**Sprint 2 report**

**Project Title: Graphical Password.**

**Name of the Group: PictPass**

**Semester: Spring 2024**

**Group Number: Group 4**

**Team member:** Andrew O’Berry, Thuan Nguyen, Sarah Mustafa**, Congqi Xu,** Amanda Khor

Section 1: Scheduling and planning table for Sprint 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Email | Task | Duration  (hours) | Dependency | Due  Date | Evaluation |
| Thuan Nguyen | tnguyen591@student.gsu.edu | Use case diagrams | 0 | None | 14/2/24 |  |
| Andrew O’Berry | aoberry1@student.gsu.edu | activity diagram fixes, section 5.1  database schema | 5 | Old files | 14/2/24 |  |
| Amanda Khor | akhor2@student.gsu.edu | Use case diagrams | 0 | None | 14/2/24 |  |
| Xu Congqi | cxu6@student.gsu.edu | Built report, html | 5 | List of use cases | 14/2/24 |  |
| Sarah Mustafa | smustafa2@student.gsu.edu | Context diagram fix | 0 | None | 14/2/24 |  |

Section 2: Problem Statement

What is your product, on a high level?

A website gatekeep with image selection order as a cybersecurity method. The website has a number of icon sized images that are used in a particular order to form a password used by the website. The icons or images should be able to be added to, replaced, or modified to increase security.

Whom is it for?

It is for anyone who runs a website. This could be for anyone who wants to install or use it on their website.

What problem does it solve?

Provides security for users who are logging in and the data contained behind the security. It adds a new method for protecting data that hasn't been used publicly yet. Using a secure method that is not currently widespread use increases security as it will take attackers some time to work out how it works and build a method or tool around the security.

What alternatives are available?

There are image rotating games (make the hippo right-side up), choosing images with objects game, write the poorly drawn letters and numbers, etc.

Why is this project compelling and worth developing?

It’s always important to develop new methods and ways of protecting data to help mitigate brute force and bot attacks. Attacks and security methods are always changing. If version 1.0 gets hacked, you need version 1.1 ready to take 1.0’s place. Any method that could provide lasting security, that is a novel approach, would be worth developing.

Describe the top-level objectives, differentiators, target customers, and scope of your product.

To provide a new security method/product that brings a new approach to login security. It’s different because most people haven't seen an image choice order-based security method. It would be developed for anyone who has sensitive data that needs protecting but wants a new option from the ones that are already available.

What are the competitors and what is novel in your approach?

Captcha and related are similar in structure and use. There are always ways around or ways to trick the security of applications, especially those that contain private personal data, which includes social security numbers, credit card numbers, banking information, email accounts, heath data, etc. Evolving new methods and changing old methods is paramount to keeping data safe and secure.

Make it clear that the system can be built, making good use of the available resources and technology:

20 images would provide a password complexity of 20! Image choice-based passwords would be a departure from standard these days that require letters, numbers, symbols, and case sensitive. Any method that could provide reasonable security that isn’t available large scale would be worthwhile. Ai and other methods eventually find away around security. With a new model, it would take some time for someone to develop a method around or build an automated workaround.

What is interesting about this project from a technical point of view?

The security app should be written in a way for it to be modified, scaled easily, or improved. It’s always interesting to see a new method of solving an old problem. Personal information and user location data is very valuable to advertisers and product manufacturers. If you could safeguard more of your personal data with password methods not available to some of these companies, it could help mitigate the loss of your personal data.

Do you have a client login and an admin login?

It should have a client log in for making user password, but also an admin login for changing and enforcing new password requirements and standards. In some cases, an admin logged into the service would be required for making a password reset. An admin reset would be required for those who can’t figure it out on their own or those who have reached the limit of either password attempts or password resets.

Section 3: Context Diagram

A piece of paper with writing on it

Description automatically generated

Section 4: Activity Diagram

A diagram of a computer

Description automatically generated

Section 5: System Requirements

Section 5.1: Use Cases

List all the use cases you have for the project. For the class project, we will need at least 15 use cases to make one project. Make sure you follow the format.

**(√)Use Case No: 1**

Use Case Name: User changes their password.

Actors: User, User panel

Description: The user logs in. The user visits their user profile management panel. The user then changes their password. The system then requires the user to log in again.

Alternate Path: The user could not complete the password change or would try an option that wouldn’t correctly change their password. The user will receive an error message telling them that their password was not changed and why it wasn’t changed. The user is then prompted to try again.

Exception Path: [Exceptional path or an error handling path when a certain use case is executed]

Pre-Condition: **User has an account, is currently logged in, and is viewing the user control panel screen.**

Post-condition: **The user’s password is changed for their account and is prompted to log in again using their new password.**

**(√)Use Case No: 2**

Use Case Name: User tries to change their password but tries to reuse an old password.

Actors: User, user database, image database

Description: A user attempts to change their password to a password they’ve already used before. The system will notify the user that they are attempting to use an old password and must construct a new password.

Alternate Path: The user won’t change their password and will continue to use the old password. Another path is for the user to not change their password and the user will be excluded from the system until they change their password to an acceptable password.

Exception Path: If the user doesn’t change their password, the system must lock them out of the system.

Pre-Condition: The user needs to change their password.

Post-condition: The user has either changed their password or has been excluded from the system.

**(√)Use Case No: 3**

Use Case Name: User tries to change their password, but new password doesn’t meet requirements.

Actors: User, user panel, image database

Description: A user tries to change their password but has not chosen an adequate number of images to construct a password.

Alternate Path: A user is trying to change their password but tries to reuse images in their password.

Exception Path: If a suitable password isn’t constructed by the end of the use case, the user is locked out of the system, unless their original password is still valid.

Pre-Condition: A user decides to change their password or is prompted by the system to change their password.

Post-condition: The user correctly changes their password.

**(√)Use Case No: 4**

Use Case Name: Someone tries to login with an email address not on file.

Actors: User, login page

Description: Someone tries to login but fails because the email address they provided isn’t on file.

They are prompted to register for an account.

Alternate Path: The someone decides to not create a user account.

Exception Path: The user is blocked from registering an account.

Pre-Condition: A user attempts to login without having an account.

Post-condition: The user either creates an account or chooses to not create an account.

**(√)Use Case No: 5**

Use Case Name: User updates their user profile with new information.

Actors: User, user account panel

Description: A user logs in and visits their user profile management panel to update their profile with a new email address.

Alternate Path: The user changes their password, profile picture, or any other changeable user information.

Exception Path: User tries to change information without being logged in.

Pre-Condition: Can remember their username and can remember their current password.

Post-condition: The user has updated the email address on file and has verified the email address by following a link that was emailed to them (the user).

**(√)Use Case No: 6**

Use Case Name: User tries to update their email but formats their input wrong.

Actors: User, user account panel

Description: A user is trying to update their email address on file with the system. They visit the user control panel and attempt to update the email address the system has. The user tries to enter an email address in a format that the system does not recognize.

Alternate Path: The user does add the @(at) symbol. The user enters an email that is incorrect.

Exception Path: The email address is correctly formatted.

Pre-Condition: User prompted to change email address on file.

Post-condition: User is prompted to try entering their new email address again, but no actual changes have been made to the user’s database file.

**(√)Use Case No: 7**

Use Case Name: A new user creates a new account.

Actors: User, login page, home page

Description: A user that has not visited the website before. New users are required to register for an account before viewing any content. The user then signs up with their email address and creates an appropriate password for their new account. The user then receives an email at the address entered to verify their email address on file with the account.

Alternate Path: The user decides to not create a new account and cannot view any content of the website outside of the home page and login page.

Exception Path:

Pre-Condition: A user wants to create a new account using an email address not currently in the system.

Post-condition: The user now has an account and is prompted to verify it by receiving an email with a code to the email address that they registered the new account with.

**(√)Use Case No: 8**

Use Case Name: A new user verifies their new account.

Actors: User, login page, home page, user’s email account

Description: A user has just created a new account. They then must verify their email address before gaining access to the rest of the system.

Alternate Path: A user does not verify their account inside the window for verification and can resend an account verification email to their email address that is on file with their user account.

Exception Path: The user does not verify their account within 24 hours and is deleted from the system.

Pre-Condition: A user has created a new account, the user has set an acceptable password, and is logging in for the first time.

Post-condition: The user verifies their email address on file with their account and is granted access to the rest of the system at large.

**(√)Use Case No: 9**

Use Case Name: Incorrect password too many times.

Actors: User, login page

Description: A user has attempted to login but has entered an incorrect password 3 times.

Alternate Path: After 3 attempts the user is prompted to reset their password. The system flags this account for auditing.

Exception Path: The user quits before using all of their attempts OR correctly enters their password.

Pre-Condition: User has tried to login too many times with an incorrect password.

Post-condition: The user’s account is locked and is required to reset their password.

**Use Case No: 10**

Use Case Name: Admin changes password.

Actors: User, admin, admin panel

Description: The user is unable to reset their password on their own. The admin sets a temporary password for the account and emails the temporary password to the user’s email address on file.

Alternate Path: The admin could choose to not reset the user's password, especially if the user cannot verify their email address or cannot visit their email account for any reason.

Exception Path: If the administrator determines that the user is not a genuine user, they can lock the user’s account until that user contacts the system administrator.

Pre-Condition: User asks an administrator to assist them in changing their password and has verified some personal information.

Post-condition: The user receives an email with the temporary password and instructions on how to access their account and set a new password.

**Use Case No: 11**

Use Case Name: Too many incorrect logins.

Actors: User, login screen, password database.

Description: A user has had to reset their password too many times in a specific time period. The

system locks their account and will not let them reset their password. The user must get in touch with the system administrator and provide information that verifies the user’s identity. The admin then sends an email to the user’s email, that is on file, that will include a password reset link.

Alternate Path: The user doesn’t get in touch with the admin and the account remains locked.

Exception Path: If the user attempts to bypass the account lock by using an unauthorized method or if the system detects suspicious activity during the lockout period, the account may be subjected to additional security measures, including an extended lockout period or alerting system security personnel.

Pre-Condition: User has reset their password more than two times daily or 5 times per week.

Post-condition: User account is locked or the user has contacted the admin for an identity

verification.

**Use Case No: 12**

Use Case Name: User already registered.

Actors: User, new user registration page, user database

Description: A user who already has an account is trying to register an account with an email address that is already on file and that email is linked to a specific user.

Alternate Path: The user stops trying to register a new account and backs out to the login page. The user then logs in or resets their password if they cannot remember their password.

Exception Path: If the email validation or user notification system fails, the user may not receive the prompt indicating that the email is already in use, leading to confusion or repeated registration attempts until the issue is resolved by technical support.

Pre-Condition: Email already in user database

Post-condition: The user logs into their account by either remembering the password associated with the account or they have reset their password.

**Use Case No:13**

Use Case Name: User tries to view website content without being logged in.

Actors: User, home page, user database

Description: A user visits the website and tries to visit the website while not being logged in.

Alternate Path: The user logs in and is then granted access to the content.

Exception Path: If the system fails to redirect the user to the login page or display the appropriate message due to a glitch or error, the user may be stuck on a blank or error page until the issue is manually resolved or the page is refreshed.

Pre-Condition: User logged out. The user tried to visit content that is perhaps in their history.

Post-condition: User is blocked from having access until they log in.

**(√)Use Case No: 14**

Use Case Name: User does not verify their account.

Actors: User, User database

Description: A user creates an account, but 24 hours passes without the user verifying their account.

Alternate Path: The user verifies their account.

Exception Path: If the user does not verify their account, their account is deleted from the system.

Pre-Condition: User has account, account is unverified, 24 hours has not passed.

Post-condition: The user’s account is verified and therefore active.

**Use Case No:15**

Use Case Name: User does not visit for extended period.

Actors: User, user database

Description: A user who has an account at the website but has not visited in 365 days.

Alternate Path: The user requests account closure from an administrator.

Exception Path: If the system's automated process for identifying and deleting inactive accounts fails, the user's account remains dormant but intact within the database, potentially requiring manual intervention for resolution.

Pre-Condition: User has account. User has not logged in one year.

Post-condition: The user’s account is deleted from the system and their password, and their password history is deleted.

Section 5.2: Requirement number:

Mention the requirements for each use case. Since we will have 15 use cases, you will need 15 requirements. The requirements must be written in the following format:

**Use Case number: 1 - User changes their password**

**Introduction**: Users can securely update their account password through a interface. It is essential for maintaining account security and user privacy, allowing users to change their passwords regularly or in case they believe their current password has been compromised.

**Inputs**: Current Password: The user's existing password that needs to be verified before allowing a change.

New Password: The password the user wants to change to, meeting the system's security criteria (length, complexity).

New Password Confirmation: A repeat of the new password to ensure there are no typographical errors.

**Requirements Description**:

* Functional Requirements:
  + Authentication: The system must verify the user's current password before allowing a password change.
  + Password Validation: The new password must adhere to defined security standards (e.g., minimum length, requires numbers, uppercase and lowercase letters, and special characters).
  + Confirmation Process: The user must confirm their new password by entering it twice to minimize the risk of typographical errors.
  + Success Feedback: Upon successful password change, the system should notify the user that their password has been updated.
  + Security Logging: The system should log the password change event for auditing purposes.
* Non-Functional Requirements:
  + Usability: The interface for changing the password should be user-friendly and accessible.
  + Security: Password transmission over the network must be encrypted.
  + Performance: The password change process should be completed within a reasonable time frame to ensure a good user experience.
  + Compatibility: The password change feature should be supported across all major browsers and devices used to access the platform

**Outputs**:

* Successful Password Change Notification: A message or email notification confirming that the user's password has been successfully changed.
* Error Messages: Appropriate feedback in case of errors (e.g., current password is incorrect, the new password does not meet security requirements).
* Audit Log Entry: An entry in the system's security log indicating the date and time of the password change, along with the user ID.

**Use Case number: 2 - User tries to change their password but tries to reuse an old password**

**Introduction**: This use case describes the process and system behavior when a user attempts to change their password but tries to use a password that they have previously used. It covers the system's response to prevent password reuse as part of security measures to enhance account safety.

**Inputs**:

* User ID: The unique identifier of the user attempting to change their password.
* Old Password: The current password that the user is using to authenticate the password change request.
* New Password& New Password Confirmation: The new password that the user wants to set.

**Requirements** **Description**:

* Functional Requirements:
  + Authentication Check: Validate the user's current password to authenticate the user's identity before proceeding with the password change.
  + Password History Validation: The system must check the new password against a history of previously used passwords for the account. The depth of this history (e.g., last 5 passwords) is defined by the system's security policy.
  + Prohibition of Old Passwords: If the new password matches any in the history, the system must reject the password change request and inform the user that the new password cannot be the same as any previously used passwords.
  + Security Requirements for New Password: Besides not being a previously used password, the new password must meet all other security criteria set by the system (complexity, length, etc.).
* Non-Functional Requirements:
  + Security: Ensure that all password transmissions are securely encrypted.
  + Usability: Provide clear, user-friendly feedback and guidance for choosing a new password that complies with all system requirements.
  + Performance: The check against the password history and the execution of the password change should happen swiftly to maintain a smooth user experience.
  + Scalability: The system should efficiently handle password history checks even as the number of users and password change requests grows.

**Use Case number: 3 - User tries to change their password, but new password doesn’t meet** **requirements**

**Introduction:** This use case outlines the scenario where a user attempts to change their password, but the new password they propose does not meet the predefined security requirements set by the system.

**Inputs:**

* User ID: The unique identifier of the user attempting to change their password.
* Old Password: The current password that the user is using to authenticate the password change request.
* New Password& New Password Confirmation: The proposed new password the user wishes to set.

**Requirements Description:**

* Functional Requirements:
  + Authentication Verification: The system must verify the current password to authenticate the user’s identity before processing the password change.
  + Password Strength Validation: The new password must be checked against the system’s password policy, which includes requirements for minimum length, complexity (inclusion of uppercase and lowercase letters, numbers, and special characters), and possibly not containing user-specific information (like username or birthdate).
  + Feedback on Non-Compliance: If the new password does not meet the required criteria, the system must inform the user which of the password requirements were not met.
  + Guidance for Compliance: The system should provide guidance or tips on how to create a password that meets the security requirements.
* Non-Functional Requirements:
  + Security: Ensure the secure transmission of all password data.
  + Usability: Provide clear, concise, and immediate feedback on the password’s compliance with the security policy to enhance user experience.
  + Performance: The evaluation of the new password against the security requirements should occur rapidly to prevent user frustration.
  + Accessibility: Ensure that instructions and feedback are accessible to all users, including those with disabilities.

**Outputs:**

* Success Message: A confirmation message that the password has been successfully changed. This message is output only when the New Password passes all the strength validation criteria.
* Error Message with Specific Feedback: A detailed error message that specifies which part(s) of the password strength requirements were not met by the proposed New Password. For example, "Password must be at least 8 characters long and include a mix of letters, numbers, and special characters.

**Use Case number: 4 - Someone tries to login with an email address not on file.**

**Introduction:** This use case addresses the scenario where an individual attempts to sign in to the system using an email address that does not exist in the user database.

**Inputs:**

* Email Address: The email address entered by the individual trying to log in.
* Password: The password entered by the individual, which is associated with the email address they are attempting to use for login.

**Requirements Description:**

* Functional Requirements:
  + Email Verification: The system must check the entered email address against those in the user database to verify if an account exists with that email.
  + Invalid Email Feedback: If no account exists with the entered email, the system should inform the individual that the login attempt was unsuccessful due to an unrecognized email address.
  + Guidance for Next Steps: The system should provide suggestions for next steps, such as checking the email address for typos, registering for a new account if the user is not yet a member, or accessing a help or support feature for further assistance.
  + Security Measures: Implement security measures to prevent this feature from being used for email enumeration attacks by attackers attempting to identify valid email addresses in the system.
* Non-Functional Requirements:
  + Security: Ensure that the feedback mechanism does not inadvertently confirm the existence of an email address in the system to unauthorized users.
  + Usability: Provide clear, understandable feedback that guides the user towards resolving their issue, enhancing the overall user experience.
  + Performance: The system should rapidly verify the email address to ensure a quick response time, minimizing user wait times.
  + Accessibility: Ensure that the error messages and guidance provided are accessible to all users, including those with disabilities.

**Outputs:**

* Login Failure Notification: A message displayed to the individual indicating that the login attempt was unsuccessful due to the email address not being recognized, without specifying whether the email address does not exist or the password was incorrect.
* Guidance Message: Suggestions provided to the user on how to proceed, including checking for typos, registering a new account, or seeking help from the support team.
* Security Log Entry: Optionally, an entry in the system's security log documenting the failed login attempt, including the time and the entered email address, to monitor for suspicious activity patterns.

**Use Case number: 5 - User updates their user profile with new information.**

**Introduction:**

This use case facilitates users in keeping their profile information current and accurate by allowing them to update their personal details, such as contact information, preferences, and other relevant data. It is critical for ensuring that the system has the most up-to-date information for communication, personalization, and security purposes.

**Inputs:**

User Authentication Details: Credentials or session token to verify the user's identity.

Updated Information: Specific details the user wishes to update, which could include:

Name, Email address, Contact number, Address, Preferences (e.g., communication preferences, language settings), Password (if applicable).

**Requirements Description:**

* Functional Requirements:
  + User Authentication: The system must verify the identity of the user making the update to prevent unauthorized changes.
  + Validation of New Information: The system must validate the new information for correctness and adherence to any format or policy requirements (e.g., valid email format).
  + Information Update Process: Upon successful validation, the system updates the user's profile with the new information.
  + Confirmation of Update: The user receives confirmation that their profile has been successfully updated.
  + Audit Trail: The system logs the update activity for security and auditing purposes, including what information was changed and the time of the update.
* Non-Functional Requirements:
  + Security: Ensure that all data transmissions are secure and that sensitive information is stored securely.
  + Usability: The interface for updating profile information should be intuitive and user-friendly.
  + Performance: The system should process updates promptly to ensure a smooth user experience.
  + Privacy: The system must comply with relevant data protection regulations, ensuring that user data is handled and stored securely and privately.

**Outputs:**

* Update Confirmation Notification: A message or email to the user confirming that their profile has been updated with the new information.
* Updated Profile: The user's profile reflects the changes made.
* Audit Log Entry: An entry in the system's audit log detailing the user's update activity, including timestamp and nature of the changes, while ensuring sensitive information is not exposed.

**Use Case number: 6 - User tries to update their email but formats their input wrong.**

**Introduction:**

This use case addresses scenarios where a user attempts to update their email address in their profile but enters it in an incorrect format. It ensures the system can guide users to correct this mistake, maintaining the integrity of contact information in the user database. Proper email format validation is essential for communication, security, and user identification purposes.

**Inputs:**

* User Authentication Details: Credentials or a session token to verify the user's identity before allowing changes to be made.
* New Email Address: The email address the user wishes to update to, which may be incorrectly formatted.

**Requirements Description:**

* Functional Requirements:
  + Email Format Validation: The system must validate the new email address against standard email formatting rules (e.g., "username@domain.com").
  + Error Notification: If the email format is incorrect, the system should inform the user of the specific formatting error.
  + Guidance for Correction: Provide guidance or examples of correct email formatting to help the user correct their input.
  + Re-input Opportunity: Allow the user to re-enter their email address correctly without having to navigate away from the current screen or process.
  + Update Process Continuation: Once a correctly formatted email address is provided, proceed with the standard email update process, including any necessary verification steps.
* Non-Functional Requirements:
  + Usability: The process for updating an email address, including error correction, should be user-friendly and intuitive.
  + Performance: The system should validate the email format in real-time or near-real-time to quickly provide feedback to the user.
  + Security: Ensure secure handling of the new email address input and any communication sent to it.
  + Accessibility: Error messages and instructions should be clear, concise, and accessible to all users, including those with disabilities.

**Outputs:**

* Format Error Notification: A clear and informative message indicating that the email address format is incorrect, with details on how to correct it.
* Corrective Guidance: Instructions or an example of a properly formatted email address to assist the user in making the necessary correction.
* Successful Update Confirmation: Upon entering a valid email format and completing any further verification steps, a confirmation that the email address has been successfully updated.

**Use Case number: 7 - A new user creates a new account.**

**Introduction:**

This use case outlines the process for a new user to register and create a new account on the platform. It is essential for expanding the user base and ensuring that new users can easily join and start using the service. The process includes collecting user information, validating inputs, and setting up credentials.

**Inputs:**

* User Information: Personal and contact information required for account creation, including:

Name, Email address, Password, Optional details such as phone number, address, etc.

* Consent: Agreement to the terms of service and privacy policy.

**Requirements Description:**

* Functional Requirements:
  + Information Collection: The system must provide a form for the user to input their details.
  + Input Validation: Validate the inputs for correctness and completeness. Email addresses must follow a standard format, and passwords must meet security criteria.
  + Duplicate Check: The system checks if the email address is already associated with an existing account to prevent duplicate registrations.
  + Consent Verification: Ensure that the user has agreed to the terms of service and privacy policy.
  + Account Creation: Upon successful validation, create the user's account and store their information securely in the user database.
  + Confirmation and Welcome: Send a confirmation email to the user with a welcome message and possibly instructions for next steps, such as verifying the email address or logging in for the first time.
* Non-Functional Requirements:
  + Usability: The registration interface should be intuitive, guiding the user through the process step by step.
  + Performance: The system should process the registration quickly to provide a smooth user experience.
  + Security: Ensure that all personal information and credentials are transmitted and stored securely, implementing encryption where necessary.
  + Accessibility: Make sure the registration form and process are accessible to users with disabilities, following best practices for web accessibility.

**Outputs:**

* Account Creation Confirmation: A message displayed to the user upon successful account creation, indicating that they can now log in to their new account.
* Welcome Email: An email sent to the user's provided address, confirming their registration and including any additional steps they need to take or useful information to get started.
* Error Messages: In case of issues (e.g., input validation fails, email already in use), appropriate error messages are displayed, guiding the user on how to resolve them.

**Use Case number: 8 - A new user verifies their new account.**

**Introduction:**

This use case details the process by which a new user verifies their account after registration. Account verification is a critical step to confirm the authenticity of the user's email address, enhancing security and reducing the risk of spam or fraudulent accounts. It typically involves the user clicking a verification link sent to their registered email address.

**Inputs:**

* Verification Link: A unique URL sent to the user's email address upon registration.
* User Email Address: The email address provided by the user during the account registration process.

**Requirements Description:**

* Functional Requirements:
  + Send Verification Email: Immediately after account registration, the system sends a verification email to the user's provided email address, containing a unique verification link.
  + Verification Link Activation: The user must click the link to verify their email address. The link should direct the user to a verification page and include a unique token to prevent unauthorized access.
  + Email Address Confirmation: Upon clicking the verification link, the system validates the token and confirms the email address associated with the new account.
  + Account Activation: Once the email address is verified, the user's account is activated, allowing full access to the system's features.
  + Feedback to User: After successful verification, the system provides feedback to the user, such as a confirmation message or redirection to the login page.
* Non-Functional Requirements:
  + Security: The verification process must securely handle the verification link and token to prevent misuse or interception.
  + Usability: The verification email should clearly instruct the user on how to complete the verification process, with a straightforward and accessible verification link.
  + Reliability: The system should reliably send the verification email immediately after registration, ensuring that the email reaches the user's inbox and not the spam folder.
  + Performance: The verification process should be quick and efficient, with immediate feedback provided to the user upon completion.

**Outputs:**

* Confirmation Message: A message displayed to the user upon successful verification of their email address, confirming that their account is now active.
* Verification Email: An email sent to the user's provided address immediately after registration, containing the unique verification link.
* Error Message: If the verification link is invalid, expired, or already used, an appropriate error message is displayed to the user, possibly with instructions on how to request a new verification link.

**Use Case number: 9 - Incorrect password too many times.**

**Introduction:**

This use case addresses the scenario where a user enters an incorrect password multiple times while attempting to log in. This situation raises security concerns, such as potential unauthorized access attempts, and necessitates a system response to protect the user's account.

**Inputs:**

* User ID/Username: The identifier the user inputs to log in.
* Password Attempt: The incorrect password entered by the user.
* Attempt Counter: A system-maintained count of consecutive incorrect password attempts.

**Requirements Description:**

* Functional Requirements:
  + Limit on Attempts: The system should define a maximum number of incorrect password attempts (e.g., 5 attempts) before taking further action.
  + Increment Attempt Counter: With each incorrect password entry, the system increments the attempt counter.
  + Account Lock: Upon reaching the maximum number of incorrect attempts, the system temporarily locks the user's account to prevent further access attempts.
  + User Notification: Notify the user of the account lock via the UI at the time of lockout and, if possible, through an email alert.
  + Reset Mechanism: Provide a method for the user to unlock their account, which may involve waiting a set period, contacting customer support, or verifying their identity through a secure process.
  + Security Logging: Log the incident of multiple incorrect password attempts for audit and security monitoring purposes.
* Non-Functional Requirements:
  + Security: The account lock and notification mechanisms must be secure to prevent exploitation or leakage of user information.
  + Usability: Instructions for resolving the account lock (e.g., unlocking the account or resetting the password) should be clear and easily accessible.
  + Performance: The system should quickly recognize and respond to multiple incorrect passwords attempts to minimize potential security risks.
  + Reliability: The account lock and user notification processes should function reliably under all expected user scenarios and system states.

**Outputs:**

* Account Lock Notification: A clear and immediate message to the user within the login interface indicating that their account has been locked due to too many incorrect password attempts.
* Email Alert: An email sent to the user's registered email address, informing them of the account lock and providing instructions on how to unlock their account.
* Security Log Entry: An entry in the system's security log documenting the date, time, and details of the account lock event.

**Use Case number: 10 - Admin changes password.**

**Introduction:**

This use case describes the process by which an administrator can change a user's password. This is typically required in scenarios where a user has forgotten their password and cannot recover it through standard means, or for administrative reasons such as security breaches. It ensures that users can regain access to their accounts with administrative assistance while maintaining system security.

**Inputs:**

* Administrator Credentials: Verification of the administrator's identity to ensure only authorized personnel can change passwords.
* User ID: The identifier of the user account for which the password needs to be changed.
* New Password: The new password to be set for the user account, optionally generated by the administrator or the system.

**Requirements Description:**

* Functional Requirements:
  + Administrator Authentication: The system must verify the credentials of the administrator to authorize the password change operation.
  + User Account Verification: Ensure the user account exists and is eligible for a password change by the administrator.
  + Password Generation and Validation: The new password must meet the system's security criteria. The system can also offer an option to automatically generate a secure password.
  + Password Change Execution: The system updates the user's account with the new password.
  + User Notification: Notify the user of the password change, typically via email, including instructions for next steps or security advice.
  + Audit Logging: Record the password change event in an audit log, noting the administrator's ID, the affected user account, and the timestamp.
* Non-Functional Requirements:
  + Security: Ensure the entire process is secure, protecting the new password and both the user's and administrator's information.
  + Usability: Provide a simple and efficient interface for administrators to change passwords, minimizing the risk of errors.
  + Reliability: The system must reliably execute the password change and deliver the notification to the user without fail.
  + Compliance: Comply with relevant privacy and security regulations governing the handling of user information and password management.

**Outputs:**

* Password Change Confirmation: A message or confirmation screen for the administrator indicating the password has been successfully changed.
* User Notification: An email or other form of notification sent to the user, informing them of the password change and any actions they need to take.
* Audit Log Entry: A new entry in the system's audit log documenting the details of the password change event.

**Use Case number: 11 - Too many incorrect logins.**

**Introduction:**

This use case outlines the system's response to multiple incorrect logins attempts by a user. It is designed to enhance security by detecting and mitigating potential unauthorized access attempts, while also providing a means for legitimate users to regain access to their account.

**Inputs:**

* User ID/Username: The identification provided by the user attempting to log in.
* Password Attempt: The password entered by the user during the login attempt.
* Login Attempt Counter: A counter tracking the number of consecutive unsuccessful login attempts.

**Requirements Description:**

**Use Case Number: 11 - Too Many Incorrect**

* Functional Requirements:
  + Threshold for Attempts: Define a maximum number of allowed incorrect login attempts (e.g., 5 attempts) before taking further action.
  + Attempt Tracking: Increment the login attempt counter with each incorrect password entry.
  + Account Lockout: Automatically lock the user's account upon reaching the threshold of incorrect attempts to prevent further login attempts.
  + Notification: Inform the user of the account lockout at the login interface and, if possible, via email, explaining the situation and the steps to unlock the account.
  + Account Recovery Option: Offer the user a secure method to unlock their account, which may include answering security questions, using a one-time password sent to their email, or contacting customer support.
  + Security Logging: Log the incident of multiple incorrect login attempts for security monitoring and potential further investigation.
* Non-Functional Requirements:
  + Security: Protect the user's account from brute force attacks while ensuring legitimate users can recover their account with minimal inconvenience.
  + Usability: Provide clear, user-friendly instructions for account recovery to ensure legitimate users are not unduly penalized.
  + Performance: Quickly detect and respond to multiple incorrect login attempts to maintain system security.
  + Reliability: Ensure the account lockout and recovery mechanisms function correctly under all expected operational scenarios.

**Outputs:**

* Account Lockout Notification: A message displayed to the user upon the final unsuccessful login attempt, indicating their account has been locked and providing instructions for account recovery.
* Email Notification: An email sent to the user's registered email address, detailing the account lockout and the steps they can take to regain access.
* Security Log Entry: An entry in the system's security log noting the date, time, and details of the lockout event, including the user ID and the IP address from which the failed login attempts were made.

**Use Case number: 12 - User already registered.**

**Introduction:**

This use case addresses the scenario where an individual attempts to register a new account using an email address that is already associated with an existing account. It aims to prevent duplicate accounts and guide users towards account recovery or login if they have forgotten their account details.

**Inputs:**

* Email Address: The email address entered by the individual attempting to register.
* Registration Details: Other personal details entered during the registration attempt, such as name, password, etc.

**Requirements Description:**

* Functional Requirements:
  + Email Address Check: The system must verify if the entered email address is already associated with an existing account.
  + Duplicate Notification: If the email is found to be associated with an existing account, notify the individual of this fact and prevent the creation of a new account with the same email address.
  + Login Redirect: Suggest that the individual logs in using the existing account associated with the email address.
  + Password Recovery Option: Provide an option for password recovery in case the individual does not remember the password for the existing account.
  + Registration Cancellation: Cancel the current registration process if the email address is already in use.
* Non-Functional Requirements:
  + Usability: Ensure that notifications and suggestions provided to the user are clear and helpful, guiding them towards the next steps (login or password recovery).
  + Security: Safeguard the privacy and security of existing accounts by not revealing too much information about the account associated with the email address.
  + Performance: The check for existing email addresses should be quick to maintain a smooth user experience during registration.
  + Accessibility: Ensure that all users, regardless of their abilities, can understand the notification and are able to take the suggested next steps.

**Outputs:**

* Duplicate Account Notification: A clear message informing the individual that the email address is already associated with an existing account and suggesting appropriate next steps (e.g., login, password recovery).
* Password Recovery Option: A link or instructions for initiating the password recovery process for the existing account.
* Registration Cancellation Confirmation: A confirmation that the current registration attempt has been cancelled, with guidance on how to access the existing account.

**Use Case number: 13 – User not logged in**

**Introduction:**

This use case describes the system behavior when a user attempts to access restricted content or perform actions that require authentication without being logged in. It ensures that only authenticated users can access certain areas or functionalities of the system, enhancing security and user experience by directing users to login or register.

**Inputs:**

* Requested Action or Page: The specific content or functionality the user attempts to access, which requires authentication.
* User State: The current state of the user, in this case, not logged in.

**Requirements Description:**

* Functional Requirements:
  + Authentication Check: The system must verify whether the user is logged in before allowing access to restricted content or functionalities.
  + Redirection to Login: If the user is not logged in, redirect them to the login page, possibly with a message explaining that login is required.
  + Return Intent: Optionally, the system could remember the action or page the user was attempting to access, to facilitate a seamless return after login.
  + Access Denial for Restricted Areas: Ensure that users cannot bypass the login requirement through direct URL access or other methods.
  + Registration Option: Provide an option for users to register for an account if they do not have one.
* Non-Functional Requirements:
  + Usability: The process of redirecting to the login page and explaining the need for authentication should be clear and user-friendly.
  + Security: Ensure that the redirection process and any subsequent login or registration actions are secure, protecting user credentials and information.
  + Performance: The check for user authentication and the redirection process should be fast to maintain a positive user experience.
  + Accessibility: Ensure that all users, regardless of their abilities, can understand the login requirement and navigate the login or registration process.

**Outputs:**

* Login or Registration Prompt: A clear message or prompt on the login page indicating that the user must log in or register to proceed, including any context-specific messages related to the content they were trying to access.
* Redirection: The user is redirected to the login page, with the option to navigate to the registration page if they need to create an account.
* Post-Login Redirection: After successful login, the user is redirected back to the originally requested action or page, if the return intent was preserved.

**Use Case number: 14 - User does not verify their account.**

**Introduction:**

This use case outlines the scenario and system response when a new user fails to verify their account within a given timeframe after registration. Account verification is a crucial step for confirming the authenticity of the user's email address or contact information, enhancing the security and integrity of the user base.

**Inputs:**

User ID/Username: The unique identifier of the user who has registered but not completed the verification process.

Verification Email Sent Timestamp: The timestamp when the verification email was initially sent to the user.

Current Date/Time: For evaluating the duration since the verification email was sent.

**Requirements Description:**

* Functional Requirements:
  + Verification Time Limit: Establish a time limit within which the user must verify their account (e.g., 48 hours from the time of registration).
  + Account Status Monitoring: Monitor the verification status of newly registered accounts to identify those that have not been verified within the specified time limit.
  + Reminder Notification: Send reminder notifications to the user's email address if the account remains unverified close to the deadline.
  + Account Deactivation: Automatically deactivate or delete accounts that have not been verified within the time limit.
  + Reactivation Option: Provide an option for users to request a new verification email if they attempt to log in with an unverified account after deactivation.
* Non-Functional Requirements:
  + Usability: Ensure the verification and reminder processes are user-friendly, with clear instructions on how to complete verification.
  + Performance: The system should efficiently monitor and update the status of user verifications without significant delays.
  + Reliability: Ensure that all reminder and reactivation communications are reliably sent to the user's provided email address.
  + Security: Maintain high security for the verification process to prevent unauthorized account access.

**Outputs:**

Reminder Notifications: Email reminders sent to the user prompting them to verify their account before the deadline.

Account Deactivation Notification: A notification (where applicable) informing the user that their account has been deactivated due to failure to verify.

Reactivation Instructions: Instructions provided to the user (via the login page or email) on how to reactivate their account by requesting a new verification email.

**Use Case number: 15 - User does not visit for extended period.**

**Introduction:**

This use case details the system's approach to handling user accounts that have been inactive for an extended period. It is important for maintaining an active user base, managing resources efficiently, and enhancing security by identifying potentially abandoned or compromised accounts.

**Inputs:**

* User ID/Username: The unique identifier for the user account.
* Last Login Timestamp: The date and time when the user last logged in to the system.
* Current Date/Time: To calculate the duration of inactivity.
* Defined Period of Inactivity: The threshold of inactivity time (e.g., 365 days) after which the system takes action.

**Requirements Description:**

* Functional Requirements:
  + Inactivity Threshold Monitoring: The system must regularly check for accounts that have not logged in for the defined period of inactivity.
  + Notification of Pending Deactivation: Before deactivating the account, send a notification to the user's email address, alerting them of their account's pending deactivation due to inactivity and offering ways to keep their account active.
  + Account Deactivation: Deactivate accounts that have not responded to the inactivity notification within a given timeframe, rendering the account inaccessible until the user takes action to reactivate.
  + Data Retention and Privacy: Ensure compliance with data protection regulations by securely handling the data of inactive accounts, including specifying data retention policies and deletion practices.
  + Account Reactivation Option: Provide users with a clear and secure process to reactivate their account if they return after it has been deactivated.
* Non-Functional Requirements:
  + Security: Safeguard user data for inactive accounts and ensure secure processes for account deactivation and reactivation.
  + Usability: Make the notification and reactivation processes clear and simple for users to follow, minimizing barriers to returning to the platform.
  + Performance: The system should efficiently identify inactive accounts and manage notifications without impacting system performance.
  + Compliance: Adhere to relevant legal and regulatory requirements concerning user data privacy and retention.

**Outputs:**

* Inactivity Notification Email: A message sent to the user's registered email address, warning them of impending account deactivation due to prolonged inactivity, with instructions on how to prevent this.
* Account Deactivation Confirmation: A system-generated notification or log entry confirming the deactivation of an account due to inactivity.
* Reactivation Instructions: Communication to the user (available upon attempt to access the deactivated account) on how to reactivate their account, including any necessary steps for verification or security checks.

Section 5.3: Use Case Diagrams

Choose at least 4 of the 15 use cases from the project to draw use case diagrams.

Section 6: Database Management

A screenshot of a computer

Description automatically generated

We will use MySQL as it supports BLOB. BLOB allows the storing of images in a database and will be easier to manage.

GitHub Link and Screenshots: