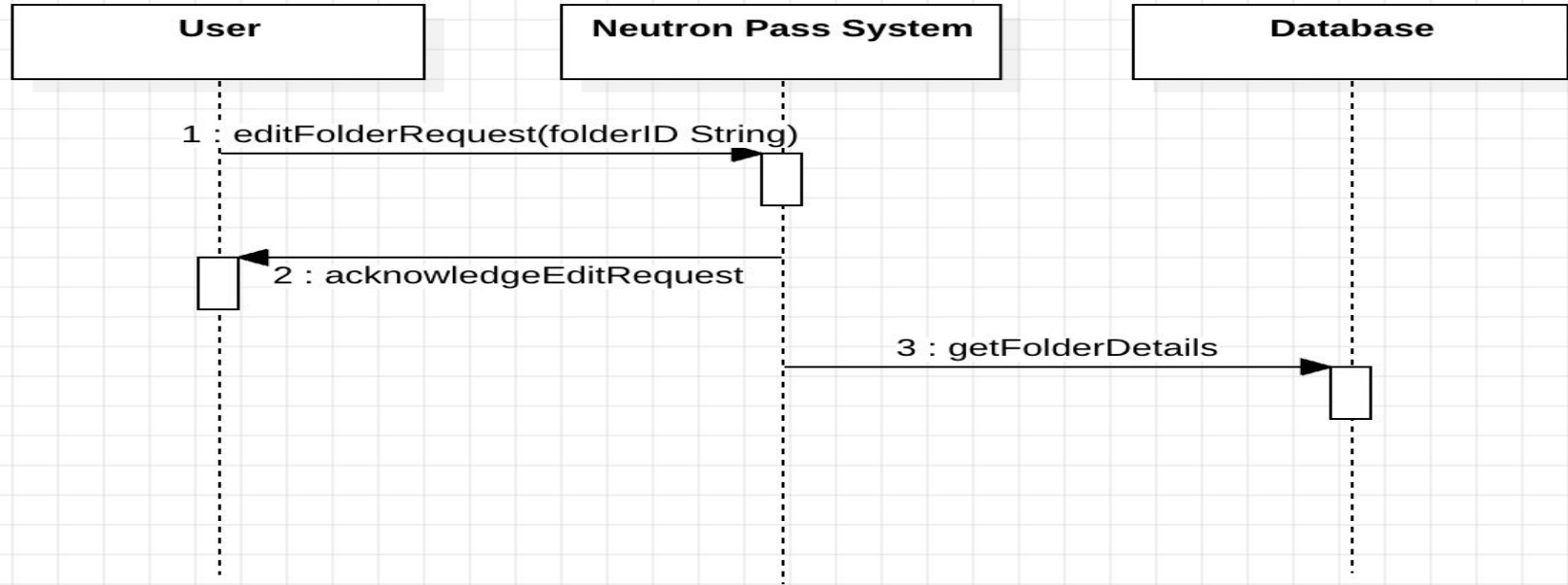
User Enters New Master Password

Edit Master Password Design Sequence Diagrams



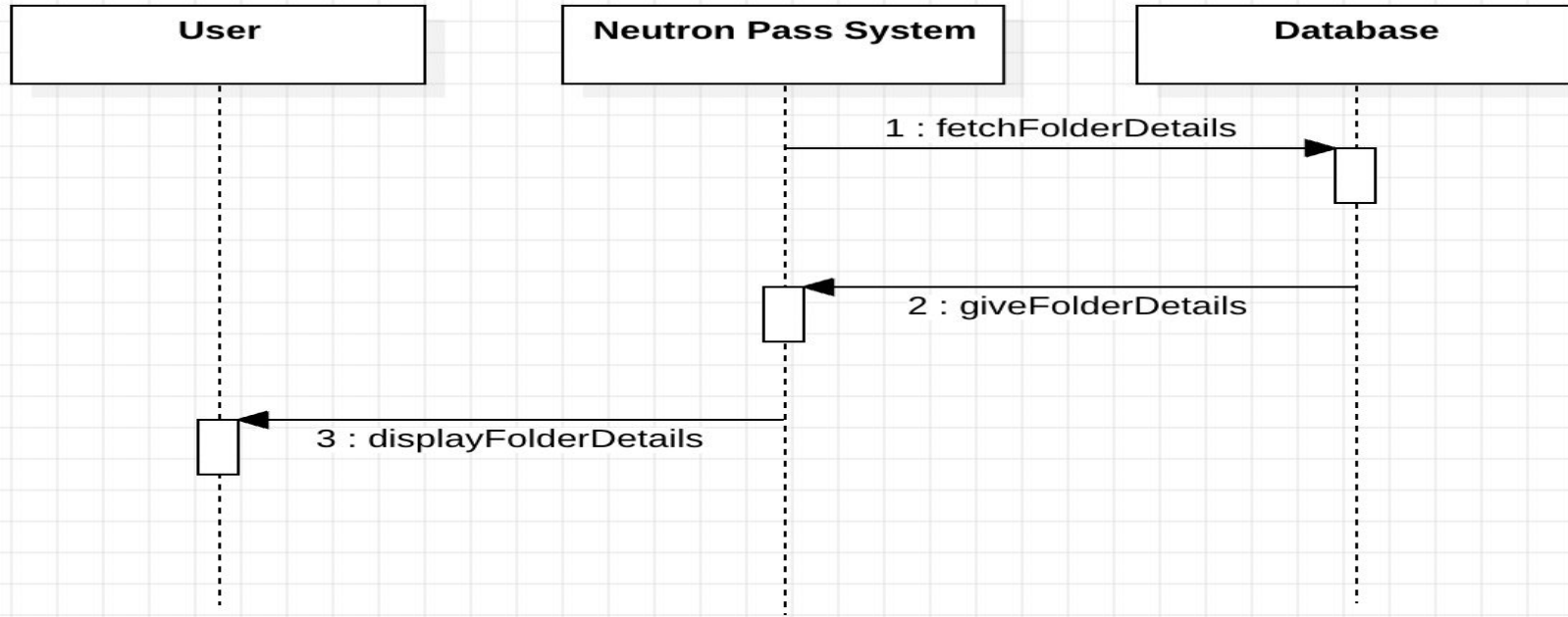
System Verifies And Updates New Master Password

Edit Stored Passwords Design Sequence Diagrams



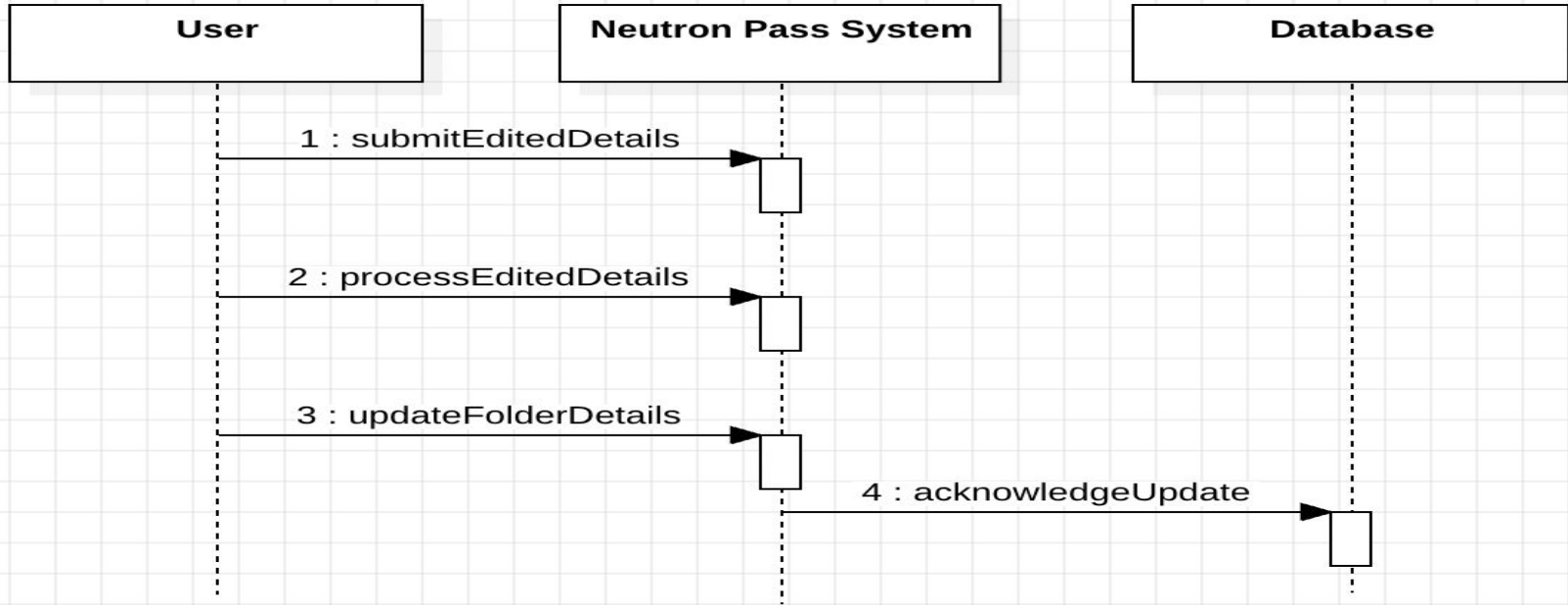
Request To Edit Folder

Edit Stored Passwords Design Sequence Diagrams



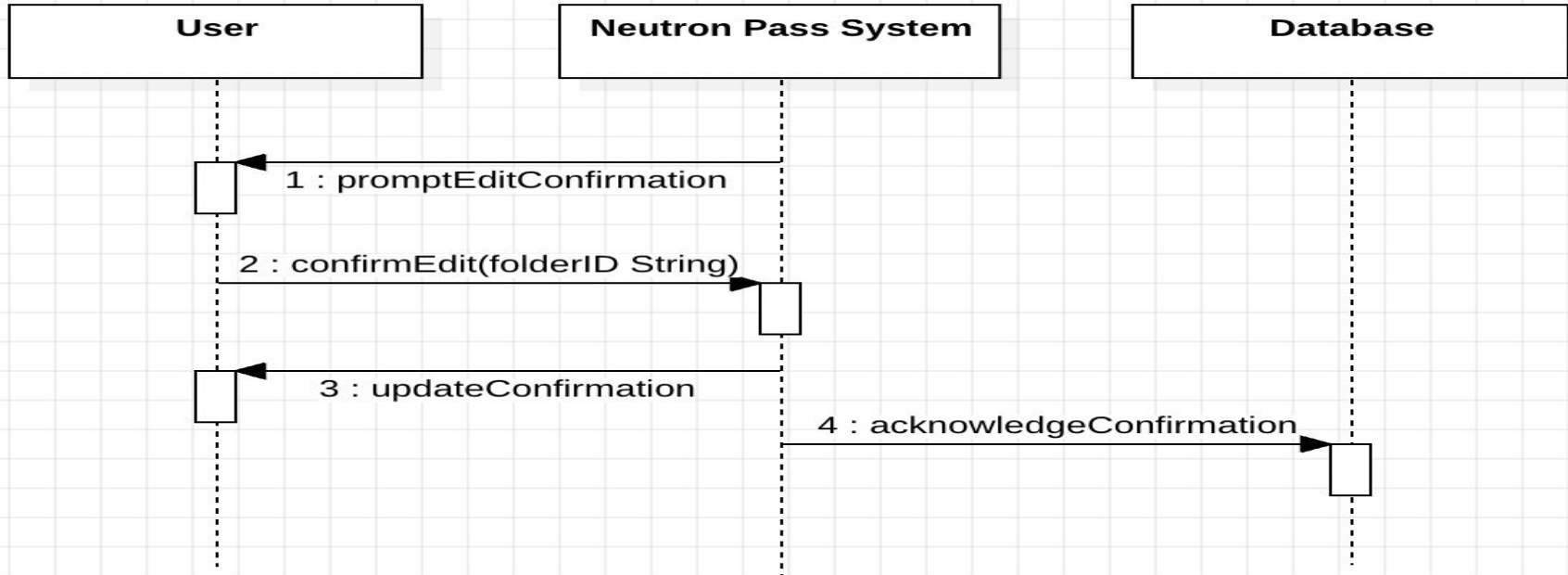
Display Folder Details

Edit Stored Passwords Design Sequence Diagrams



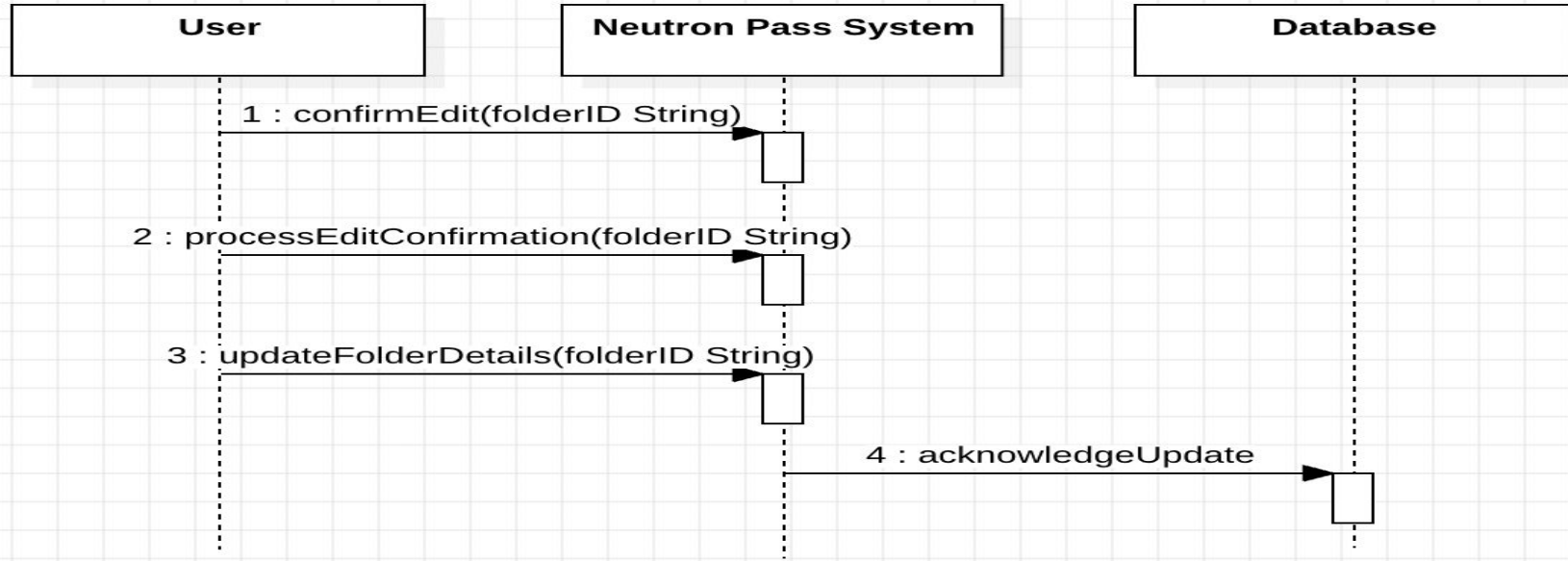
Edit Folder Details

Edit Stored Passwords Design Sequence Diagrams



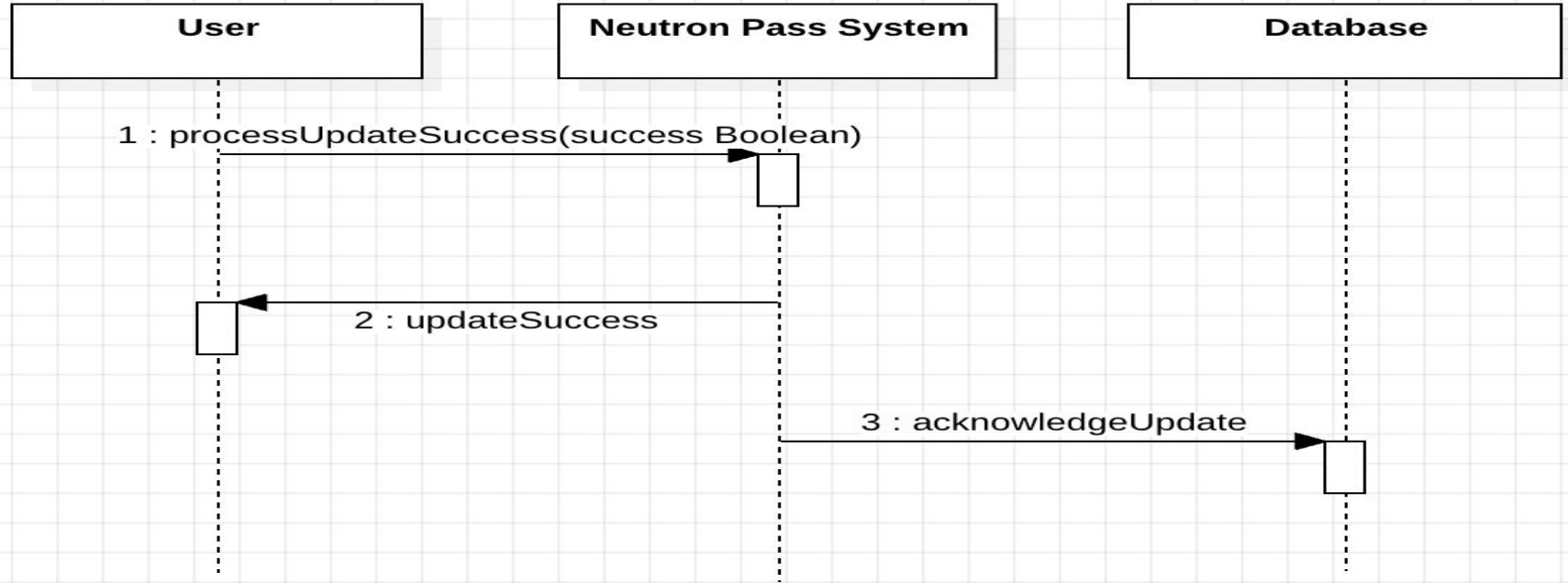
Prompt For Confirmation

Edit Stored Passwords Design Sequence Diagrams



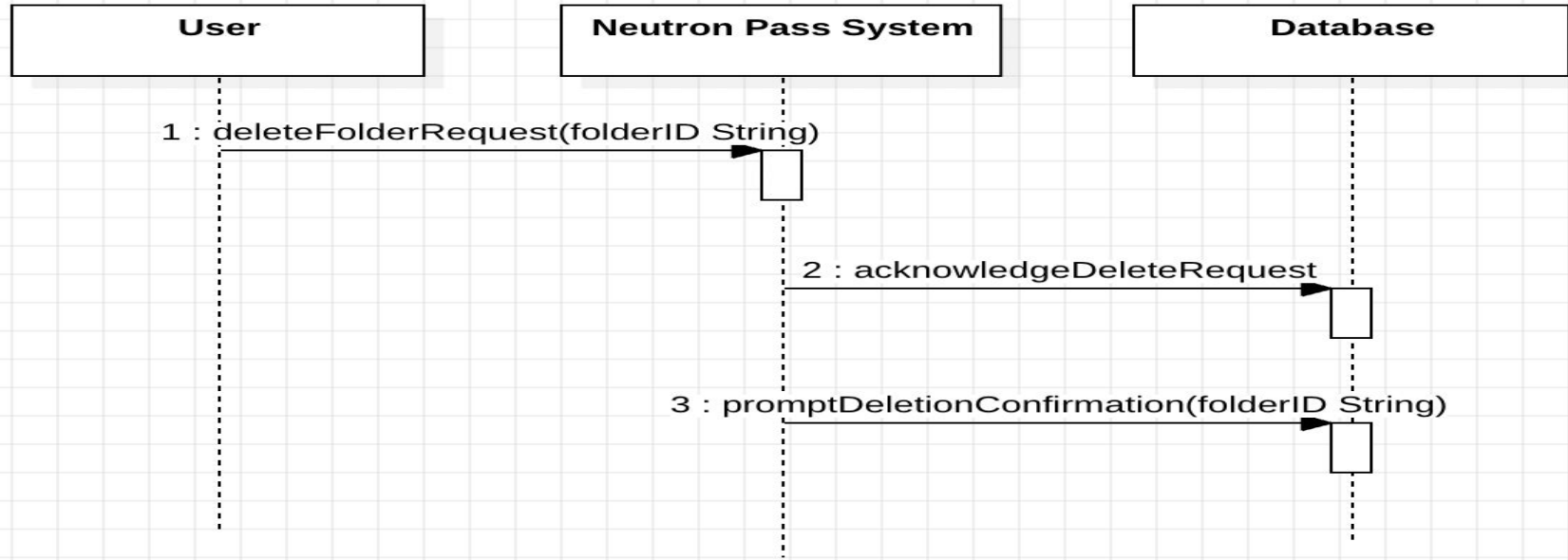
Confirm Changes

Edit Stored Passwords Design Sequence Diagrams



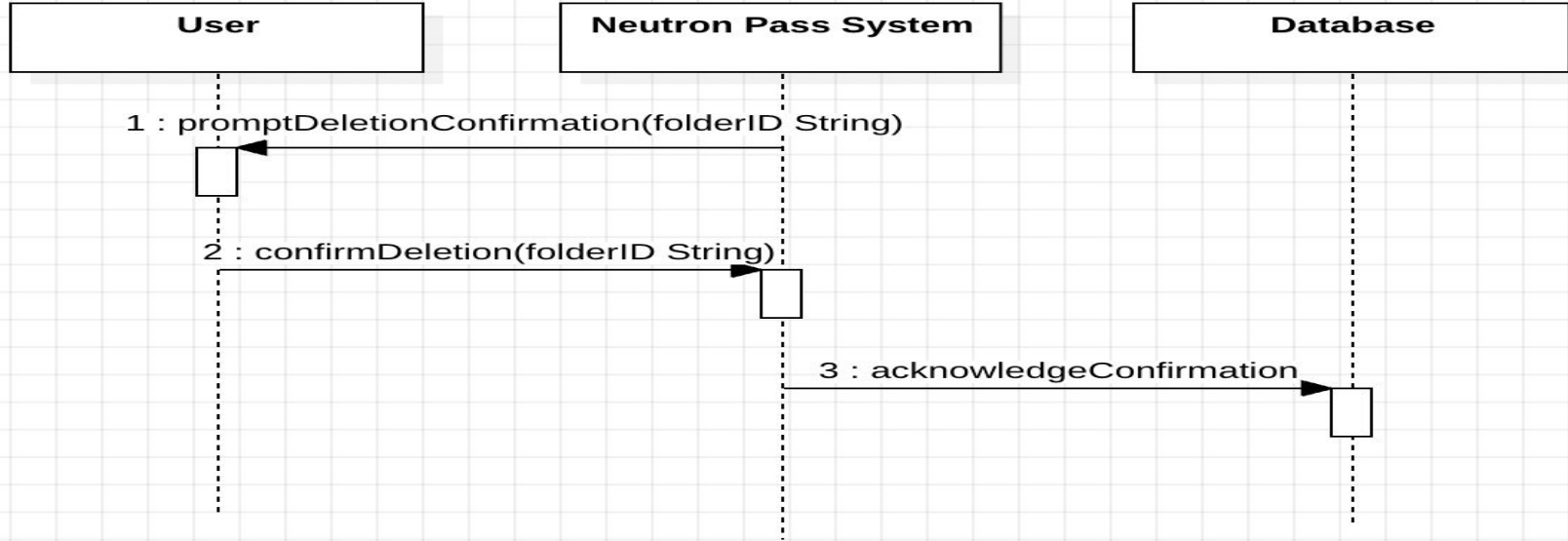
Confirm Update Success

Delete Passwords Design Sequence Diagrams



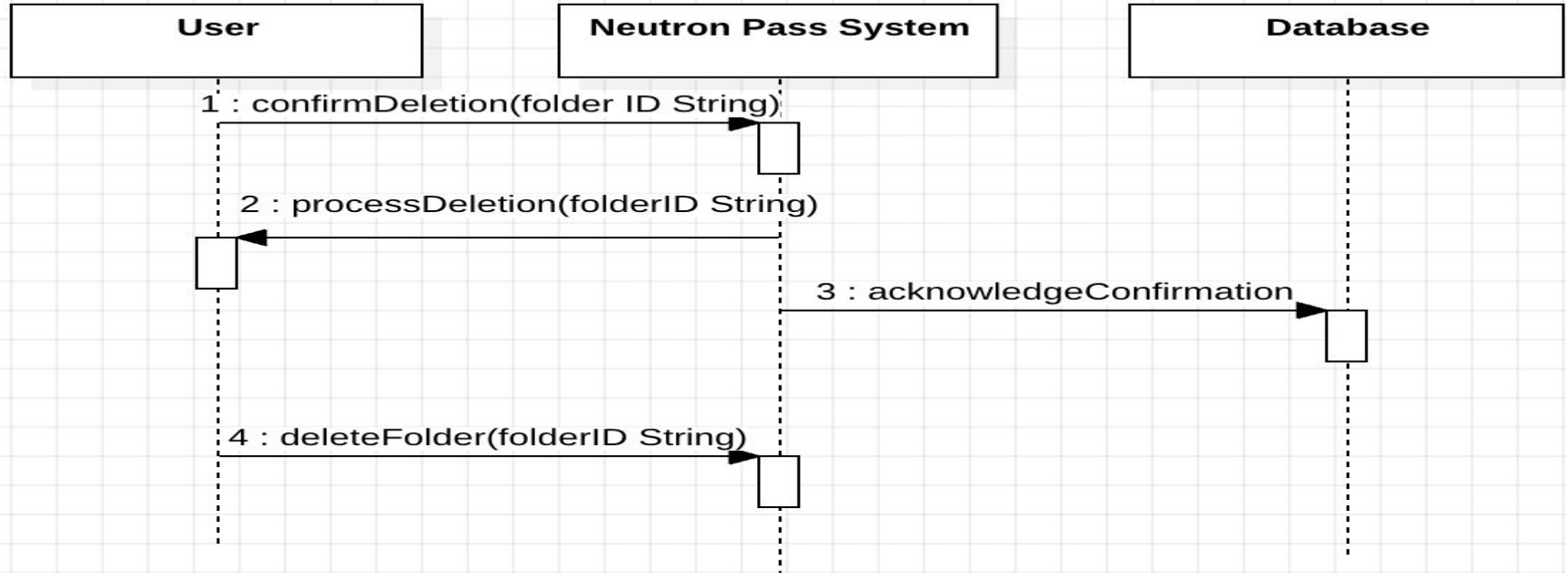
Request To Delete Folder

Delete Passwords Design Sequence Diagrams



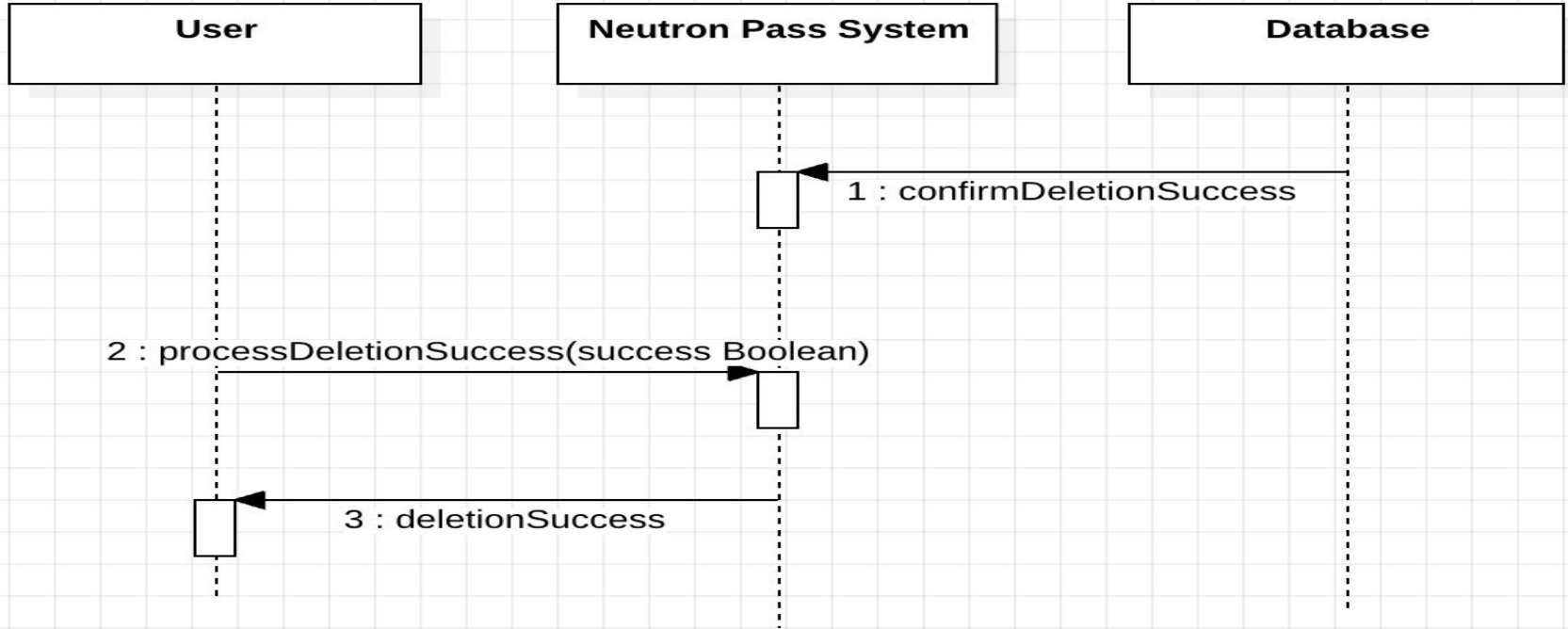
Prompt For Confirmation

Delete Passwords Design Sequence Diagrams



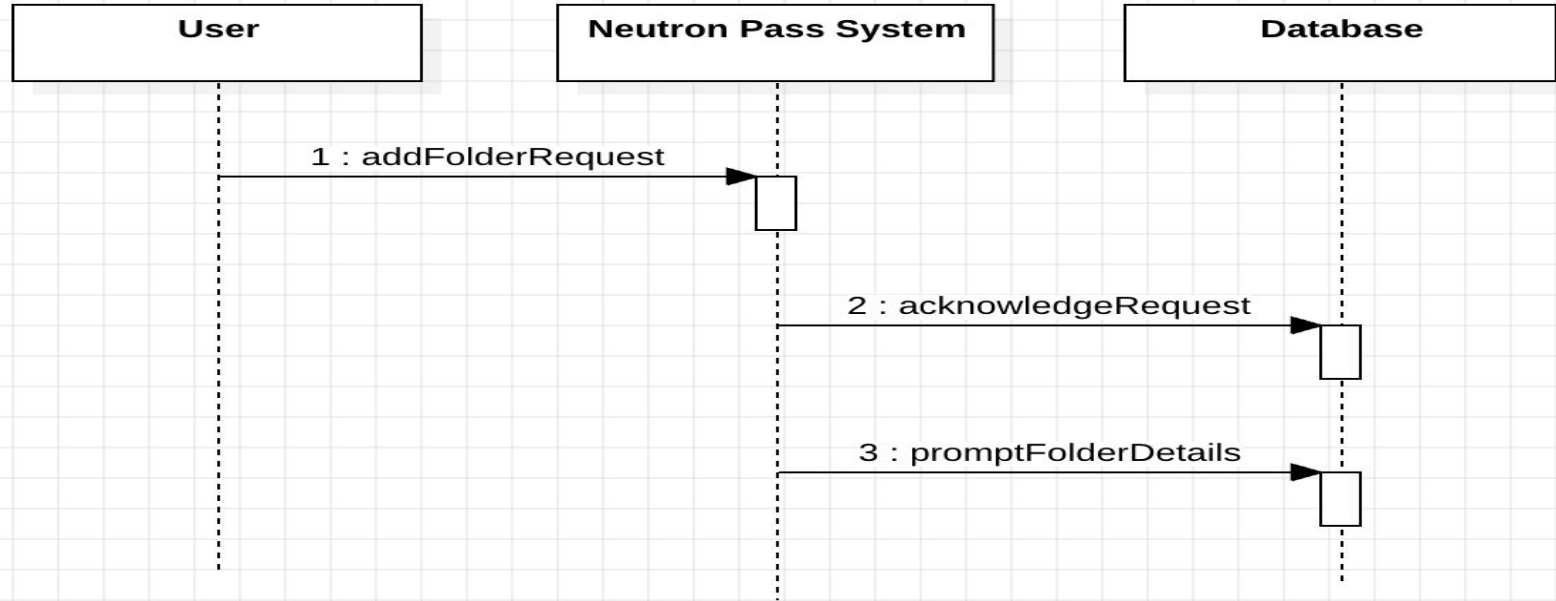
Confirm Deletion

Delete Passwords Design Sequence Diagrams



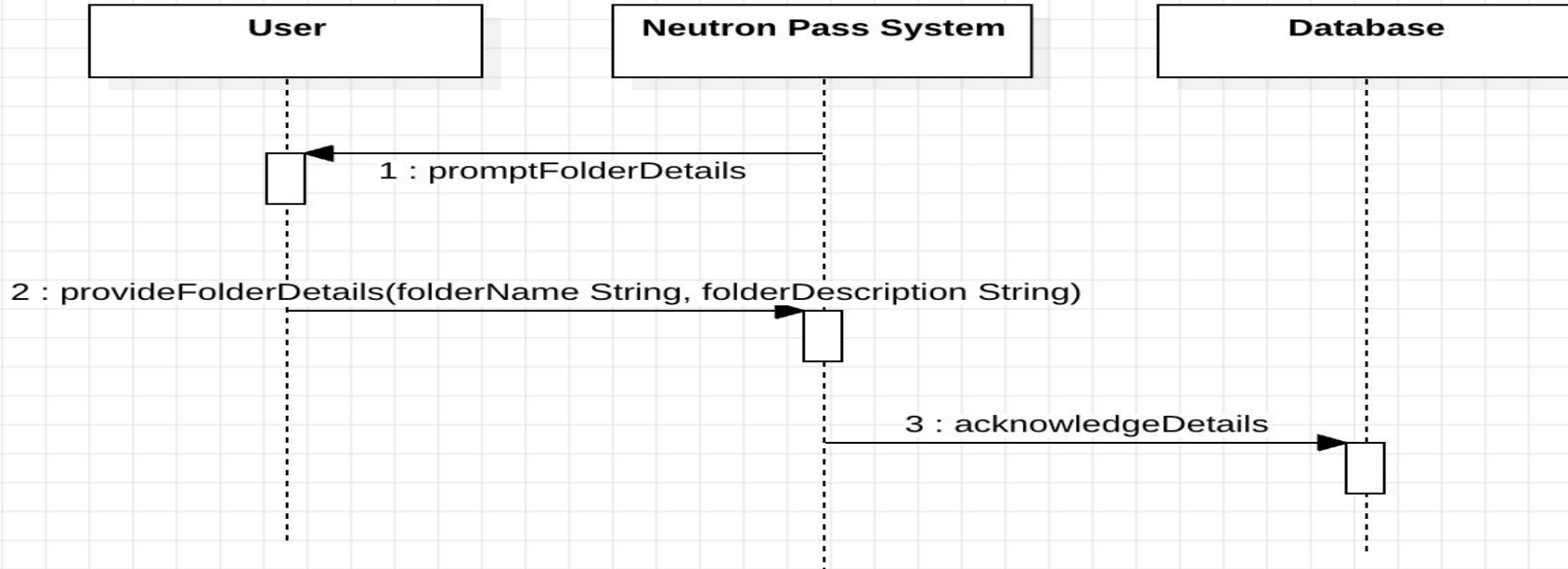
Confirm Deletion Success

Add Passwords Design Sequence Diagrams



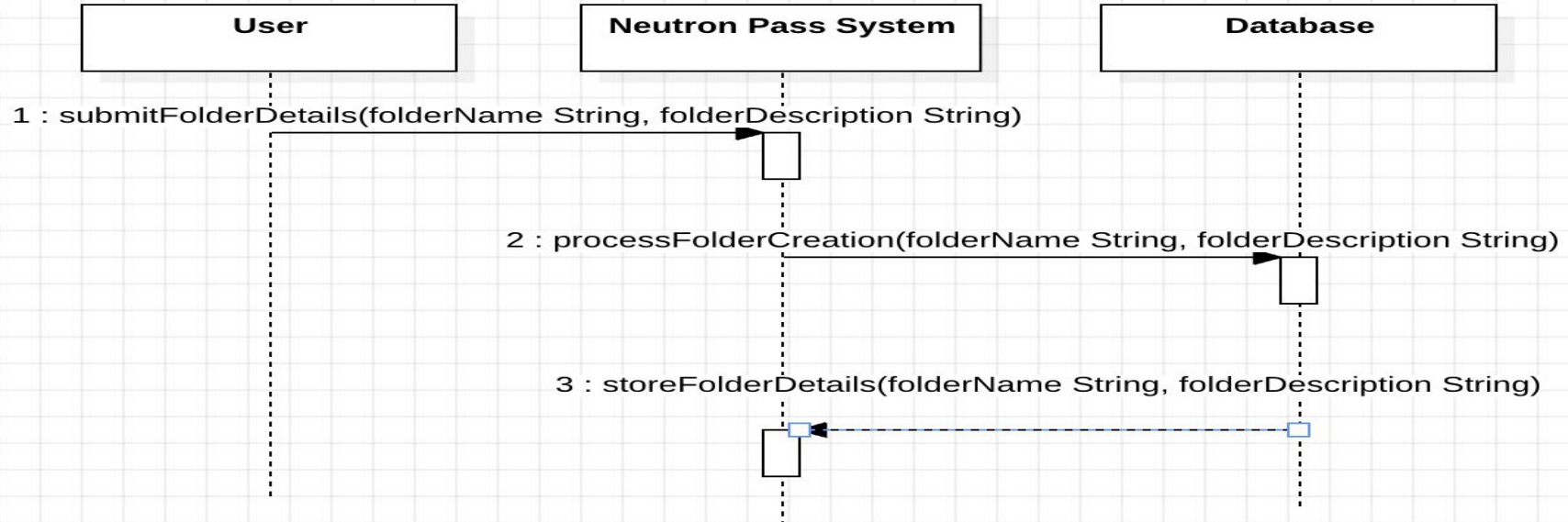
Request To Add Folder

Add Passwords Design Sequence Diagrams



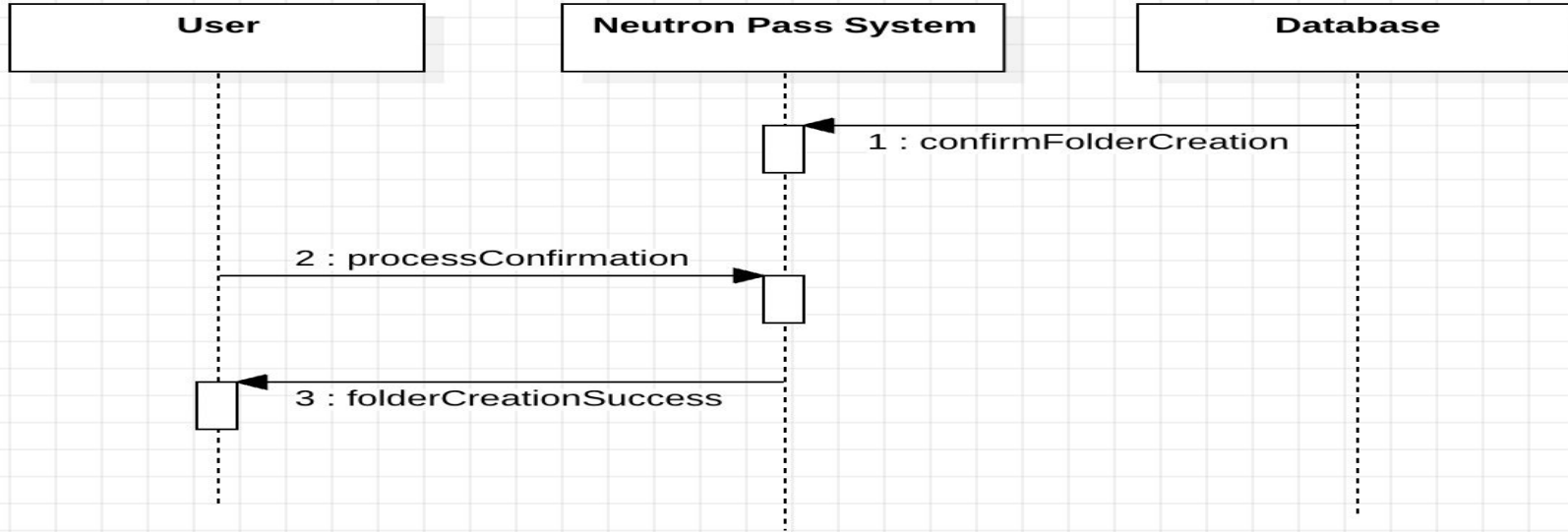
Prompt For Folder Details

Add Passwords Design Sequence Diagrams



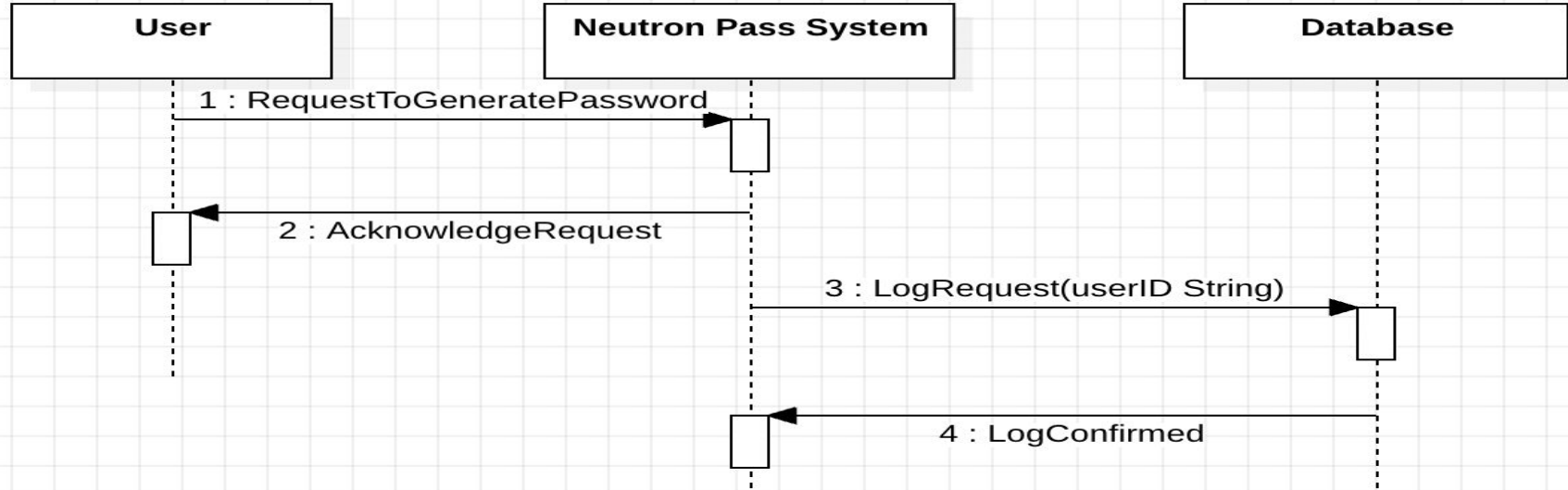
Submit Folder Details

Add Passwords Design Sequence Diagrams



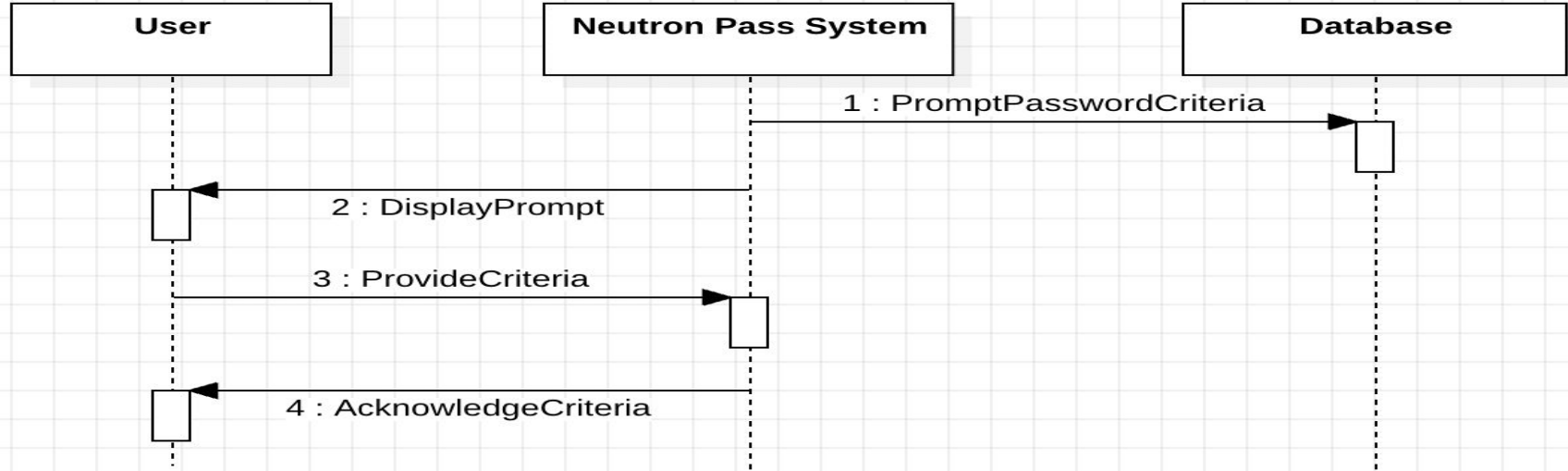
Confirm Folder Creation

Generate A Password Design Sequence Diagrams



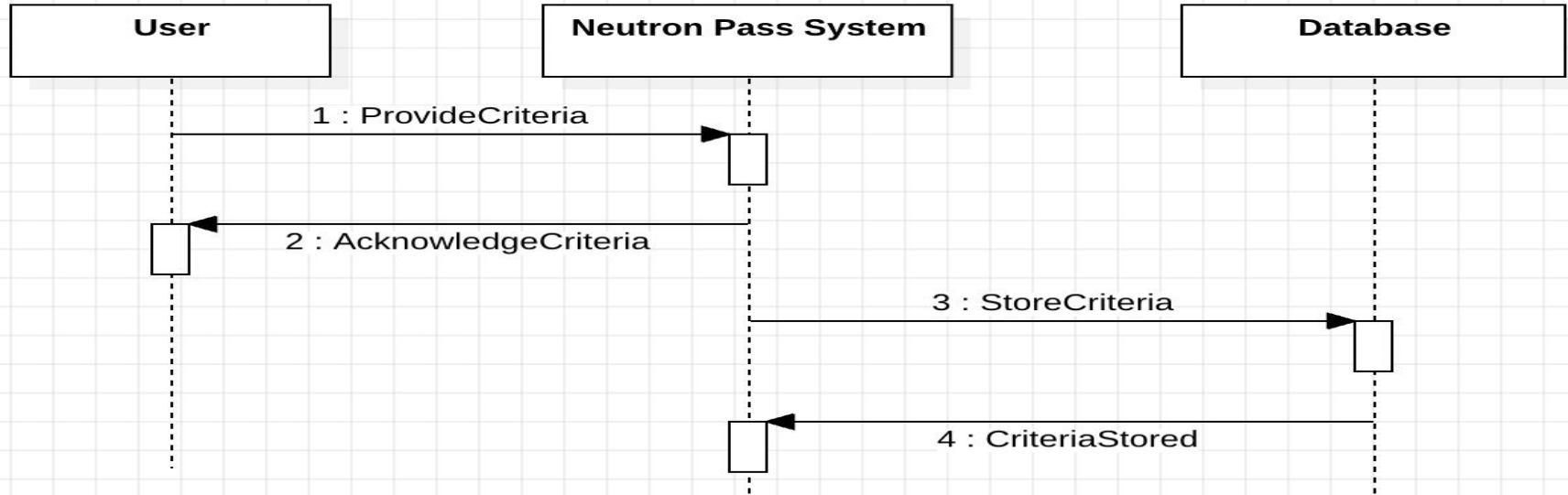
Request To Generate A New Password

Generate A Password Design Sequence Diagrams



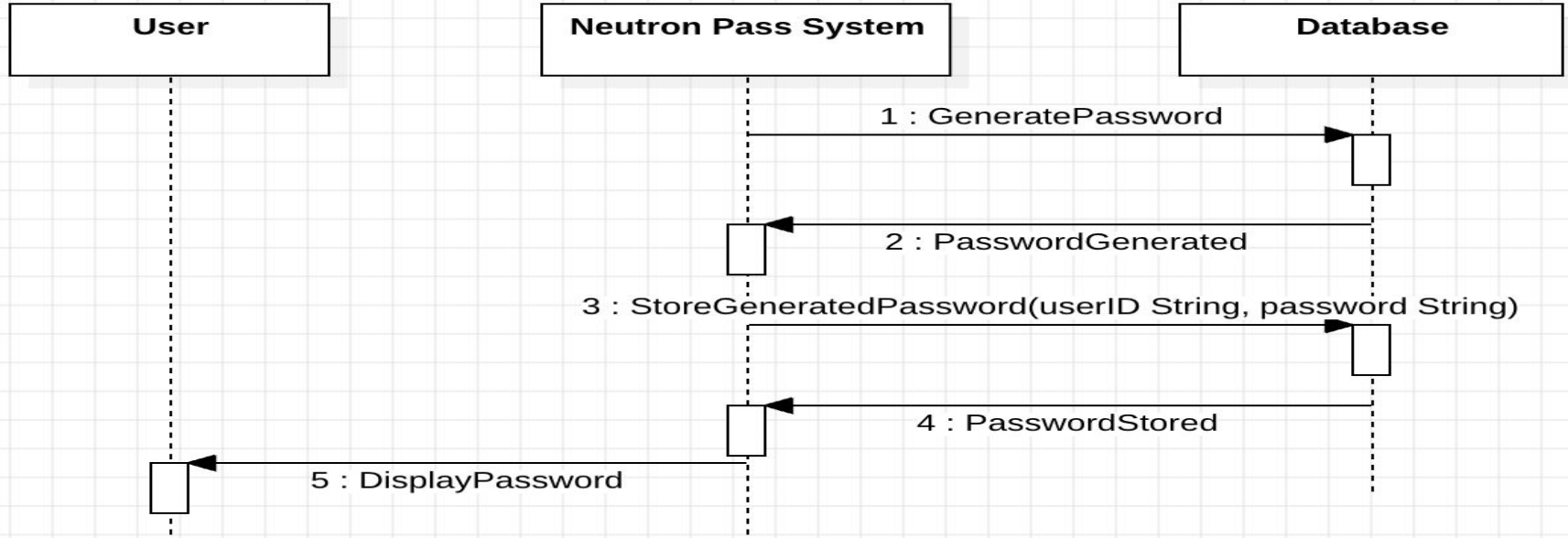
Prompt For New Password Criteria

Generate A Password Design Sequence Diagrams



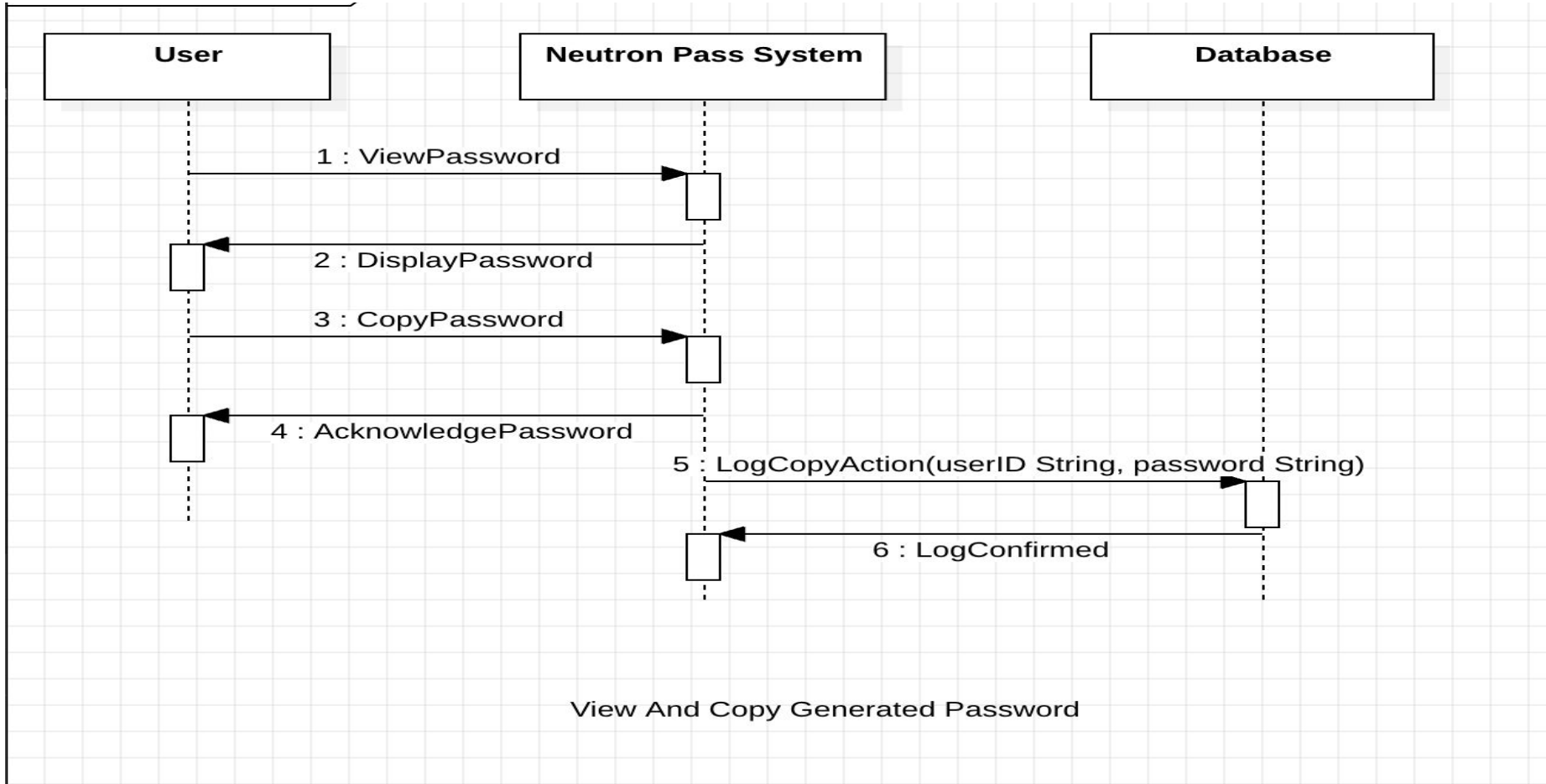
Provide Password Criteria

Generate A Password Design Sequence Diagrams



Display Generated Password

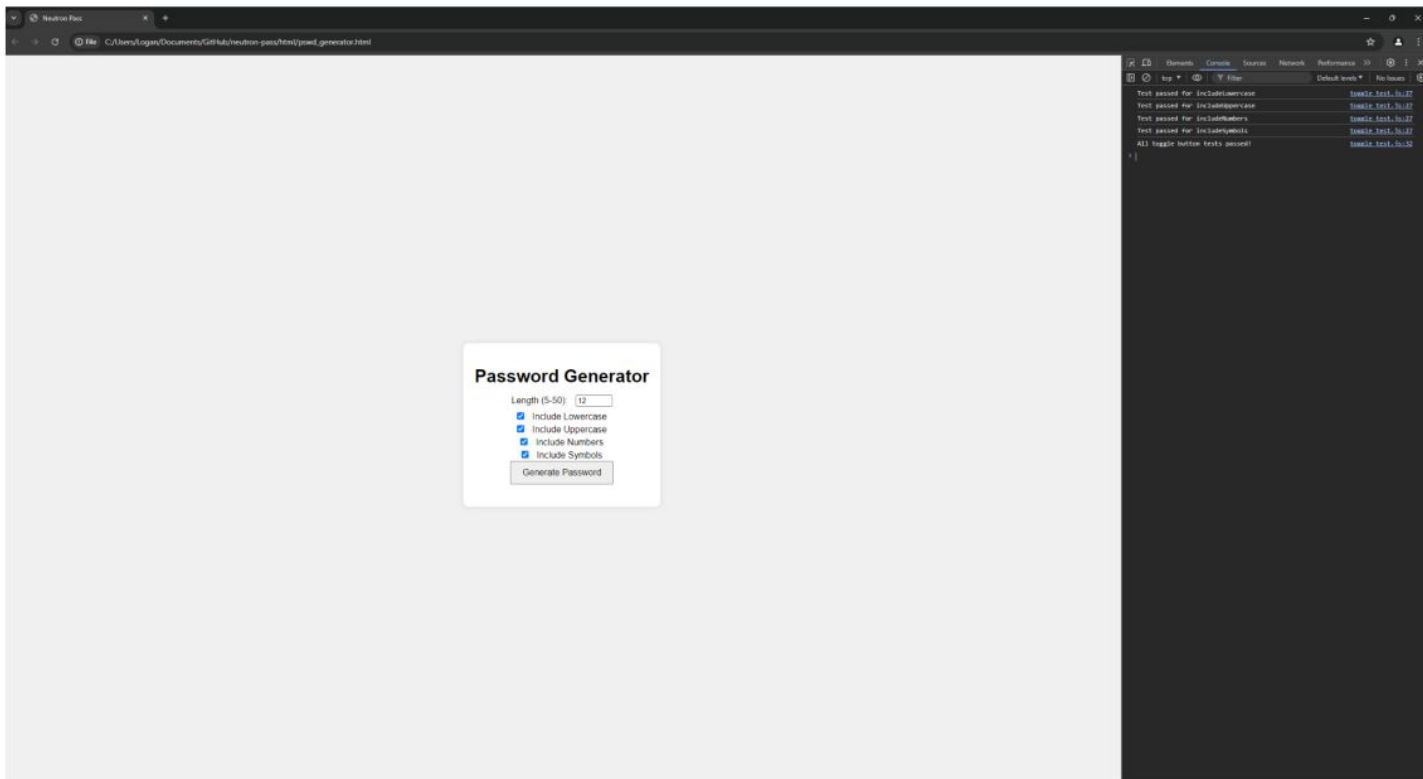
Generate A Password Design Sequence Diagrams



Phase 4

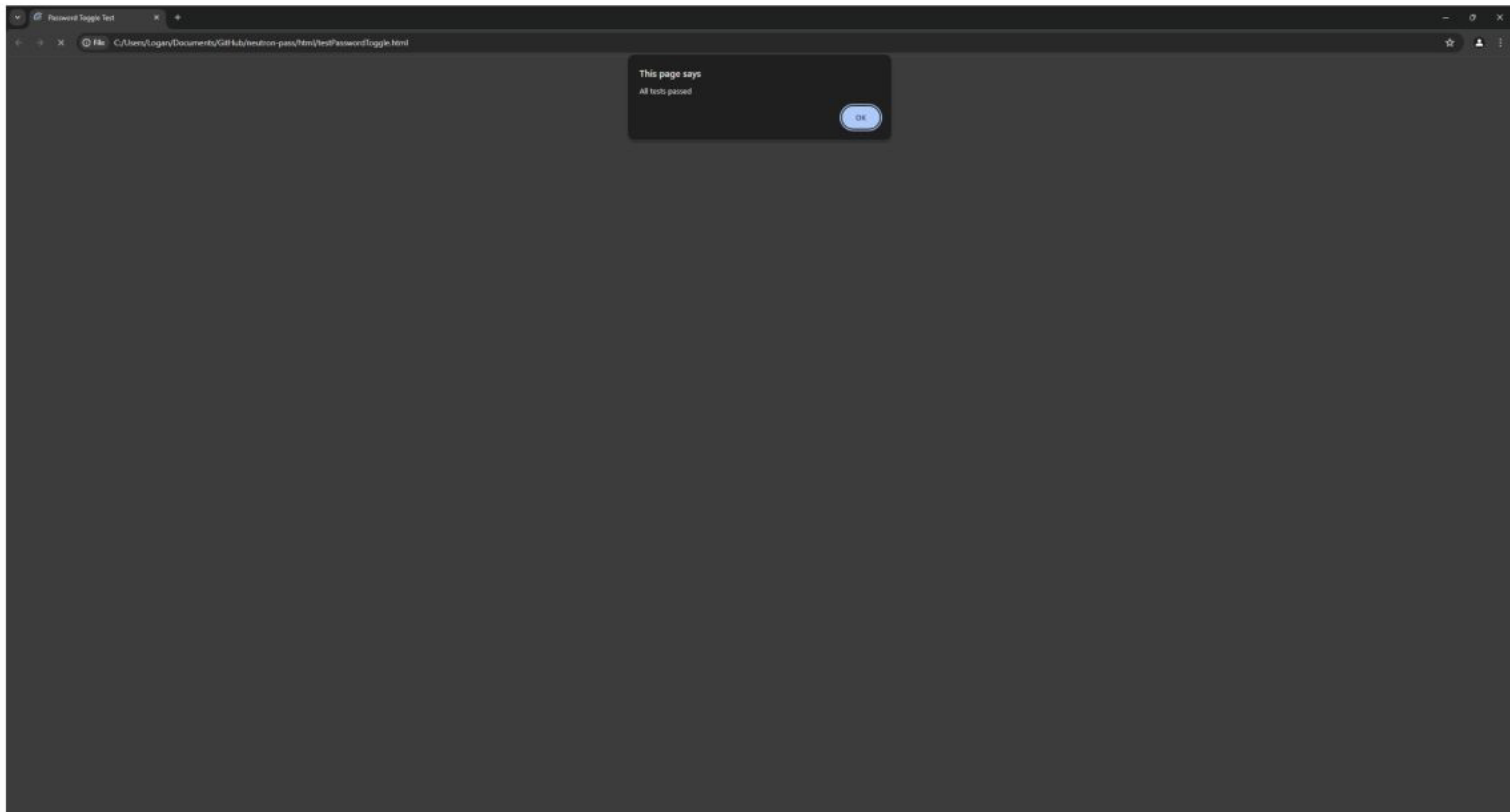
Testing

generateBtn



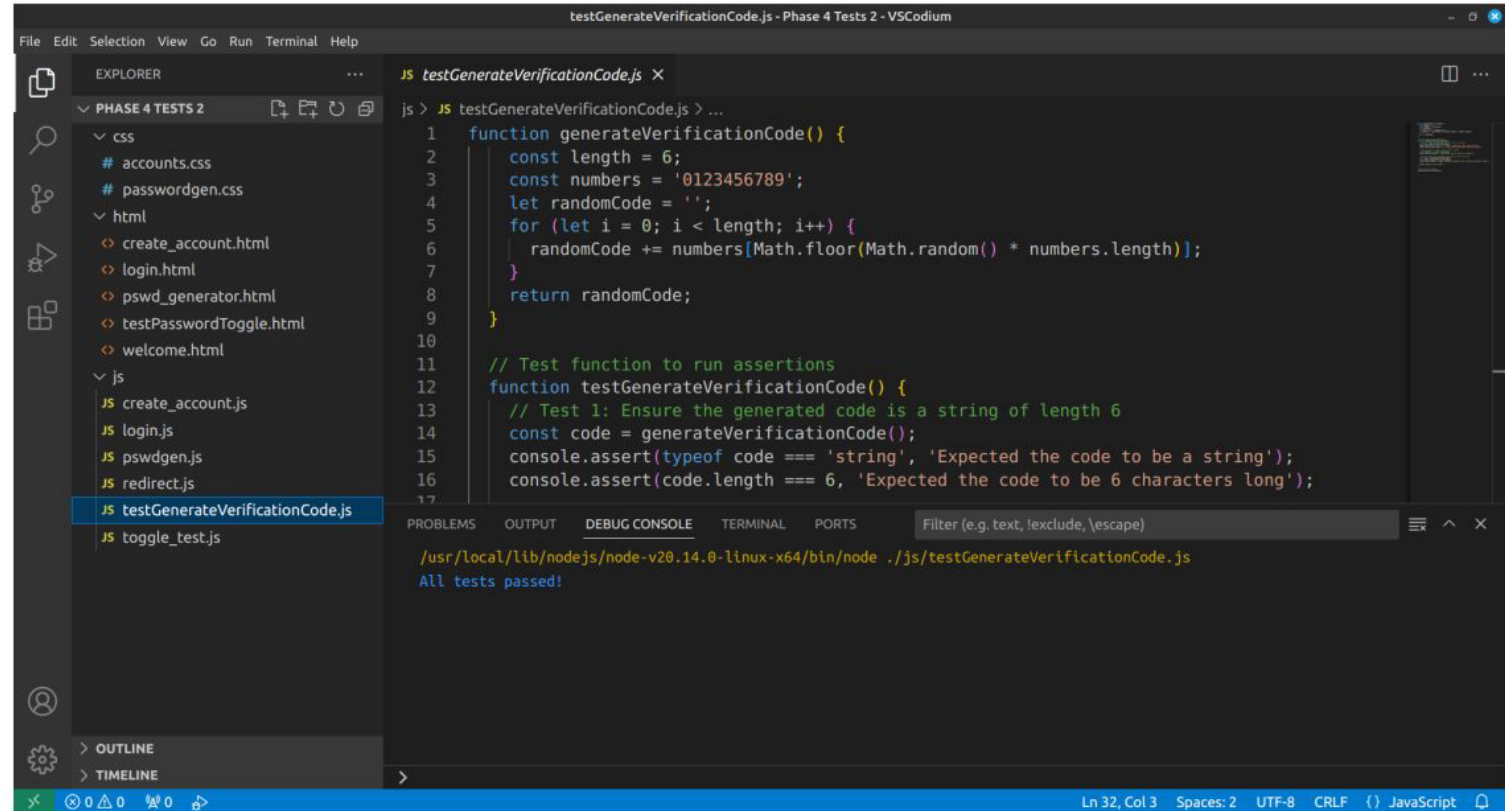
Open pswd_generator.html
Open inspect element (ctrl + shift + i)
Navigate to the console
Test result should be displayed

showPasswordButton



Open testPasswordToggle.html
Test should be run and given confirmation message

generateVerificationCode - This function we didn't embed in a html file and just used the node.js runtime. A browser runtime with a script onload of testGenerateVerificationCode() would work just fine as well.



The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays a project structure with folders 'css', 'html', and 'js'. The 'js' folder is expanded, showing several files including 'testGenerateVerificationCode.js', which is currently selected. The main editor area displays the code for 'testGenerateVerificationCode.js'. The code defines a 'generateVerificationCode()' function that creates a random 6-digit code from a set of numbers. It also includes a 'testGenerateVerificationCode()' function with two assertions: one to check if the code is a string and another to check if its length is 6. The bottom panel shows the 'DEBUG CONSOLE' tab with the command to run the file and the output 'All tests passed!'. The status bar at the bottom indicates the current position is Line 32, Column 3, with 2 spaces, in UTF-8 encoding, with CRLF line endings, and the file is a JavaScript file.

```
testGenerateVerificationCode.js - Phase 4 Tests 2 - VSCodeium
File Edit Selection View Go Run Terminal Help
EXPLORER
PHASE 4 TESTS 2
  css
    # accounts.css
    # passwordgen.css
  html
    create_account.html
    login.html
    pswd_generator.html
    testPasswordToggle.html
    welcome.html
  js
    create_account.js
    login.js
    pswdgen.js
    redirect.js
    testGenerateVerificationCode.js
    toggle_test.js

JS testGenerateVerificationCode.js
1 function generateVerificationCode() {
2   const length = 6;
3   const numbers = '0123456789';
4   let randomCode = '';
5   for (let i = 0; i < length; i++) {
6     randomCode += numbers[Math.floor(Math.random() * numbers.length)];
7   }
8   return randomCode;
9 }
10
11 // Test function to run assertions
12 function testGenerateVerificationCode() {
13   // Test 1: Ensure the generated code is a string of length 6
14   const code = generateVerificationCode();
15   console.assert(typeof code === 'string', 'Expected the code to be a string');
16   console.assert(code.length === 6, 'Expected the code to be 6 characters long');
17 }
18
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Filter (e.g. text, \exclude, \escape)
/usr/local/lib/nodejs/node-v20.14.0-linux-x64/bin/node ./js/testGenerateVerificationCode.js
All tests passed!
Ln 32, Col 3 Spaces: 2 UTF-8 CRLF {} JavaScript
```

Within the IDE navigate to the js folder
Run testGenerateVerificationCode.js
Test result should be displayed in the console

Demo

References

Hyett, P. Wanstrath, C. Preston-Werner, T. (2024) Microsoft. GitHub. <https://github.com>

AndersLindman. (June 3, 2024).
<https://github.com/AndersLindman/SHA256/blob/254e29ca2fb2d0697ced7e4fea03149e210ce8b9/js/sha256.js>

(June 7, 2024). <https://www.npmjs.com/package/body-parser>

(June 7, 2024). <https://www.npmjs.com/package/cors>

(June 7, 2024). <https://www.npmjs.com/package/express>

(June 7, 2024). <https://www.npmjs.com/package/nodemailer>

Pryor, M. Spolsky J. (2024). Atlassian. Trello. <https://trello.com>