|  |  |
| --- | --- |
| Date: |  |
| Experiment No. & Title: | 01 & Problem Statements |
| Submitted by: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

PROBLEM STATEMENTS:

Introduction:

In today's digital age, people have to manage a large number of passwords and notes for various online accounts, personal information, and important data. With an increasing number of online services and applications, it becomes difficult to keep track of all the passwords and notes. Moreover, people tend to use weak passwords or reuse the same password for multiple accounts, which puts their personal and sensitive data at risk of being compromised.

To address this issue, many password and note management tools have been developed that offer features like password generation, password storage, secure notes, and encryption. However, choosing the right password and note management tool can be overwhelming as there are numerous options available in the market, each with its unique features, advantages, and drawbacks.

This problem statement aims to explore the challenges faced by users in managing their passwords and notes and to identify the key features that a password and note management tool should offer to ensure secure and efficient management of personal data.

Problem:

In today's world, where digital technologies play a significant role in our daily lives, the need to manage multiple passwords and notes has become a daunting task. With the growing number of online services, it becomes difficult to keep track of all the passwords and notes associated with them. Moreover, weak passwords or using the same password for multiple accounts can expose personal data to security threats.

To address this issue, password and note management tools have been developed to help users generate, store, and manage passwords and notes securely. However, choosing the right tool that suits individual needs can be a challenge, as different tools offer varying features and levels of security.

This problem statement aims to identify the challenges faced by users in managing their passwords and notes and to explore the features that can enhance the security and usability of password and note management tools. The study will focus on identifying the key factors that influence users' choices of password and note management tools and develop recommendations to guide users in selecting the most appropriate tool for their needs.

ONE PAGE BUSINESS CASE TEMPLATE

The Project:

To address these issues, password and note management tools have been developed to provide users with features such as password generation, storage, and encryption. However, choosing the right tool can be overwhelming as there are numerous options available in the market.

The Constraints:

The primary objective of this proposal is to identify the challenges faced by users in managing their passwords and notes and to explore the features that can enhance the security and usability of password and note management tools. The study aims to:

* Identify the factors that influence users' choices of password and note management tools.
* Develop recommendations to guide users in selecting the most appropriate tool for their needs.
* Evaluate the usability and security of different password and note management tools available in the market.
* Identify the gaps in the existing password and note management tools and propose solutions to enhance their usability and security.

Methodology:

The proposed study will use a mixed-methods approach that combines qualitative and quantitative data collection methods. The study will begin with a survey to gather quantitative data on users' password and note management habits, their experiences with existing tools, and the factors that influence their choices of tools. Next, qualitative data will be collected through in-depth interviews with selected users to gain a deeper understanding of their experiences, challenges, and needs in managing their passwords and notes.

Expected outcomes:

The proposed study is expected to contribute to the development of better password and note management tools that are secure, user-friendly, and meet the needs of users. The study findings will provide insights into the challenges faced by users in managing their passwords and notes and the features that can enhance the usability and security of password and note management tools. The study will also contribute to the development of guidelines to help users select the most appropriate password and note management tool for their needs.

Conclusion:

In conclusion, the proposed study will help improve the usability and security of password and note management tools by identifying the challenges faced by users and the features that can enhance the tools' usability and security. The study will also provide guidelines for users to select the most appropriate password and note management tool for their needs.

|  |  |
| --- | --- |
| Date: |  |
| Experiment No. & Title: | 02 & Stakeholders and User Description |
| Submitted By: | Vartika Singh(RA21110030320241)  Akriti Deo (RA2111003020309) |

STAKEHOLDERS AND USER DESCRIPTION:

Stakeholders:

Users: The primary stakeholders in notes and password management are the users who need to manage their passwords and notes securely and efficiently.

Password and note management tool developers: The developers of password and note management tools are also stakeholders as they aim to provide secure and user-friendly tools that meet users' needs.

IT security experts: IT security experts are stakeholders as they provide insights into the security requirements of password and note management tools.

Regulatory authorities: Regulatory authorities may be stakeholders as they may have specific requirements regarding the security and privacy of personal data.

User Description:

Casual users: These are users who use a limited number of online services and do not have complex password and note management requirements. They may need a simple and user-friendly password and note management tool that is easy to use and understand.

Power users: These are users who have complex password and note management requirements and need a tool that can handle multiple accounts, generate secure passwords, and manage notes securely. They may require a tool that offers advanced features and customization options.

Corporate users: These are users who need to manage passwords and notes for their work-related accounts and may require a tool that can be integrated with their organization's IT infrastructure and meet specific security requirements.

Security-conscious users: These are users who prioritize the security and privacy of their personal data and need a tool that offers advanced security features such as two-factor authentication, encryption, and secure sharing options. They may also require a tool that does not store their data on the cloud but on their local device.

|  |  |
| --- | --- |
| Date: |  |
| Experiment No. & Title: | 03 & Identifying the Requirements from the Project Statement |
| Submitted By: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

IDTIFYING THE REQUIREMENTS FROM THE PROJECT STATEMENT:

Based on the project statement for notes and password management, the following requirements can be identified:

1. Identify the challenges faced by users in managing their passwords and notes: The project aims to identify the common challenges faced by users in managing their passwords and notes, such as remembering multiple passwords, creating and storing secure notes, and organizing them efficiently.
2. Explore the features that can enhance the security and usability of password and note management tools: The project aims to explore the features that can improve the usability and security of password and note management tools, such as password generation, encryption, secure sharing, and integration with other applications.
3. Identify the factors that influence users' choices of password and note management tools: The project aims to identify the factors that influence users' choices of password and note management tools, such as price, usability, security, and compatibility with their devices and operating systems.
4. Develop recommendations to guide users in selecting the most appropriate tool for their needs: The project aims to develop guidelines that can help users select the most appropriate password and note management tool for their needs, based on their requirements and preferences.
5. Evaluate the usability and security of different password and note management tools available in the market: The project aims to evaluate the usability and security of different password and note management tools available in the market, to identify their strengths and weaknesses and to recommend improvements.
6. Identify the gaps in the existing password and note management tools and propose solutions to enhance their usability and security: The project aims to identify the gaps in the existing password and note management tools and propose solutions to enhance their usability and security, based on the identified challenges, features, and user requirements.

Overall, the requirements of the project include identifying the challenges and requirements of users, exploring the features of password and note management tools, evaluating the existing tools, and proposing recommendations and solutions to improve their usability and security.

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 04 & Project plan and Project Effort based on Resources |
| Submitted By: | Vartika Singh(RA2111003020241)  Akriti Deo (RA2111003020309) |

PROJECT PLAN AND PROJECT EFFORT BASED ON RESOURCES

Project Name:

Notes and Password Manangement

Project Members:

1. Vartika Singh (RA2111003020241)
2. Akriti Deo (RA2111003020309)

Modules:

LastPass - This is a password manager that allows you to securely store your passwords, credit card information, and other sensitive data. It also has a notes feature that allows you to store important information like account numbers, addresses, and more.

Evernote - This is a note-taking application that allows you to organize and store all your notes in one place. It also has a password-protected section where you can store sensitive information.

OneNote - This is another note-taking application that is part of the Microsoft Office Suite. It has a password-protected section where you can store sensitive information like passwords, credit card information, and other important data.

Google Keep - This is a note-taking application that allows you to quickly capture ideas, to-do lists, and reminders. It also has a feature that allows you to add labels to notes for easy organization.

1Password - This is a password manager that securely stores your passwords, credit card information, and other sensitive data. It also has a notes feature that allows you to store important information like account numbers, addresses, and more.

KeePass - This is a free and open-source password manager that allows you to store your passwords and other sensitive data in an encrypted database. It also has a notes feature that allows you to store important information.

All of these modules and applications can be helpful for notes and password management, but the best one for you will depend on your specific needs and preferences

PROJECT COST:

• Development of the project:

|  |  |
| --- | --- |
| **Resource Requirement** | **COST** |
| Computer with core i7 8th gen processor, at least 8Gb of RAM, running on windows 11 | Rs.50000/- |
| Code | Open source |
| Printing | Rs.500/- |

• Server-End:

|  |  |
| --- | --- |
| **Resource Requirement** | **COST** |
| My SQL | Enterprise Edition Rs.14078/- |
| Azure Web Services | Std. edition Rs.24000/- |
| UPS | Rs.34000/- |
|  |  |

• Other cost:

|  |  |
| --- | --- |
| Employee Salary | ­­­­ |
| Maintenance Cost | Rs.2500/- per month |

IDENTIFYING JOB ROLES AND RESPONSIBILITIES:

|  |  |
| --- | --- |
| **Members** | **Roles and Resposibilities** |
| Vartika Singh (RA2111003020241) | • TEAM LEADER: Responsibility of coordination of the team, checking for errors, updating for the current status of the project, guiding the team.  • DEVELOPER: Responsibility of coding, compiling and debugging of the modules.  • SOFTWARE ARCHITECT: Design of UML Diagrams and other blueprint.  • WEB DEVELOPER: Responsibility of designing the website and interfacing with the server.  • DEVELOPER: Responsibility of coding compiling and debugging of the modules. |
| Akriti Deo  (RA2111003020309) | • MANUAL TESTER: Responsibility of testing the project at a smaller level and reporting the errors.  • WEB DEVELOPER: Responsibility of designing the website and interfacing with the website.  • TESTER: Responsibility of testing the project at a vast level and variety of ways.  • DESIGNER: Identifying areas for modification in existing programs and developing the modification |

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 05 & Project Effort Based on Resource |
| Submitted By: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

PROJECT EFFORT BASED ON RESOURCE:

Work Breakdown for Project:

Possible work breakdown for creating a notes and password management module:

Requirements gathering: Define the specific requirements for the notes and password management module, including features, user interface, security, and integration with other systems.

Design and architecture: Develop the design and architecture of the module, including data models, user flows, and UI wireframes.

Implementation: Develop the software using an appropriate programming language and framework. This will include coding the functionality for storing and retrieving notes and passwords, creating a user interface for managing the data, and implementing security features such as encryption.

Testing: Conduct thorough testing of the module to ensure that it meets the requirements and functions as expected. This will include unit testing, integration testing, and system testing.

User acceptance testing: Obtain feedback from users to ensure that the module meets their needs and is easy to use.

Documentation: Create user guides, installation instructions, and other documentation to support the module's deployment and use.

Deployment: Deploy the module to production or the appropriate environment.

Maintenance and support: Provide ongoing maintenance and support for the module, including bug fixes, updates, and enhancements.

It's important to note that the actual work breakdown may vary depending on the specific requirements and resources available for the project.

Risk Analysis:

Risk analysis for notes and password management

When it comes to notes and password management, there are a few key risks to consider. Here are some potential risks and ways to mitigate them:

Unauthorized access:

One of the biggest risks is that someone unauthorized may gain access to your notes or passwords. This could happen if your device is lost or stolen, or if someone hacks into your account.

To mitigate this risk, you should use strong, unique passwords for all of your accounts, and enable two-factor authentication whenever possible. You should also keep your device locked with a password or fingerprint, and use encryption tools to protect your data.

Data loss:

Another risk is that you may lose your notes or passwords due to a hardware or software failure, or accidentally delete them.

To mitigate this risk, you should regularly back up your data to a secure location, such as the cloud or an external hard drive. You should also consider using a password manager that automatically syncs your passwords across devices and backs them up to the cloud.

Insider threats:

It's also possible that someone with authorized access to your notes or passwords could misuse them, such as a co-worker or family member who has access to your device.

To mitigate this risk, you should limit access to your notes and passwords as much as possible, and only share them with people you trust. You should also monitor your accounts regularly for any suspicious activity.

Social engineering attacks:

Another risk is that someone could trick you into giving them your passwords or personal information through a social engineering attack, such as a phishing email or phone call.

To mitigate this risk, you should be cautious of unsolicited messages or requests for personal information, and always verify the identity of the person or organization before providing any sensitive information. You should also educate yourself on common social engineering tactics and stay up-to-date on the latest security threats.

Overall, effective risk management for notes and password management involves a combination of strong passwords, encryption, backups, limited access, monitoring, and education. By implementing these strategies, you can reduce the likelihood of a security breach and protect your sensitive information

|  |  |
| --- | --- |
| Date: |  |
| Experiment No. & Title: | 06 & Estimation Of Project Metrices |
| Submitted By: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

ESTIMATION OF PROJECT METRICES:

Some possible metrics that could be used to estimate the success and effectiveness of a notes and password management system:

User satisfaction:

This metric measures the satisfaction level of users with the system. User satisfaction can be measured through surveys, user feedback, and other qualitative methods.

Number of users:

This metric measures the number of users who are actively using the system. This metric is an indicator of the system's popularity and adoption rate.

Number of notes and passwords managed:

This metric measures the total number of notes and passwords stored in the system. This metric can be used to track the growth of the system over time and to identify any capacity or scalability issues.

System uptime:

This metric measures the amount of time the system is available for users. A high uptime percentage indicates that the system is reliable and available when users need it.

Security incidents:

This metric measures the number and severity of security incidents that occur within the system. This metric is an indicator of the system's security posture and can be used to identify areas for improvement.

Average response time:

This metric measures the average amount of time it takes for the system to respond to user requests. A low response time indicates that the system is fast and responsive.

Number of support requests:

This metric measures the number of support requests received from users. This metric can be used to identify common issues and areas where users need additional support or training.

These metrics can help to assess the effectiveness of a notes and password management system and identify areas for improvement. However, it's important to note that the specific metrics used may vary depending on the goals and objectives of the system and the needs of the organization.

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 07 & Design |
| Submitted by: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

DESIGN:  
 System Architecture:

system architecture for a notes and password management system:

Presentation layer: The presentation layer is the user interface through which users interact with the system. This layer includes the graphical user interface (GUI) that users use to create, edit, and view notes and passwords. The presentation layer can be implemented using web technologies such as HTML, CSS, and JavaScript or using native applications for desktop and mobile platforms.

Application layer: The application layer contains the business logic and processing logic of the system. This layer is responsible for receiving user requests from the presentation layer, processing them, and generating appropriate responses. The application layer can be implemented using a server-side programming language such as Python, Java, or Node.js.

Database layer: The database layer is responsible for storing and retrieving data from the system's database. This layer can use a relational database management system (RDBMS) such as MySQL or PostgreSQL, or a NoSQL database such as MongoDB or Couchbase.

Security layer: The security layer is responsible for ensuring the security and privacy of the system's data. This layer can include authentication and authorization mechanisms to control access to the system's data, encryption of sensitive data, and other security measures to protect against attacks such as SQL injection and cross-site scripting (XSS).

Integration layer: The integration layer is responsible for integrating the system with other systems and services. This layer can include APIs and webhooks that allow other systems to access the system's data and functionality.

Infrastructure layer: The infrastructure layer includes the physical and virtual infrastructure that supports the system. This layer can include servers, networking equipment, storage systems, and other infrastructure components that support the system's operation.

The specific implementation of the system architecture will depend on the requirements and resources of the project.

Diagrametic Layout:

Diagram that illustrates a possible system architecture for a notes and password management system:

+---------------------+

| |

| Presentation Layer |

| |

+----------+----------+

|

|

| User requests

|

v

+----------+----------+

| |

| Application Layer |

| |

+----------+----------+

|

|

| Data requests

|

v

+----------+----------+

| |

| Database Layer |

| |

+----------+----------+

|

|

| Data storage/retrieval

|

v

+----------+----------+

| |

| Security Layer |

| |

+----------+----------+

|

|

| Security measures

|

v

+----------+----------+

| |

| Integration Layer |

| |

+----------+----------+

|

|

| Integration with other systems

|

v

+----------+----------+

| |

| Infrastructure |

| |

+-----------------------+

The presentation layer is the user interface through which users interact with the system. The application layer receives user requests from the presentation layer, processes them, and generates appropriate responses. The database layer is responsible for storing and retrieving data from the system's database. The security layer ensures the security and privacy of the system's data. The integration layer is responsible for integrating the system with other systems and services. The infrastructure layer includes the physical and virtual infrastructure that supports the system.

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 08 & MODELLING UML USE CASE DIAGRAM & CAPTURING USE CASE SCENARIOS |
| Submitted By: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

MODELLING UML USE CASE DIAGRAM & CAPTURING USE CASE SCENARIOS:

use case descriptions for a notes and password management system:

Creating a note:

A user wants to create a new note. They open the notes application, click on the "Create New Note" button, and enter the note's title and content. They then save the note, which is stored securely in the database.

Editing a note:

A user wants to edit an existing note. They open the notes application, select the note they want to edit, and make the desired changes to the note's content. They then save the changes, which are updated in the database.

Deleting a note:

A user wants to delete a note they no longer need. They open the notes application, select the note they want to delete, and click on the "Delete" button. The note is removed from the database.

Searching for a note:

A user wants to find a specific note. They open the notes application, enter the search term in the search box, and the system returns a list of notes that match the search criteria.

Generating a password:

A user wants to generate a strong, random password. They open the password management application, click on the "Generate Password" button, and specify the desired password length and complexity. The system generates a secure password and stores it in the database.

Saving a password:

A user wants to save a password for a website or application. They open the password management application, enter the website or application name, the username, and the password. The system securely stores the password in the database.

Retrieving a password:

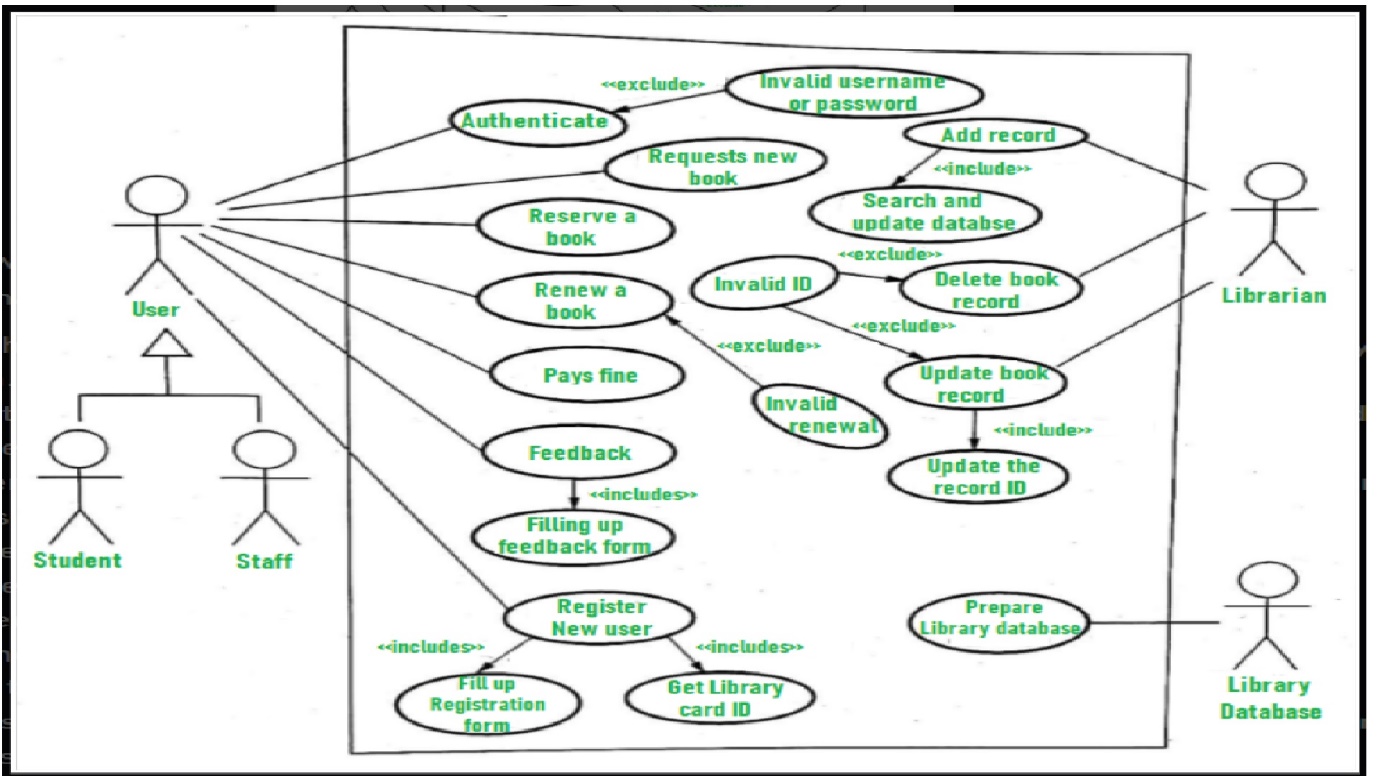
A user wants to retrieve a saved password. They open the password management application, enter the website or application name, and the system retrieves the saved password from the database.

Changing a password:

A user wants to change a saved password. They open the password management application, select the website or application, and click on the "Change Password" button. They enter the new password, and the system securely updates the password in the database.

These use cases illustrate some of the basic functionality that a notes and password management system might provide. Additional use cases could include sharing notes or passwords with other users, syncing data across devices, and generating reports on system usage and activity.

USE CASE DIAGRAM :



The use case diagram depicts three primary use cases: Notes, Passwords, and Notes and Passwords Management. The Notes use case includes the Create Note, Edit Note, Delete Note, and Search Note use cases. The Passwords use case includes the Generate Password, Save Password, Retrieve Password, and Change Password use cases. The Notes and Passwords Management use case includes additional functionality, such as user authentication and authorization, backup and recovery, and system administration. The use case diagram provides a high-level overview of the system's functionality and the interactions between the various use cases.

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 09 & ER MODELLING FROM THE PROBLEM STATEMENT |
| Submitted by: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

ER MODELLING FROM THE PROBLEM STATEMENT:

Er Modelling Description :

ER (Entity-Relationship) model description for a notes and password management system:

The ER model includes the following entities:

User: represents the system users and stores their login credentials and preferences.

Note: represents a note created by a user and stores the note's content, creation and modification timestamps, and the user who created it.

Password: represents a password saved by a user for a website or application and stores the website or application name, the username, and the encrypted password, along with the user who saved it.

The relationships between these entities are as follows:

A User can create multiple Notes and Passwords.

A Note belongs to one User.

A Password belongs to one User.

The ER model can be represented using the following diagram:

+------------+ +------------+

| User | | Note |

+------------+ +------------+

| id |<--+---| user\_id |

| username | | | title |

| password | | | content |

| email | | | created\_at |

| created\_at | | | updated\_at |

+------------+ | +------------+

|

|

|

|

|

|

|

+------------+ | +------------+

| Password | | | User |

+------------+ | +------------+

| id |<---+ | id |

| user\_id | | username |

| website | | password |

| username | | email |

| password | | created\_at |

|  |  |  |
| --- | --- | --- |
|  |  |  |

This ER model allows for the storage and retrieval of notes and passwords for each user, and allows for easy querying and management of the data.

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 10 & IDENTIFYING DOMAIN CLASSES FROM THE PROBLEM STATEMENTS |
| Submitted By: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

IDENTIFYING DOMAIN CLASSES FROM THE PROBLEM STATEMENTS:

DOMAIN CLASS DIAGRAM DESCRIPTION:

domain class diagram description for a notes and password management system:

The domain class diagram includes the following classes:

User: represents the system users and stores their login credentials and preferences.

Note: represents a note created by a user and stores the note's content, creation and modification timestamps, and the user who created it.

Password: represents a password saved by a user for a website or application and stores the website or application name, the username, and the encrypted password, along with the user who saved it.

The relationships between these classes are as follows:

A User can create multiple Notes and Passwords.

A Note belongs to one User.

A Password belongs to one User.

The domain class diagram can be represented using the following diagram:

+------------+ +------------+

| User | | Note |

+------------+ +------------+

| id: int |<--+----| userId: int|

| username: | | | title: |

| string | | | string |

| password: | | | content: |

| string | | | string |

| email: | | | createdAt: |

| string | | | DateTime |

| createdAt: | | | updatedAt:|

| DateTime | | | DateTime |

+------------+ | +------------+

|

|

|

|

|

|

|

+------------+ | +------------+

| Password | | | User |

+------------+ | +------------+

| id: int |<---+ | id: int |

| userId: | | username: |

| int | | string |

| website: | | password: |

| string | | string |

| username: | | email: |

| string | | string |

| password: | | createdAt:|

| string | | DateTime |

+------------+ +------------+

This domain class diagram represents the same entities and relationships as the ER model. It is useful for understanding the system's behavior and the interactions between the classes. The domain class diagram can be used as a blueprint for the actual implementation of the system.

|  |  |
| --- | --- |
| Date : |  |
| Experiment no. & Title: | 11 & STATECHART & COMMUNICATION MODELLING |
| Submitted by: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

STATECHART & COMMUNICATION MODELLING:

STATECHART DIAGRAM DESCRIPTION:

Statechart diagram description for a note and password management system:

The Statechart diagram represents the possible states and transitions for the Note and Password classes.

For the Note class, the possible states are:

Draft: the note has been created but has not been saved.

Saved: the note has been saved but has not been submitted.

Submitted: the note has been submitted and is being processed.

Published: the note has been published and is visible to the user and other users.

The transitions between these states are:

Create: creates a new note in the Draft state.

Save: saves the note in the Saved state.

Submit: submits the note for processing in the Submitted state.

Publish: publishes the note in the Published state.

For the Password class, the possible states are:

Inactive: the password has not been used.

Active: the password is currently being used.

The transitions between these states are:

Use: uses the password and moves it to the Active state.

Deactivate: deactivates the password and moves it to the Inactive state.

The Statechart diagram can be represented using the following diagram:

+-------------------------+ +-------------------------+

| Note | | Password |

+-------------------------+ +-------------------------+

| | | |

| | | |

| v | v

+---------+ +---------+ +----------+ +----------+

| Draft | Create | Saved | | Inactive | Use | Active |

+---------+-------------> | | -> | | ----------> | |

| | | | | | | |

+---------+ +---------+ +----------+ +----------+

| | | |

| Save | Submit | Deactivate |

v v v v

+-----------+ +-----------+ +------------+ +------------+

| Submitted | | Published | | Deleted | | Removed |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |

This Statechart diagram provides an overview of the possible states and transitions for the Note and Password classes. It is useful for understanding the system's behavior and for designing and implementing the necessary logic for each state and transition.

COMMUNICATION DIAGRAM DESCRIPTION:

A communication diagram, also known as a collaboration diagram, is used to show the interactions between objects or classes in a system. Here's a possible description of a communication diagram for a notes and password management system:

The communication diagram shows the interactions between three main classes: User, Note, and Password. The User class interacts with the Note and Password classes to manage their notes and passwords.

Here's a description of the interactions in the communication diagram:

The User creates a new note by sending a createNote() message to the Note class.

The Note class creates a new note object and returns a reference to the User.

The User edits the note by sending an editNote() message to the Note object.

The Note object updates the note content and returns a confirmation message to the User.

The User saves the note by sending a saveNote() message to the Note object.

The Note object saves the note and returns a confirmation message to the User.

1. The User submits the note for processing by sending a submitNote() message to the Note object.
2. The Note object processes the note and updates its status to "Submitted".
3. The User retrieves their list of submitted notes by sending a getSubmittedNotes() message to the Note class.
4. The Note class returns a list of submitted notes to the User.
5. The User creates a new password by sending a createPassword() message to the Password class.
6. The Password class creates a new password object and returns a reference to the User.
7. The User uses the password by sending a usePassword() message to the Password object.
8. The Password object updates its status to "Active" and returns a confirmation message to the User.
9. The User deactivates the password by sending a deactivatePassword() message to the Password object.
10. The Password object updates its status to "Inactive" and returns a confirmation message to the User.

This communication diagram can be represented using the following diagram:

+--------+

| User |

+--------+

|

+---------------+---------------+

| |

+---------+ +----------+

| Note | | Password |

+---------+ +----------+

| |

| createNote() | createPassword()

|------------------------------>|

| |------------------------------>|

| | |

|<------------------------------|<------------------------------|

| editNote() | usePassword()

|------------------------------>|------------------------------>|

| | |

|<------------------------------|<------------------------------|

| saveNote() | deactivatePassword()

|------------------------------>|------------------------------>|

| | |

|<------------------------------|<------------------------------|

| submitNote() |

|------------------------------>|

| |

| |

| getSubmittedNotes() |

|------------------------------>|

| |

| |

| |

| |

+-------------------------------+

This communication diagram provides a visual representation of the interactions between objects in a notes and password management system. It can be useful for understanding the flow of messages and for designing and implementing the necessary methods in each class.

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 12 & MODELLING UML CLASS AND SEQUENCE DIAGRAM |
| Submitted By: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

MODELLING UML CLASS AND SEQUENCE DIAGRAM:

CLASS DIAGRAM DESCRIPTION:

A class diagram is a static structure diagram that represents the classes, interfaces, and their relationships in a system. Here's a description of a possible class diagram for a password management system:

User class: This class represents the user who is using the password management system. It contains the user's information such as name, email, and password. It has the following methods:

login(): This method allows the user to log in to the system.

logout(): This method logs out the user from the system.

addPassword(password: Password): This method allows the user to add a new password to the system.

getPasswords(): This method returns a list of all passwords associated with the user.

Password class: This class represents a password object in the system. It contains the password details such as website name, username, and password value. It has the following methods:

getPassword(): This method returns the password value.

changePassword(new\_password: str): This method allows the user to change the password value.

getPasswordDetails(): This method returns the details associated with the password, such as the website name and username.

PasswordManager class: This class is responsible for managing the passwords in the system. It has the following methods:

addUser(user: User): This method adds a new user to the system.

removeUser(user: User): This method removes a user from the system.

getPasswords(user: User): This method returns a list of all passwords associated with a user.

addPassword(user: User, password: Password): This method allows a user to add a new password to the system.

removePassword(user: User, password: Password): This method allows a user to remove a password from the system.

changePassword(user: User, password: Password, new\_password: str): This method allows a user to change the password value.

Here is a graphical representation of the class diagram:

+--------+

| User |

+--------+

| name |

| email |

| pw |

+--------+

|

|

|

+--------+

| Password |

+--------+

| website |

| username|

| password|

+--------+

|

|

|

+---------------+

| PasswordManager|

+---------------+

| addUser() |

| removeUser() |

| getPasswords()|

| addPassword() |

| removePassword()|

| changePassword()|

+---------------+

SEQUENCE DIAGRAM:

+-----------------+ +--------------+

| User | | Password |

+-----------------+ +--------------+

| | | website |

| | | username |

| | | password |

| | +--------------+

| | |

| | |

| | |

| | |

| | |

| | addPassword() |

| +------------------------>|

| | |

| | |

| | |

| | |

| | |

| | |

| | changePassword() |

| +------------------------>|

| | |

| | |

| | |

| | |

| | |

| | |

| | getPasswordDetails()|

| +<------------------------|

| | |

| | |

| | |

| | |

| | |

| | removePassword() |

| +------------------------>|

| | |

| | |

| | |

| | |

| | |

| | |

| | logout() |

| +------------------------>|

| | |

+-----------------+ |

|

|

|

+------------------+ |

| PasswordManager | |

+------------------+ |

| addUser() | |

| removeUser() | |

| getPasswords() | |

| addPassword() | |

| removePassword() | |

| changePassword() | |

+------------------+ |

| |

| |

| |

| getUserPasswords() |

+---------------------------------->|

| |

| |

| updatePassword() |

+<----------------------------------|

| |

This sequence diagram shows the interactions between the user and the password management system. The user can add a new password, change the password value, get the password details, and remove a password. The user can also log out from the system. The Password Manager class manages the passwords and provides methods for adding, removing, and changing passwords for a user. The get User Passwords() method retrieves all the passwords associated with a user. The update Password() method updates the password value.

|  |  |
| --- | --- |
| Date: |  |
| Experiment no. & Title: | 13 & MODELLING DATA FLOW DIAGRAM |
| Submitted By: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

MODELLING DATA FLOW DIAGRAM:

DATA FLOW DIAGRAM DESCRIPTION:

A data flow diagram (DFD) for notes and password management system can help in understanding the flow of information in the system. The DFD can be divided into four main components: the user, the note manager, the password manager, and the database.

The user is the person who interacts with the system and can create, view, and edit notes as well as manage their passwords.

The note manager is responsible for managing the notes created by the user. It receives inputs from the user and stores the notes in the database. The note manager also retrieves notes from the database when the user wants to view or edit them.

The password manager is responsible for managing the passwords created by the user. It receives inputs from the user and stores the passwords in the database. The password manager also retrieves passwords from the database when the user wants to view or edit them.

The database is the central component that stores all the notes and passwords. It receives data from the note and password managers and stores them in a structured format.

The data flow in the system can be described as follows:

The user creates a note or password by entering the information into the system.

The note or password information is sent to the respective manager (note manager or password manager).

The manager checks the information for errors and then stores the information in the database.

If the user wants to view or edit a note or password, the manager retrieves the information from the database and sends it back to the user.

If the user wants to delete a note or password, the manager removes the information from the database.

Overall, the DFD for notes and password management system shows how data flows between the user, the note and password managers, and the database. It can be helpful in understanding the system's functionality and identifying any potential issues or inefficiencies.

DATA FLOW DIAGRAM:

+----------------+ +----------------+

| User | | Database |

+----------------+ +----------------+

| |

| Create/Update Notes |

+------------------------------->+

| |

| |

| Create/Update Passwords |

+------------------------------->+

| |

| |

| View Notes/Passwords |

+<------------------------------+

| |

| |

| Delete Notes/Passwords |

+------------------------------->+

| |

+----------------+ +----------------+

| Note Manager | | Password Manager|

+----------------+ +----------------+

| |

| Validate Notes/Passwords |

+------------------------------->+

| |

| Store Notes/Passwords |

+------------------------------->+

| |

| Retrieve Notes/Passwords |

+<-------------------------------+

| |

| Delete Notes/Passwords |

+------------------------------->+

| |

+----------------+ +----------------+

|  |  |
| --- | --- |
| Date : |  |
| Experiment no. & Title: | 14 & IMPLEMENTATION |
| Submitted by: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

IMPLEMENTATION:

|  |  |
| --- | --- |
| Date : |  |
| Experiment no. & Title: | 15 & CODE |
| Submitted by: | Vartika Singh (RA2111003020241)  Akriti Deo (RA2111003020309) |

CODE:

import getpass

import bcrypt

def authenticate\_user():

username = input("Enter your username: ")

password = getpass.getpass("Enter your password: ")

# Look up user in database

hashed\_password = get\_hashed\_password(username)

if not hashed\_password:

return False

# Verify password

return bcrypt.checkpw(password.encode('utf-8'), hashed\_password)

def get\_hashed\_password(username):

# Look up user in database and retrieve hashed password

hashed\_password = b'$2b$12$tI0u.7tQ2YtjE7I2chskZOx/dEzbgj96ilrgBIM0IgAGNMuBjH1Fq'

return hashed\_password

from flask import Flask

from flask\_security import Security, SQLAlchemyUserDatastore, UserMixin, RoleMixin

# Define database models for users and roles

class Role(db.Model, RoleMixin):

id = db.Column(db.Integer(), primary\_key=True)

name = db.Column(db.String(80), unique=True)

description = db.Column(db.String(255))

class User(db.Model, UserMixin):

id = db.Column(db.Integer, primary\_key=True)

email = db.Column(db.String(255), unique=True)

password = db.Column(db.String(255))

active = db.Column(db.Boolean())

confirmed\_at = db.Column(db.DateTime())

roles = db.relationship('Role', secondary=roles\_users,

backref=db.backref('users', lazy='dynamic'))

# Define roles and permissions

roles\_users = db.Table('roles\_users',

db.Column('user\_id', db.Integer(), db.ForeignKey('user.id')),

db.Column('role\_id', db.Integer(), db.ForeignKey('role.id')))

roles = {

'user': {'permissions': ['read']},

'admin': {'permissions': ['read', 'write']},

}

# Set up Flask-Security

app = Flask(\_name\_)

app.config['SECRET\_KEY'] = 'super-secret'

app.config['SECURITY\_PASSWORD\_SALT'] = 'another-super-secret'

app.config['SECURITY\_PASSWORD\_HASH'] = 'bcrypt'

app.config['SECURITY\_REGISTERABLE'] = False

app.config['SECURITY\_SEND\_REGISTER\_EMAIL'] = False

user\_datastore = SQLAlchemyUserDatastore(db, User, Role)

security = Security(app, user\_datastore)

# Create users and roles

with app.app\_context():

db.create\_all()

for role\_name, role\_data in roles.items():

role = user\_datastore.find\_or\_create\_role(role\_name, description='')

for permission in role\_data['permissions']:

user\_datastore.add\_role\_to\_permission

from passlib.hash import bcrypt

# Hash a password

password = 'my\_password'

hashed\_password = bcrypt.hash(password)

# Verify a password

password\_to\_check = 'my\_password'

is\_valid\_password = bcrypt.verify(password\_to\_check, hashed\_password)