High Level Design

Esin Sari

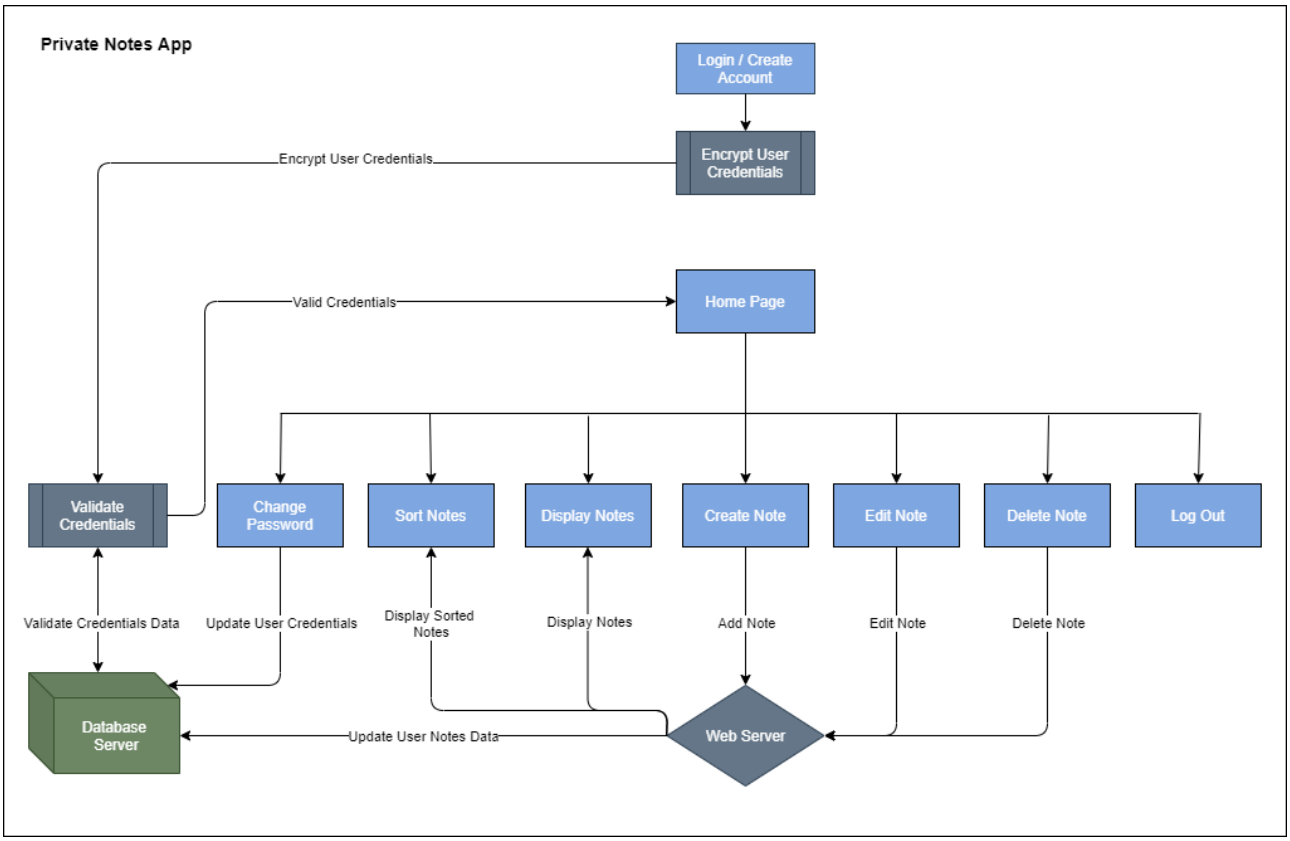
Project 3: Private Notes

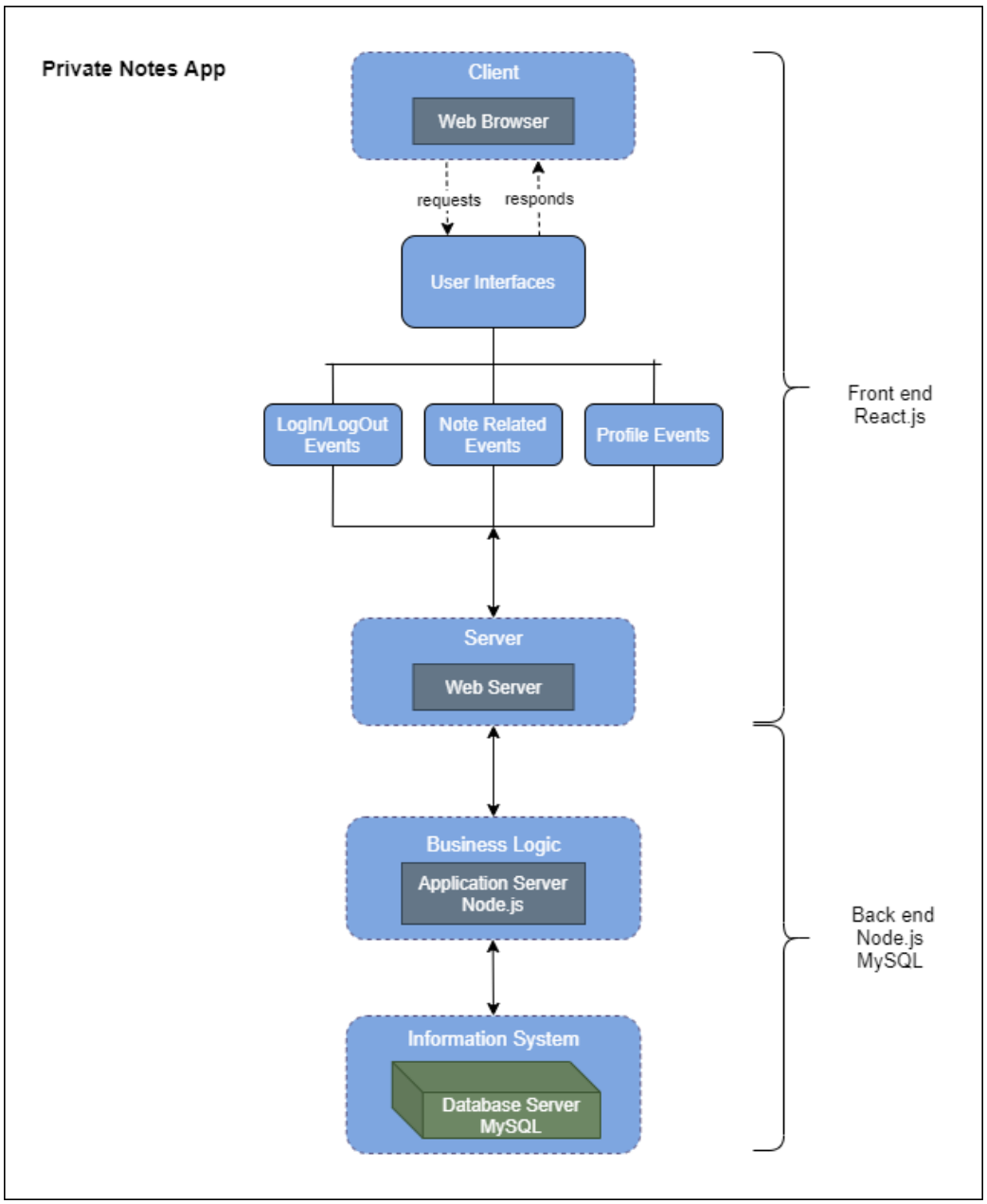
COP4331, Fall, 2020

Contents of this Document

High-Level Architecture
Design Issues

High-level Architecture





System Architectural Style

For Private Notes software system, client-server architectural design style is used to produce the desired result. In provided diagram, system has server components <database and web server> offer services and clients which access offered services using a request/reply protocol. Due to its applicable aspect to web applications, client-server architectural style is preferred to design system.

Major Components	Component Descriptions
Web Server	○ Used to respond to client requests made over the World Wide Web ○ Interact with components like Create Note, Display Notes, Sort Notes, etc. and Database server to generate successful responds to client requests
Database Server	○ Used to store and manage databases that are stored on the server and to provide data access for authorized users. ○ Validates user credentials ○ Logs the user in if credentials are verified ○ Prompts the user to create account ○ Update any changes happened in data
Log In / Create Account	○ First presented UI component when opening Private Notes web application ○ Required to enter credentials and click 'Login' button ○ Interface passes input to Encrypt User Credentials process and interact with database server ○ Database Server validate user credentials <username and password> ○ After verifying credentials, allow user to access home page
Home Page	○ UI component where user access his/her notes and other functionalities ○ Displays created notes and provides sorting options ○ Users are able to create or edit notes with using equivalent buttons ○ There would be a button which supports log out evet
Change Password	○ User Interface to change password information ○ There would be an equivalent button in the user home page, which leads to another screen ○ Requires user to enter current and new password ○ Interface interact with database server to update new credentials successfully
Display Notes	○ Component embedded in home page ○ Displays created notes with their titles and each note has link which gives access to another screen where user can only see selected note and perform desired actions like editing, deleting, etc. ○ Interact with web server to direct user requests
Create Note	○ User Interface to create a new note and implemented directly on home page ○ There would be an equivalent button for user to perform action ○ 'Create Note' button has link to another screen where user type new note with title, category, and additional information. ○ User can click on 'Save Note' button to store new note object in database ○ Interact with web server to direct user requests
Edit Note	○ User Interface to edit existing note ○ User can click on desired note which has link to another screen where user can only see selected note and perform desired actions like editing, deleting, etc.

	○ User can use 'Edit Note' button to perform action ○ Changes can be performed on any data related with note object (content, title, category, etc.) ○ User can click on 'Save Note' button to store changes in database ○ Interface interacts with web server to direct user requests
Delete Note	○ User Interface to delete existing note ○ User can click on desired note which has link to another screen where user can only see selected note and perform desired action. ○ User can use 'Delete Note' button to delete selected note ○ After clicking on 'Delete Note' button, database will be updated ○ Interface interacts with web server to direct user requests
Sort Note	○ User Interface to sort notes and implemented directly on home page ○ User can use 'Sort by Title', 'Sort by Category', or 'Sort by Date' button to display notes in sorted way based on the selected option ○ Interact with web server to direct user requests
Log Out	○ Component embedded in home page ○ There would be an equivalent button for user to log out ○ 'Log Out' button is used to return to Log In / Create Account interface

---

Design Issues

Design Issues Relevance	Issue Descriptions
Reliability	○ Reliability issues will be minimized by enabling system to correctly perform its required functions under assumed conditions, which are specified in Software Requirements Specification.
Reusability	○ Designed system aims to have high cohesion and low coupling by lowering dependencies which also enables components to be more reusable and easier to modify. ○ Some reusable codes are used for system and necessary updates are implemented to adopt system components more general; thus, specific parts could be used for further projects. ○ Components are well-documented, and documents of system include information about faults found and fixed
Maintainability	○ Maintainability issues will mostly arise from the server side of this project.

	○ To minimize issues such as user interface, security, and performance related, system will make necessary updates on components.
Testability	○ System should be evaluated during all developmental phases of the project. ○ All interfaces should be built to be easily testable.
Performance	○ System performance should be evaluated during all developmental phases of the project. ○ Performance of Private Notes app should run error-free and very successfully. ○ System performance should address specified time and capacity in Software Requirements Specification document.
Portability	○ System is a web-app, so should have portability.
Security	○ System ensures that all security features are included in design and components handle possible errors successfully and not allowed unauthorized activities.
Safety	○ Safety of system will be evaluated by using fault-tree analysis which also used to determine which faults to correct/avoid/tolerate.

Prototyping

- Various prototypes evaluating different database solutions
- Various prototypes evaluating multiple UI design and layout schemes
- Various prototypes evaluating multiple request-respond mechanisms
- Various prototypes evaluating functionalities related with note events

Technical Difficulties

- Technical difficulty expected to be encountered is lack of programming knowledge about Web Application development. To solve these difficulties, programmer will learn new tools such as database management system, React.js, and Node.js, perform some research, and adjust new environments and programming language like JavaScript, SQL, etc.
- Technical difficulties can also arise while from combining front end, back end, and database parts. These technical difficulties will be solved by watching relevant tutorials on web applications.
- Another technical difficulty may arise from Git and GitHub version-control system due to lack of understanding of version control paradigm. These technical difficulties will be solved by practicing and adjusting during software development.

Design Trade-offs

- Due to advantages of JavaScript on achieving quality attributes, JavaScript is selected as the primary programming language of the system; it is also natively supported on Web Applications. JavaScript tends to run immediately within the client's browser and is not slowed down by calls to a backend server, which promotes system performance in terms of speed. Technical risks can include improper coding syntax and security; moreover, learning process will take time, which might generate problem on project schedule.
- React.js and Node.js were chosen due to web app development simplicity, high performance, rich interfaces, and extended functionality. Same technical risks and solutions of JavaScript trade-off also apply to this trade-off.
- High Cohesion and low coupling methods will apply to this system, which enables system to be reusable and easier to modify and boost performance. To get desired solution from this trade-off, various UML patterns are created which address alternative designs to lower component and data dependencies.

Detailed Design

Esin Sari

Project 3: Private Notes

COP4331, Fall, 2020

Contents of this Document

Design Issues
Detailed Design Information
Trace of Requirements to Design

Detailed Design Issues

- **Reliability:** Reliability issues will be minimized by enabling system to correctly perform its required functions under assumed conditions, which are specified in Software Requirements Specification. As it is mentioned in SRS, system handles all possible faults caused by user; for instance, system has verification password process which checks credentials and interact with database; in case of error, system will display login error and user cannot enter the home page. System also generates error messages when user tries to save note inappropriate format, such as notes without title. By applying all these, system tries to address and minimize reliability issues.
- **Reusability:** Designed system aims to have high cohesion and low coupling by lowering dependencies which also enables components to be more reusable and easier to modify. To lower dependencies, inheritance method will be used for User and Notes object. The reason is I realized that some functionalities are mostly related with wither note or user object, I grouped related ones under user class and note class.
  Reusable codes are used for system and necessary updates are implemented to adopt system components more general; thus, specific parts could be used for further projects. Hashing methods and login and create account components are adapted from another project; after modifying these components, they could be used for other projects which are specific for Web applications. Database schema and overall class diagram are specific to this project, so they do not have reusability aspect; however, components like login, logout, encrypt password, or change password can be reused in other projects. Again, user interfaces for main and login page could be reusable with a few changes. In addition to these, to increase reusability of application, all components are well-documented, and documents of system include information about faults found and fixed.
- **Maintainability:** Maintainability issues will mostly arise from the server side of this project. To minimize issues such as user interface, security, and performance related, system will make necessary updates on components. Any changes on components provided in class diagram may affect the other components; thus, multiple class diagrams should be designed, and decision should be made for the best version. The purpose here is adding functionality/modules without affecting other components. Also, version control systems should be used to compare previous and new versions to make better analysis on issues related maintainability.

- **Testability:** System should be evaluated during all developmental phases of the project. All interfaces should be built to be easily testable. Also, error handling be tested to make sure system catches unauthorize login activity and saving notes without title; to minimize related issues, both client and server side must interact well with database.
- **Performance:** System performance should be evaluated during all developmental phases of the project. Performance of Private Notes app should run error-free and very successfully. System performance should address specified time and capacity in Software Requirements Specification document. To minimize performance related issues, system is built with JavaScript programming language, which known as its high performance with web applications. Also, class diagram is designed in purpose of lowering dependencies, which has positive impacts on performance. Other performance related issues should be investigated during the testing phase.
- **Portability:** System is a web-app, so should have portability. Portability issues may arise between different devices, such as difference between desktop or mobile platforms. Thus, portability issues should be investigated during the testing phase.
- **Security:** System ensures that all security features are included in design and components handle possible errors successfully and not allowed unauthorized activities. Security related issues may mostly arise from authentication and encryption process. System should have effective hashing method which enables password difficult to decrypt by outside. Also, system must interact well with database in verification phase and when storing password changes.
- **Safety:** Safety of system will be evaluated by using fault-tree analysis which also used to determine which faults to correct/avoid/tolerate. However, expected safety related issues are lower for this project.

Prototyping

- Various prototypes evaluating different database solutions
- Various prototypes evaluating multiple UI design and layout schemes
- Various prototypes evaluating multiple request-respond mechanisms
- Various prototypes evaluating functionalities related with note events

Technical Difficulties

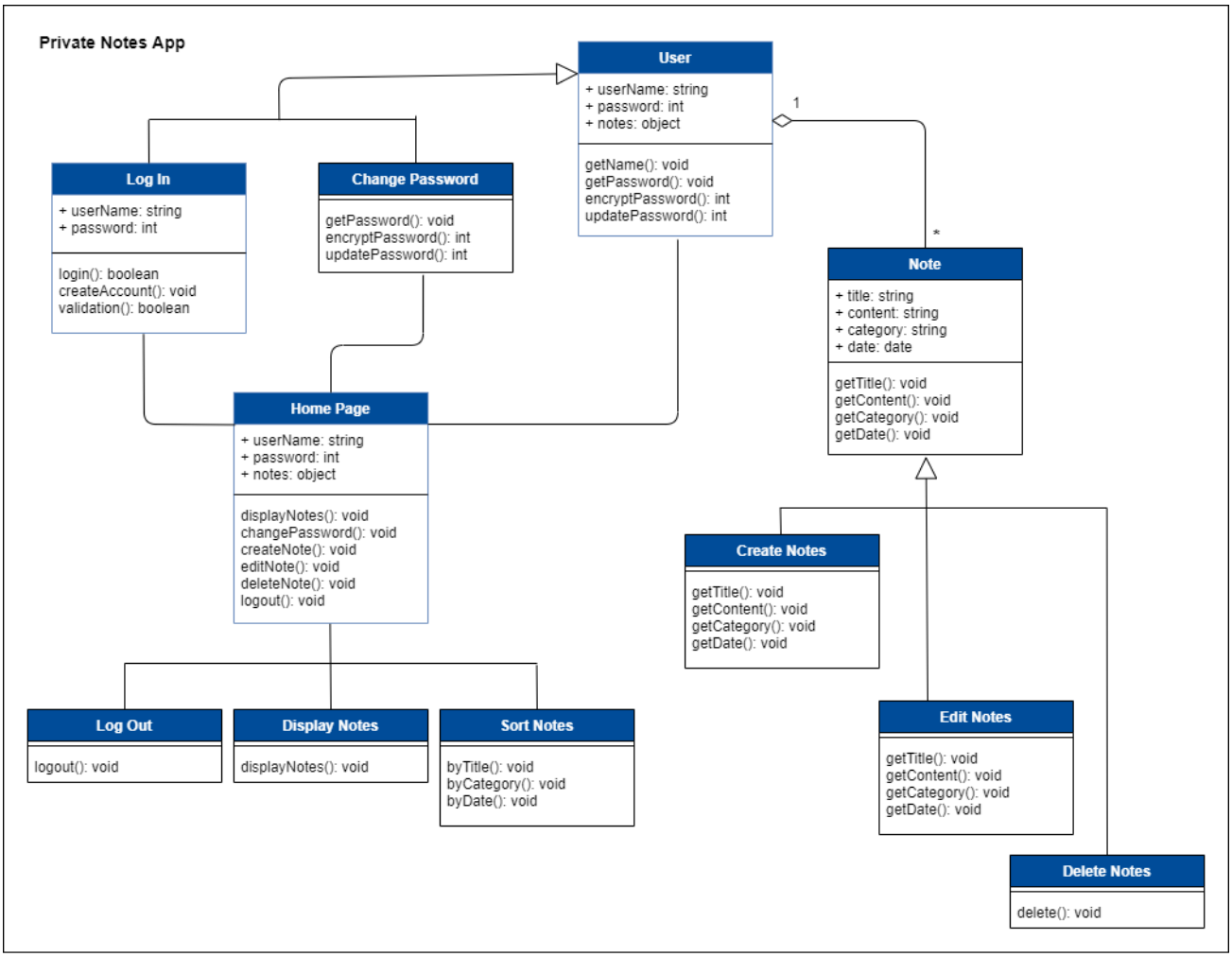
- Technical difficulty expected to be encountered is lack of programming knowledge about Web Application development. To solve these difficulties, programmer will learn new tools such as database management system, React.js, and Node.js, perform some research, and adjust new environments and programming language like JavaScript, SQL, etc.
- Technical difficulties can also arise while from combining front end, back end, and database parts. These technical difficulties will be solved by watching relevant tutorials on web applications.
- Another technical difficulty is security related. To maintain information security, each user should have unique username and password. Otherwise, system will not allow user to create account or login.

- Another technical difficulty may arise from Git and GitHub version-control system due to lack of understanding of version control paradigm. These technical difficulties will be solved by practicing and adjusting during software development.

Design Trade-offs

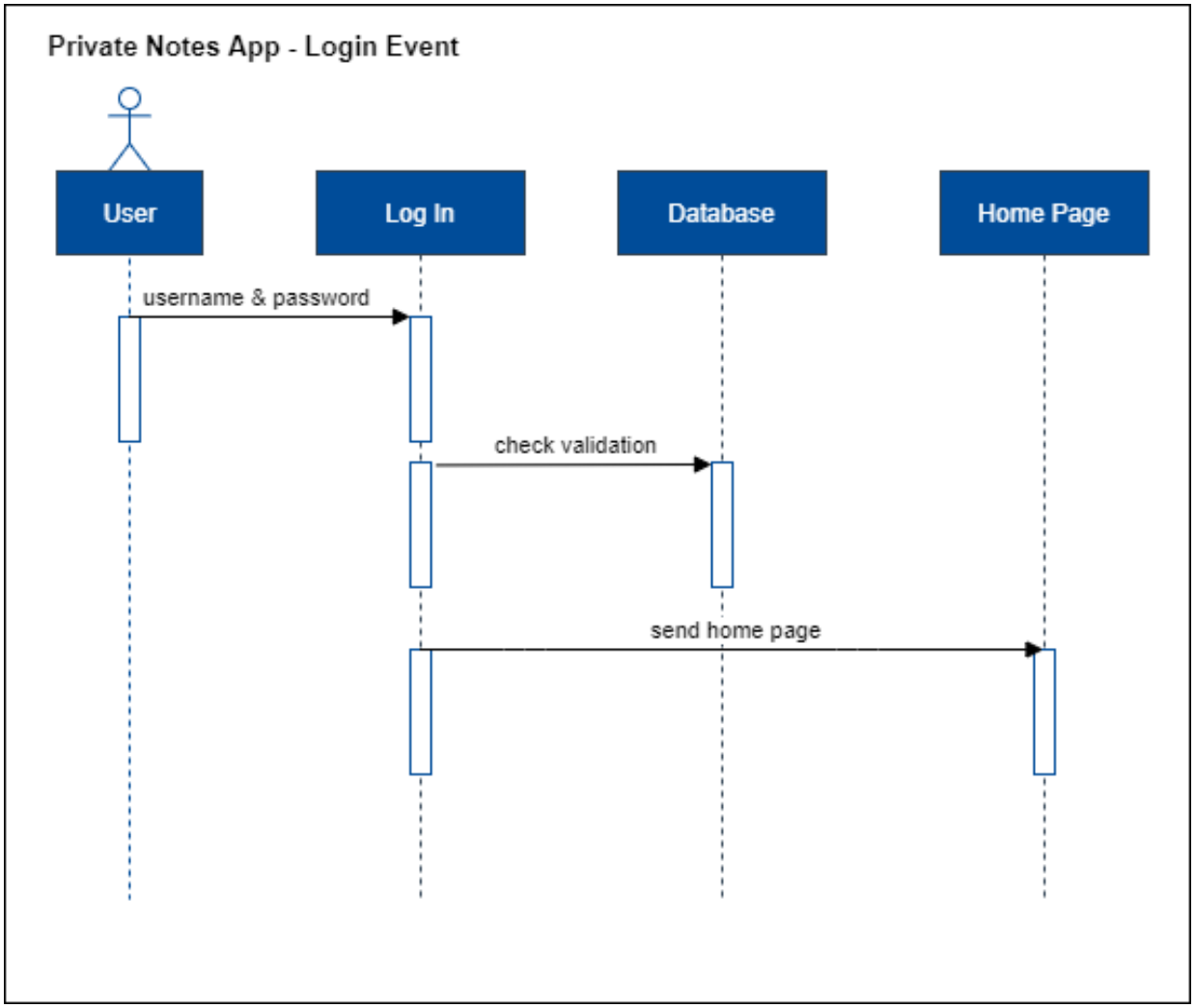
- Due to advantages of JavaScript on achieving quality attributes, JavaScript is selected as the primary programming language of the system; it is also natively supported on Web Applications. JavaScript tends to run immediately within the client's browser and is not slowed down by calls to a backend server, which promotes system performance in terms of speed. Technical risks can include improper coding syntax and security; moreover, learning process will take time, which might generate problem on project schedule.
- React.js and Node.js were chosen due to web app development simplicity, high performance, rich interfaces, and extended functionality. Same technical risks and solutions of JavaScript trade-off also apply to this trade-off.
- High Cohesion and low coupling methods will apply to this system, which enables system to be reusable and easier to modify and boost performance. To get desired solution from this trade-off, various UML patterns are created which address alternative designs to lower component and data dependencies.

Detailed Design Information



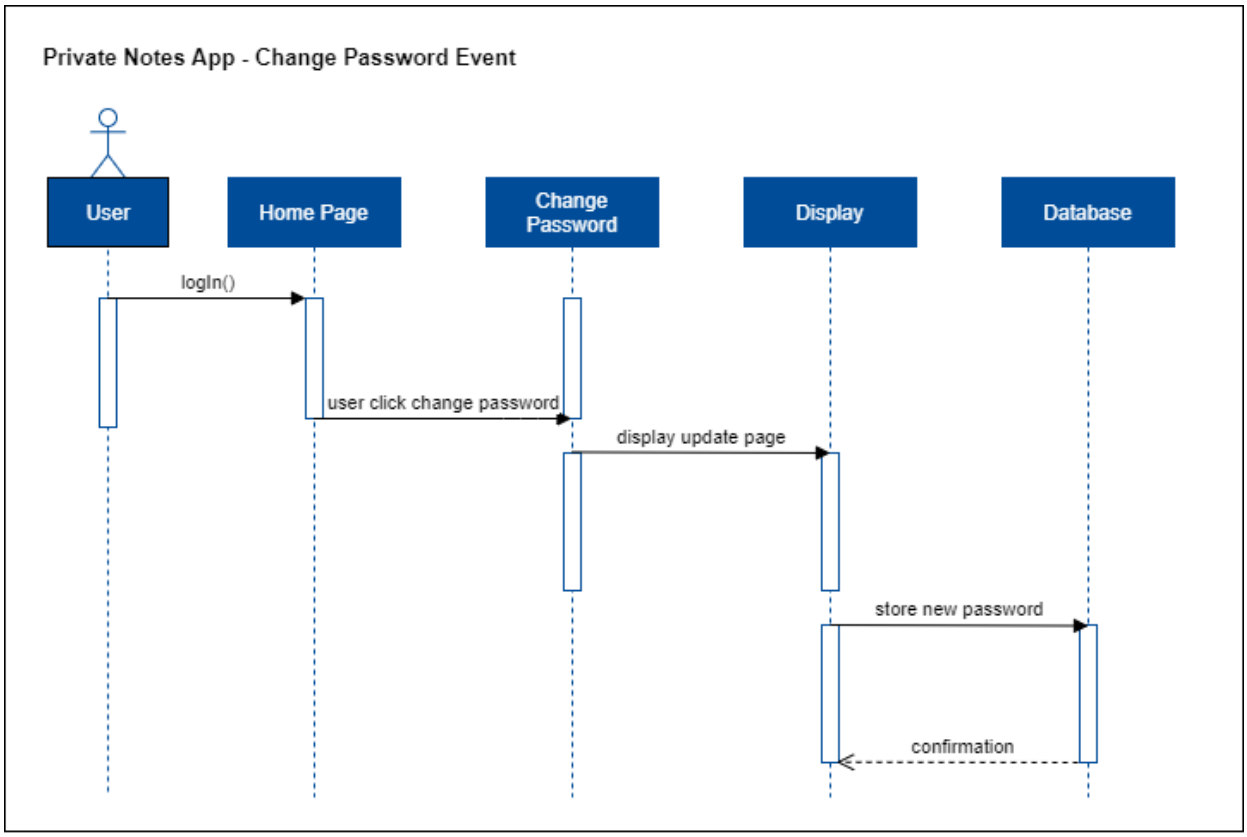
Login Event:

First event of Private Notes web application. User enters credentials into username and password sections and click on 'Login' button. For first time users, system will encrypt the password before storing into database. After getting credentials as input, system applies appropriate hash method to encrypt user password; then, data will pass into server's database for validation. Database returns whether the user has entered valid credentials or not. If credentials are matched, then user will successfully send to home page.



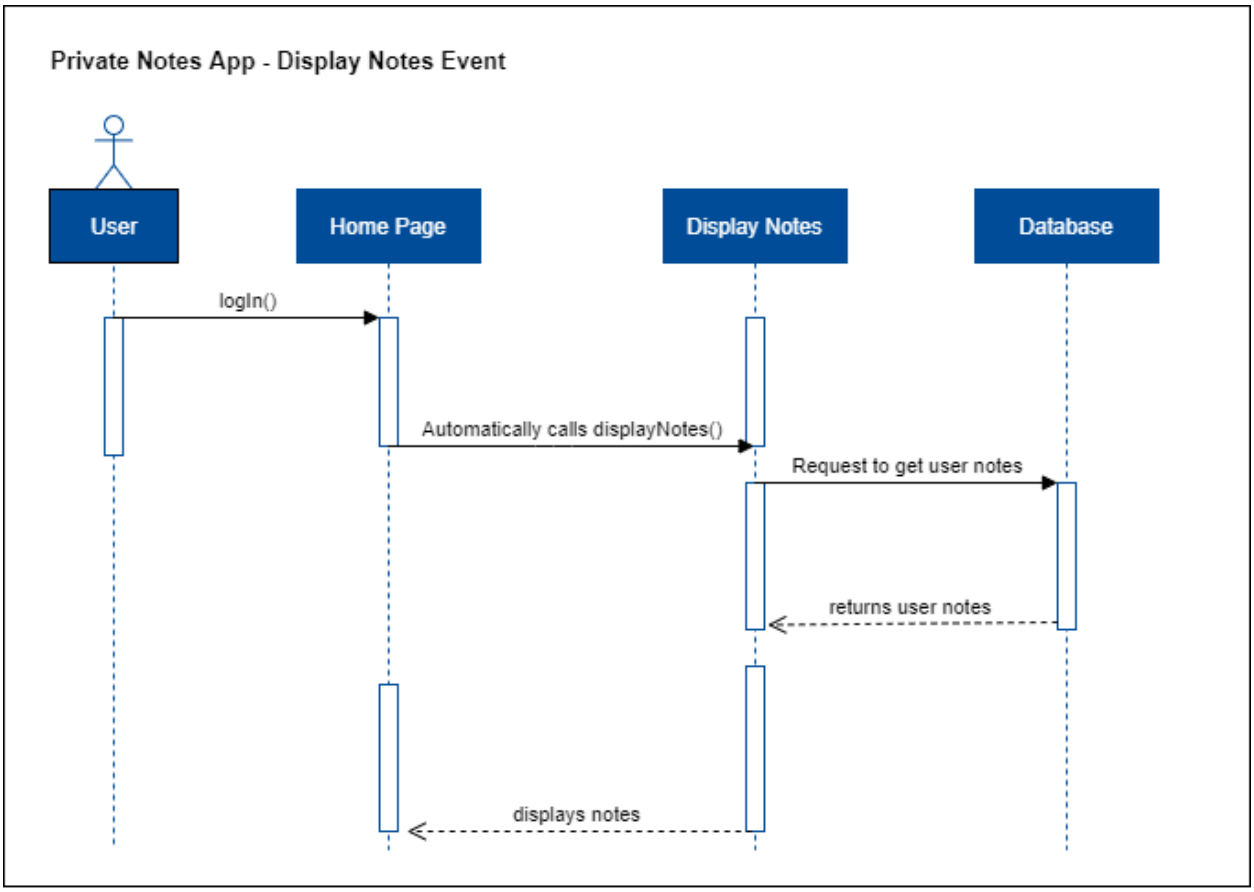
Change Password Event:

After successful login, user have option to update password by clicking on ‘Change Password’ present on the homepage. Then, link embedded in button will send user to another page where they can enter new password information. System will pass new credentials into server’s database to store changes on user profile.



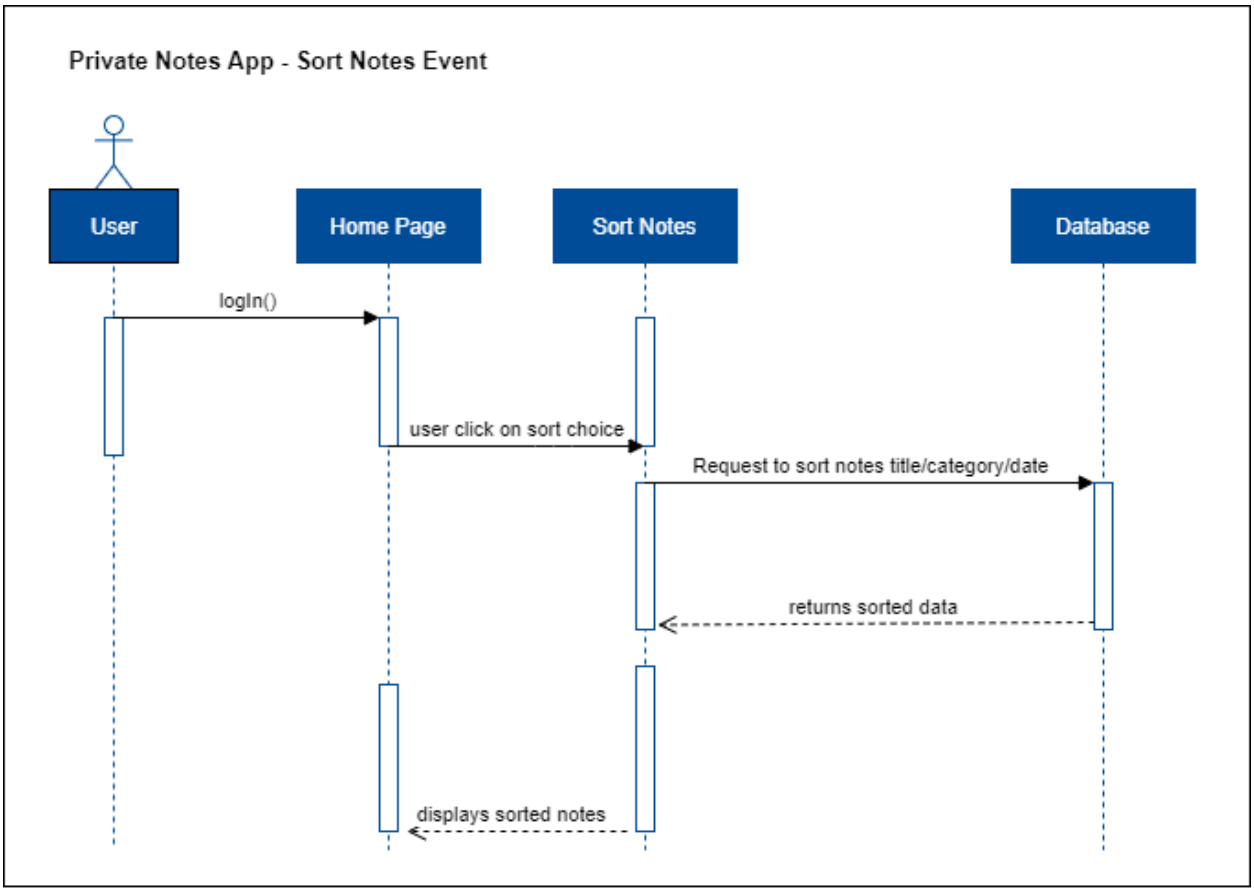
Display Note:

After successful login, user will lead to home page where most functionality of system is available. Home Page automatically calls display Notes() without having need to user request and get created notes from database to display in homepage. Database is queried for a list of already create notes. This functionality also allows user to edit/delete selected note; selection will be done by clicking on desired note.



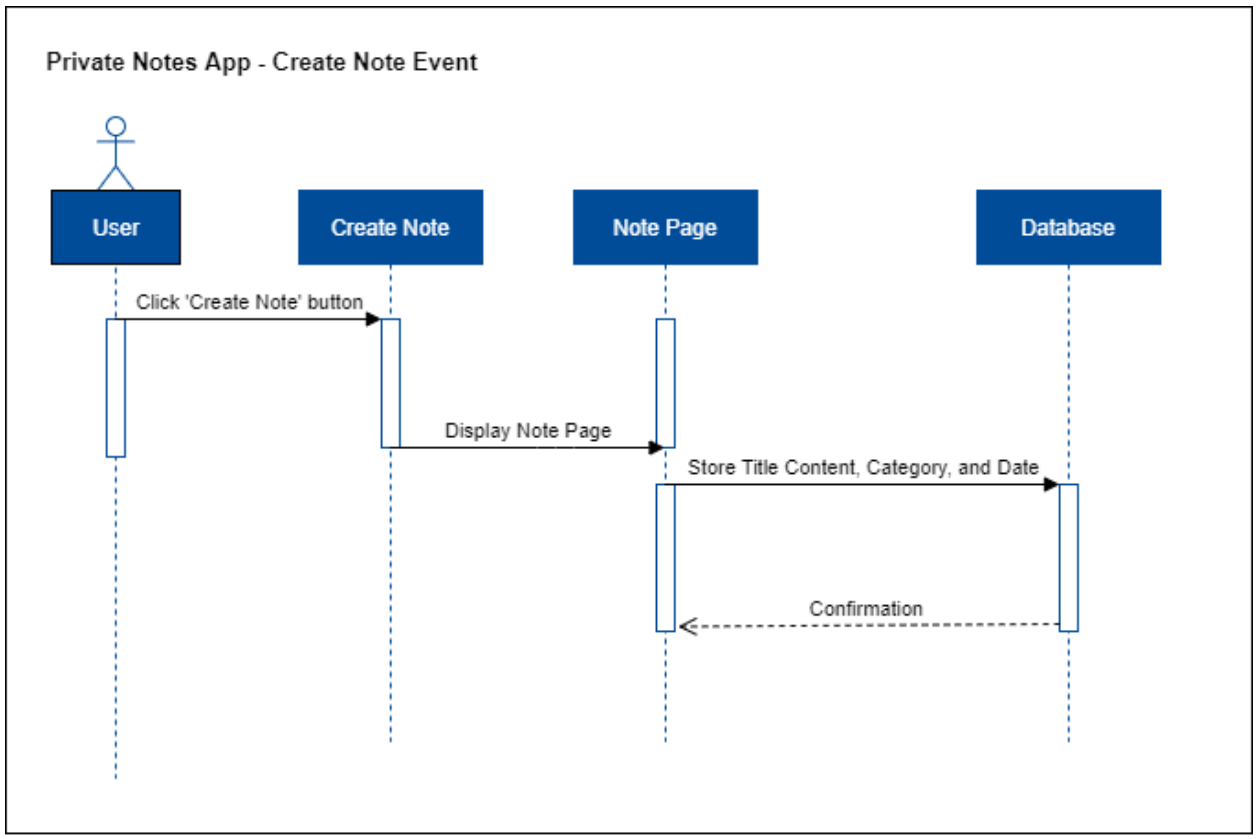
Sort Note:

After successful login, user will lead to home page where most functionality of system is available. Home Page automatically calls display Notes() without having need to user request and get created notes from database to display in homepage. Database is queried for a list of already create notes. This functionality also allows user to edit/delete selected note; selection will be done by clicking on desired note.



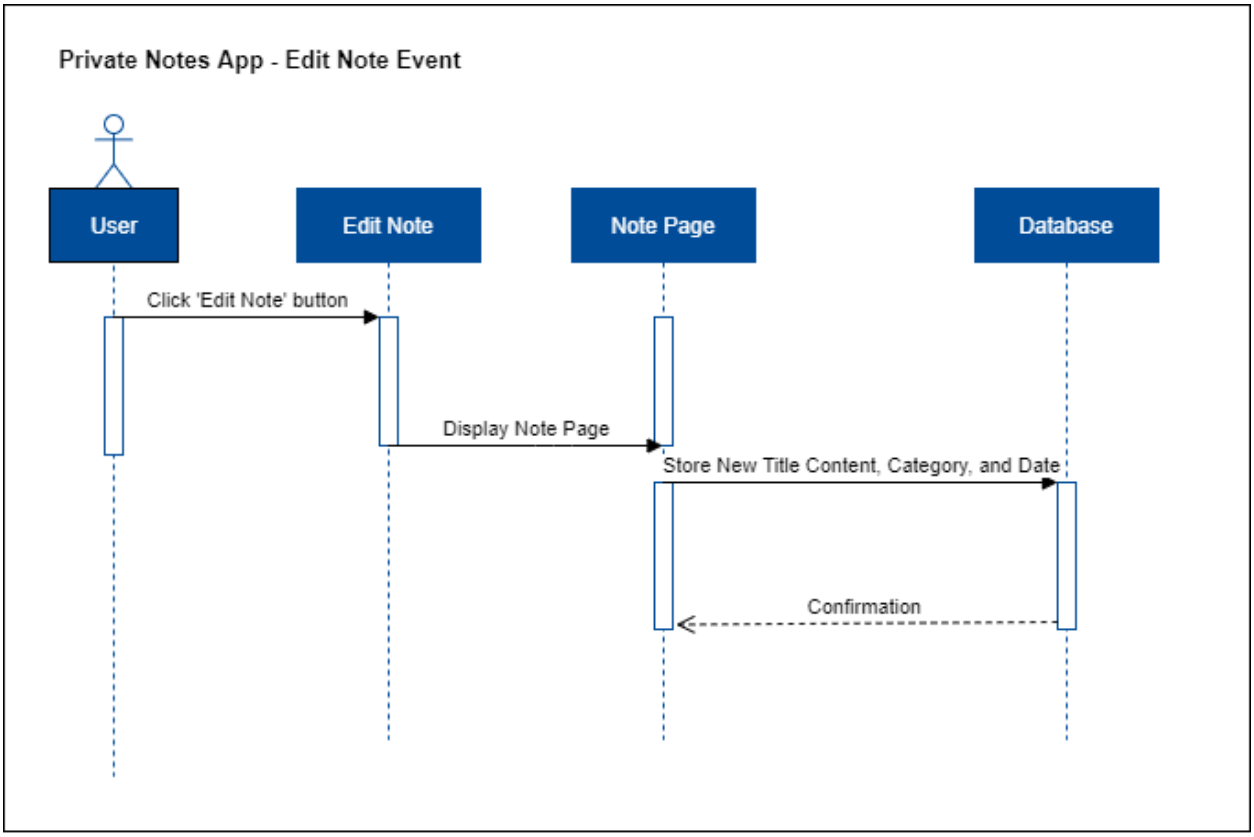
Create Note Event:

Home page provides most functionality of system and user can request to create new note by clicking on ‘Create Note’ button and link embedded in button will send user to another page where user is able to create new note. After entering inputs, user must click on save note button, so system can interact with database. System will pass new note’s data (title, content, category, and data) into server’s database to store new note under user profile.



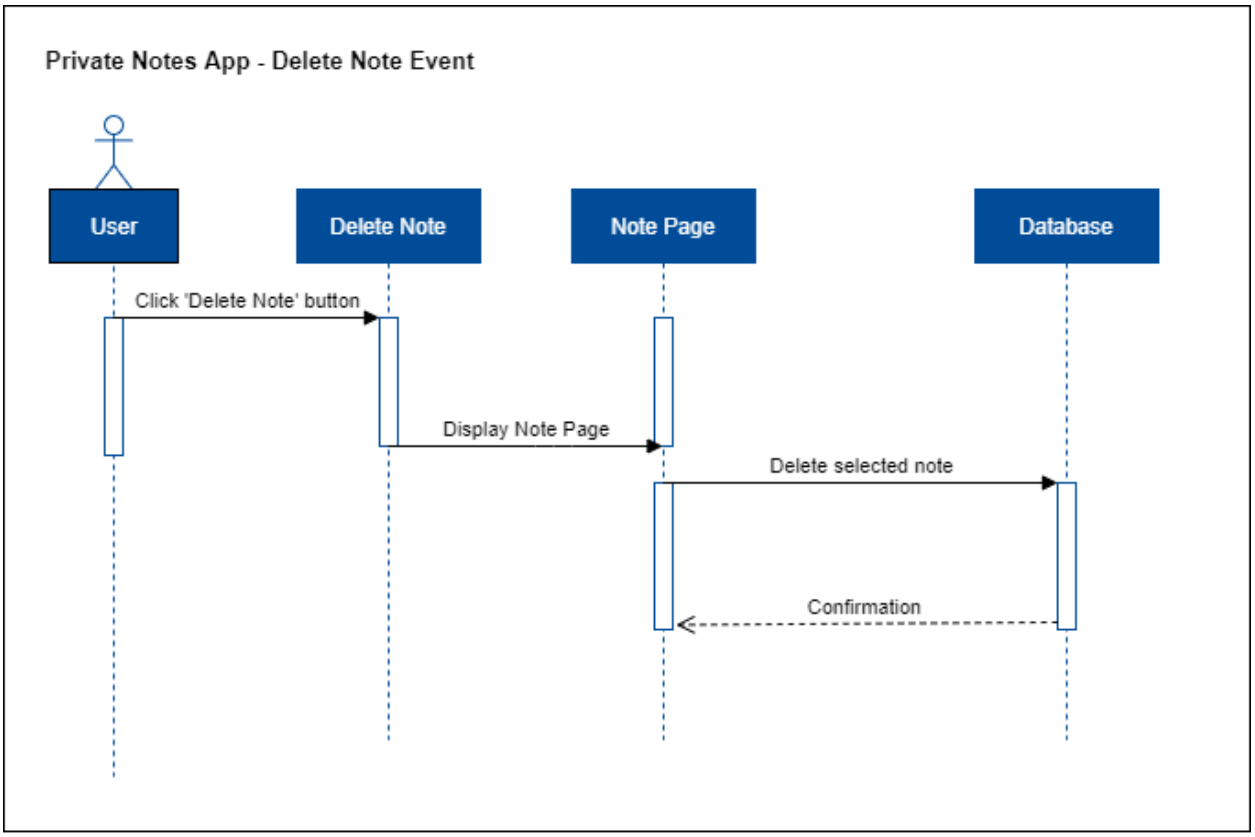
Edit Note:

Home page provides most functionality of system and user can request to edit existed notes by clicking on desired note, which displayed in home page. Link embedded in note will send user to another page where user is able to edit selected note. In new page, user is able to see all variables like title, content, etc. which depend on instance of note object. After getting new inputs, system will pass note's data (title, content, category, and data) into server's database to store updates under user profile.



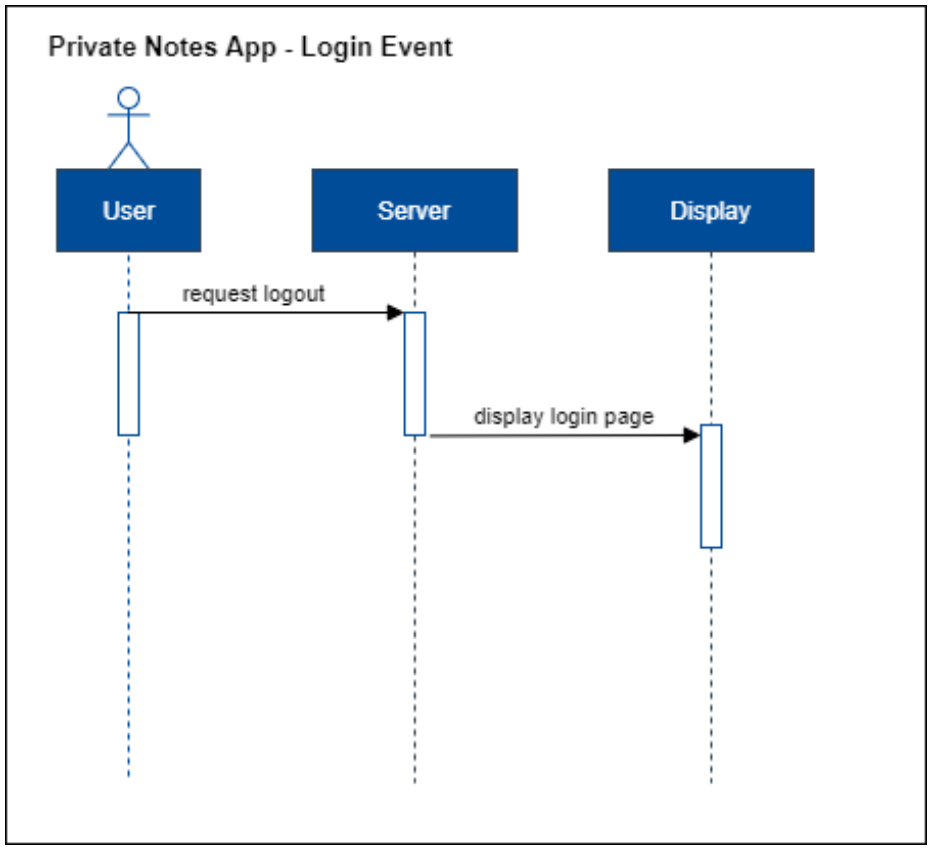
Delete Note:

Home page provides most functionality of system and user can request to delete existed notes by clicking on desired note, which displayed in home page. Link embedded in note will send user to another page where user is able to delete selected note. In new page, user simply click on ‘Delete Note’ to perform action. After getting request from user, system will update database.



Logout Event:

User click on 'Log Out' button which present in home page. System will process user request and user will be returned to log in page.



Trace of Requirements to Design

Req ID	Requirement Description	Architectural Reference (high level design reference)	Design Reference (class diagram reference)
1	**Functional Requirements**		
1.1	System shall be web application	Client (Web Browser) Server(Web Server)	All components from class diagram
1.2	System shall accept credentials from user and have capability of processing the authentication process and validity checks of inputs	Login/Create Account Validate Credentials Database	Log in User Class
1.3	System shall handle all possible errors caused by user	Login/Create Account Create/Edit Note	Log in Note Class (Create/Edit Note)
1.4	System shall handle changing password action and authentication process	Change Password Database	User Class Change Password

1.5	System shall allow user to make changes on inputs for notes, title, and category.	Edit/Delete Note Database	Note Class (Edit/Delete Note)
1.6	System shall allow user to sort notes based on its title, category, and date.	Sort Notes Database	Home Page Sort Notes
1.7	System shall respond to users' requests	Web Server	All components from class diagram
2	**Interface Requirements**		
2.1	System shall accept data items	Login/Create Account Change Password Create/Edit Note Database	User Class (Login/Change Password) Note Class (Create/Edit Note)
2.2	System shall different data types for different inputs	Login/Create Account Change Password Create/Edit Note Database	User Class (Login/Change Password) Note Class (Create/Edit Note)
2.3	System shall output error message in case of authentication issue or saving notes without title	Login/Create Account Create/Edit Note	Login Note Class (Create/Edit Note)
2.4	System shall encrypt user password with appropriate hashing methods	Encrypt User Credentials	User Class Change Password
2.5	System shall receive and sent data items based on the user request. System shall display created notes interface every time user logs in to system.	Web Server	Home Page Display Notes
2.6	System shall handle authentication process	Login/Create Account Validate Credentials Database	User Class Login/Create Account
3	**Physical Environment Requirements**		
3.1	System requires an active internet connection to operate	Web Server	All components from class diagram
3.2	System shall operate in any location where user can access to internet connection	Web Server	All components from class diagram
3.2	System shall run on equipment on web browser with an internet connection	Web Bowser Web Server	All components from class diagram
4	**User and Human Factors Requirements**		
4.3	System shall detect and prevent misuse with authentication.	Login/Create Account Validate Credentials Database	Login
5	**Documentation Requirements**		
6	**Data Requirements**		
6.1	System shall store username/password to manage login actions	Login/Create Account Encrypt User Credentials Database	User Class Login
6.2	System shall perform hash function calculation to encrypt user password to ensure security	Encrypt User Credentials	User Class Change Password

6.3	System shall store each information related with specific note	Database	Note Class (Create/Edit Note)
6.4	System shall accept user input on search/organize options. System shall display notes sequentially	Sort Note Display Notes	Home Page Sort Notes Display Notes
7	**Resource Requirements**		
7.3	System shall utilize from determined software tools such as React.js, JavaScript programming language, or MySQL to create the product System shall utilize system design and system schedule/plan System shall work in any web browser with reliable internet connection	All components from high level design	All components from class diagram
8	**Security Requirements**		
8.1	System must handle encryption operations for user password and handle authentication System must isolate one user's date from others System must control one's access to other systems and information	Login/Create Account Encrypt User Credentials Validate Credentials Database	User Class Login Change Password
8.2	System shall backup every time user add/delete/edit information and store backup copies in different location for recovery	Database	Change Password Note Class (Create/Edit/Delete Note)
9	**Quality Assurance Requirements**		
9.1	System must detect mismatch password and output error message; System must also detect notes without title and output error message. In both cases, system will restrict user to complete action.	Login/Create Account Validate Credentials Database Create/Edit Note	User Class (Login/Change Password) Note Class (Create/Edit Note)
9.3	Any user should be able to complete tasks in less than 10 minutes	Web Server Web Browser Database	All components from class diagram

Requirement ID

1. Functional Requirements
   1.1 Web application
   1.2 Authentication Process / Validity Checks of Inputs
   1.3 Responses to abnormal situations / Generate Error Message
   1.4 Editing Credentials
   1.5 Editing Notes and Related Information
   1.6 Sorting Notes
   1.7 Relationship of Outputs to Inputs

2. Interface Requirements
    2.1 What Data Items Are Input
    2.2 What is the Data Type
    2.3 What Data Items Are Output
    2.4 Data Encryption
    2.5 How Often Will Each Data Item Be Received or Sent
    2.6 Accuracy of Data
3. Physical Environment Requirements
    3.1 Active Internet Connection
    3.2 Environmental Consideration
    3.3 Types of Equipment
4. User and Human Factors Requirements
    4.1 Types of Users
    4.2 Additional Training and Documentation
    4.3 Misuse of System
5. Documentation Requirements
    5.1 Documentation
6. Data Requirements
    6.1 Input for Credentials
    6.2 Calculation for Password Encryption
    6.3 Inputs for Notes
    6.4 Input for Search/Organize Options
7. Resource Requirements
    7.1 Skilled Personnel to Build, Use, and Maintain System
    7.2 Schedule
    7.3 Hardware/Software/Tools
8. Security Requirements
    8.1 Security of User Account
    8.2 Backup
9. Quality Assurance Requirements
    9.1 Detect and Isolate Faults
    9.2 Availability of System
    9.3 Prescribed Time for User to Adjust Product

Test Plan

Esin Sari

Project 3: Private Notes

COP4331, Fall, 2020

Contents of this Document

Overall Objective for Software Test Activity
Description of Test Environment
Overall Stopping Criteria
Description of Individual Test Cases
Appendices

Overall Objective for Software Test Activity

The software test effort will confirm that all functionalities correctly performs and meet with the requirements specified in Software Requirements Specification document. During testing phase, I am planning to test adding/editing/deleting/sorting/displaying note functionalities; the expected result is system should be able to add/edit/delete/sort/display notes and interact with database. Moreover, I am planning to test login and authentication process and system should encrypt password, validate credentials, and complete login action correctly. In addition to these, I am planning to find and fix all possible errors and display appropriate error message. At the end, I should have a working web application. (All previous documents will be reference in testing phase.)

Description of Test Environment

The application will be tested on computer which has access to internet connection. When testing the application, used environment will be the same environment in which the software will operate.

Web Application

- Browser Environment (Google Chrome, Firefox, Safari, Internet Explorer)

The testers for this project will be the developer (Esin Sari).

Stopping Criteria

Decisions when to stop testing the software and either deliver it or send it "back" to development is described below.

- If any errors are found during testing: I will stop testing each time I find a problem and immediately fix that problem. I will also record all errors and group them based on their features in document to increase reusability.
- If no errors are found during testing: At that moment, software will be good enough to deliver. All defined (reasonably possible) test cases should be run before declaring the software to be "good enough to deliver".
- The system will be defined as "good enough to deliver", if it passes all defined test cases successfully. It requires that there be no known errors other than cosmetic errors and errors for which there is a well-defined workaround. Cosmetic errors should not affect processing of the application and not cause any display issues.

Description of Individual Test Cases

Test Case ID: 1
Test Objective: User logs in to the system successfully with valid username and password.
Test Description: The developer runs the application and in the first screen (login screen) clicks on username field and types username, clicks on the password field and types password. Then the user clicks on “log in” button.
Test Conditions: This test should be performed in the login/create account module. For more information, see Test Environment.
Expected Results: The home page opens, displays notes, sorting functionality, and logout button.

Test Case ID: 2
Test Objective: User tries to log in to the system with invalid username or password.
Test Description: The developer runs the application and in the first screen (login screen) clicks on username field and types username, clicks on the password field and types invalid password. Then the user clicks on “log in” button.
Test Conditions: This test should be performed in the login/create account module. For more information, see Test Environment.

Expected Results: System displays error message

Test Case ID: 3
Test Objective: User successfully adds note.
Test Description: The developer successfully logs in and in the home screen, clicks on “create note button” and application opens note screen. Developer clicks on the title field and types title; clicks on the content field and types content; clicks on the category field and types category. Then the user clicks on “save note” button. Date should also be saved in database.
Test Conditions: This test should be performed in the homepage and note module. For more information, see Test Environment.
Expected Results: Created note should be visible in home page.

Test Case ID: 4
Test Objective: User tries to save note without giving title.
Test Description: The developer successfully logs in and in the home screen, clicks on “create note button” and application opens note screen. Developer clicks on the content field and types content; then, user clicks on “save note” button. Date should also be saved in database.
Test Conditions: This test should be performed in the homepage and note account module. For more information, see Test Environment.
Expected Results: System displays error message and does not save note.

Test Case ID: 5
Test Objective: User edits note.
Test Description: The developer successfully logs in and in the home screen, clicks on (already created) desired note and application opens note screen. Developer clicks on the title field and types new title; clicks on the content field and types new content; clicks on the category field and types new category. Then the user clicks on “save note” button. Date should also be updated database.

Test Conditions: This test should be performed in the homepage/note module. For more information, see Test Environment.
Expected Results: Edited note should be visible in home page and its detail should be visible in note page.

Test Case ID: 6
Test Objective: User delete notes.
Test Description: The developer successfully logs in and in the home screen, clicks on (already created) desired note and application opens note screen. Developer clicks on “delete note” button.
Test Conditions: This test should be performed in the homepage/note module. For more information, see Test Environment.
Expected Results: Deleted note is no longer visible in home page and not stored in database.

Test Case ID: 7
Test Objective: User sorts notes.
Test Description: The developer successfully logs in and in the home screen, clicks on desired sorting option note (by title/category/date).
Test Conditions: This test should be performed in the homepage module. For more information, see Test Environment.
Expected Results: Notes are displayed in sorted way (based on the selected option) in home page.

Test Case ID: 8
Test Objective: User logs out.
Test Description: The developer clicks on “log out” button in the home screen.
Test Conditions: This test should be performed in the homepage module. For more information, see Test Environment.

Expected Results: Developer returns to login page

Trace of Individual Test Cases to Requirements

Req ID	Requirement Description	Test Case Reference	Status
1	**Functional Requirements**		
1.1	System shall be web application	All test cases	Progress
1.2	System shall accept credentials from user and have capability of processing the authentication process and validity checks of inputs	1, 2	Progress
1.3	System shall handle all possible errors caused by user	2, 4	Progress
1.4	System shall handle changing password action and authentication process	1, 2	Progress
1.5	System shall allow user to make changes on inputs for notes, title, and category.	5	Progress
1.6	System shall allow user to sort notes based on its title, category, and date.	7	Progress
1.7	System shall respond to users' requests	All test cases	Progress
2	**Interface Requirements**		
2.1	System shall accept data items	1,2,3,4,5	Progress
2.2	System shall different data types for different inputs	1,2,3,4,5	Progress
2.3	System shall output error message in case of authentication issue or saving notes without title	2	Progress
2.4	System shall encrypt user password with appropriate hashing methods	1, 2	Progress
2.5	System shall receive and sent data items based on the user request. System shall display created notes interface every time user logs in to system.	3, 4, 5, 7	Progress
2.6	System shall handle authentication process	1, 2	Progress
3	**Physical Environment Requirements**		
3.1	System requires an active internet connection to operate	All test cases	Progress
3.2	System shall operate in any location where user can access to internet connection	All test cases	Progress
3.2	System shall run on equipment on web browser with an internet connection	All test cases	Progress

4	**User and Human Factors Requirements**		
4.3	System shall detect and prevent misuse with authentication.	2	Progress
5	**Documentation Requirements**		
6	**Data Requirements**		
6.1	System shall store username/password to manage login actions	1	Progress
6.2	System shall perform hash function calculation to encrypt user password to ensure security	1, 2	Progress
6.3	System shall store each information related with specific note	1, 2, 3, 4, 7	Progress
6.4	System shall accept user input on search/organize options. System shall display notes sequentially	7	Progress
7	**Resource Requirements**		
7.3	System shall utilize from determined software tools such as React.js, JavaScript programming language, or MySQL to create the product System shall utilize system design and system schedule/plan System shall work in any web browser with reliable internet connection	All test cases	Progress
8	**Security Requirements**		
8.1	System must handle encryption operations for user password and handle authentication System must isolate one user's date from others System must control one's access to other systems and information	1, 2	Progress
8.2	System shall backup every time user add/delete/edit information and store backup copies in different location for recovery	1, 3, 5, 6	Progress
9	**Quality Assurance Requirements**		
9.1	System must detect mismatch password and output error message; System must also detect notes without title and output error message. In both cases, system will restrict user to complete action.	2, 4	Progress
9.3	Any user should be able to complete tasks in less than 10 minutes	All test cases	Progress

Appendices: Test files will not be used in test cases.

Requirement ID

10. Functional Requirements
    - 10.1 Web application
    - 10.2 Authentication Process / Validity Checks of Inputs
    - 10.3 Responses to abnormal situations / Generate Error Message
    - 10.4 Editing Credentials
    - 10.5 Editing Notes and Related Information
    - 10.6 Sorting Notes
    - 10.7 Relationship of Outputs to Inputs
11. Interface Requirements
    - 11.1 What Data Items Are Input
    - 11.2 What is the Data Type
    - 11.3 What Data Items Are Output
    - 11.4 Data Encryption
    - 11.5 How Often Will Each Data Item Be Received or Sent
    - 11.6 Accuracy of Data
12. Physical Environment Requirements
    - 12.1 Active Internet Connection
    - 12.2 Environmental Consideration
    - 12.3 Types of Equipment
13. User and Human Factors Requirements
    - 13.1 Types of Users
    - 13.2 Additional Training and Documentation
    - 13.3 Misuse of System
14. Documentation Requirements
    - 14.1 Documentation
15. Data Requirements
    - 15.1 Input for Credentials
    - 15.2 Calculation for Password Encryption
    - 15.3 Inputs for Notes
    - 15.4 Input for Search/Organize Options
16. Resource Requirements
    - 16.1 Skilled Personnel to Build, Use, and Maintain System
    - 16.2 Schedule
    - 16.3 Hardware/Software/Tools
17. Security Requirements
    - 17.1 Security of User Account
    - 17.2 Backup
18. Quality Assurance Requirements
    - 18.1 Detect and Isolate Faults
    - 18.2 Availability of System
    - 18.3 Prescribed Time for User to Adjust Product