A diagram of a system

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-01 Sign-up** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 27/09/2024 |
| Primary Actor: | Guest | Secondary Actors: |  |
| Description: | This use case describes the process by which a new, unregistered user creates an account in the system. The user navigates to the sign-up page, enters required personal information (email, password, firstname, lastname and role), and submits the form. The system verifies the provided information, ensuring it is valid and unique. Upon successful verification, the system creates the new account, stores the data securely, and send an email for further verification (if required). The user is then notified of the successful account creation and can proceed to log in to the system.. | | |
| Trigger: | The use case is triggered when a guest who does not have an account selects the "Sign Up" or "Register" option from the system's login page or menu. | | |
| Preconditions: | PRE-1. The guest must not have an existing account in the system..  PRE-2. The user has access to the internet and a device that can connect to the system.  PRE-3. The sign-up interface (web page) is available and operational.  PRE-4. Google authentication services are available. | | |
| Postconditions: | POST-1. The user account is successfully created and stored in the database.  POST-2. The system sends a confirmation email for verification. | | |
| Normal Flow: | **1.0 Sign-up**   1. The user navigates to the "Sign Up" page of the system. 2. The system displays a form requiring the following details:    1. First name    2. Last name    3. Email    4. Password    5. Re-password    6. Role 3. The user fills out the form and submits the information 4. The system validates the information:    1. Ensures that all required fields are completed.    2. Checks if the email address is unique.    3. Ensures the password meets security requirements (length). 5. The system creates the account and stores the user’s information in the database. 6. The system sends a verification email with a confirmation code to the provided email address. 7. The system displays a success message indicating that the account was created successfully. | | |
| Alternative Flows: | **7.1 Invalid Data Submission:**  If the user submits invalid information (e.g., invalid email format, weak password), the system displays appropriate error messages, and the user is prompted to correct the errors and resubmit.   * 1. **Duplicate Email or Username:**   If the email address or username is already registered in the system, the system displays a message indicating the duplication and prompts the user to choose a different one.  **7.3 Sign-up with Google:**   * The user clicks the "Sign up with Google" button. * The system redirects the user to the Google authentication page. * The user grants permission for the system to access their Google profile information. * The system receives the user's Google profile information (e.g., first name, last name, email). * **7.3.1 New Google User:** If the email address is not already registered, the system creates a new account using the retrieved information and prompts the user to choose a role (required) and role. Steps 5, 6, and 7 of the Normal Flow then apply (database storage, optional verification email, success message). * **7.3.2 Existing User:** If the email address is already registered, the system links the Google account to the existing account and logs the user in. (Skip email verification as they already have an account). | | |
| Exceptions: | **1.0.E1 System Down:**  If the system is unavailable during the sign-up attempt, the system displays a maintenance or error message and logs the incident.  **1.0.E2 Failed Email Delivery:**  If the verification email cannot be delivered, the user is informed and provided with an option to resend the email. | | |
| Priority: | High | | |
| Frequency of Use: | The sign-up process is typically a one-time activity for each new user. The frequency of use is expected to be low for individual users, but in systems with growing user bases, it will be used continuously by different new users. The system should be able to handle concurrent sign-ups during peak traffic times. | | |
| Business Rules: |  | | |
| Other Information: | 1. The system must respond to sign-up requests within 5 seconds. 2. The sign-up page must be available 99.9% of the time. 3. Passwords must be stored securely using industry-standard hashing algorithms. | | |
| Assumptions: | 1.The system has a functioning email service to send out verification emails.  2.The password policies are enforced for account security.  3. The system has proper integration with Google's authentication API. | | |

A diagram of a company

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-02 Login** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 27/09/2024 |
| Primary Actor: | Tenant, host | Secondary Actors: |  |
| Description: | This use case describes the process by which a registered user logs into the system using their credentials. After navigating to the login page, the user enters their email along with the password. The system verifies the provided credentials against stored records. If the information is valid, the user is granted access to the system. If the credentials are invalid, the system prompts the user to try again. | | |
| Trigger: | The use case is triggered when a registered user selects the "Login" option on the system's login page or application interface. | | |
| Preconditions: | PR  E-1. The user has already registered and has an active account in the system.  PRE-2. The user has the correct credentials (username/email and password).  PRE-3. Google authentication services are available. | | |
| Postconditions: | POST-1. The user is successfully logged into the system and has access to authorized features.  POST-2. The system tracks the user's cookies until logout. | | |
| Normal Flow: | 1. **Login** 2. The user/staff navigates to the "Login" page of the system. 3. The system displays fields for entering the following:    1. Email    2. Password   3. The user/staff enters their credentials and submits the form.  4. The system validates the credentials:   * 1. Checks that the email/username exists.   2. Verifies that the password matches the stored password for that account.   5 . The system grants access to the user/staff and redirects them to the dashboard or home page. | | |
| Alternative Flows: | **6.1 Invalid Credentials:**  If the user enters incorrect credentials, the system displays an error message indicating that the email/username or password is incorrect, and the user is prompted to try again.  If user login in wrong screen (login for staff) , the system displays an message and user/staff must redirect to right screen.  **6.2 Forgotten Password:**  If the user has forgotten their password, they can click the "Forgot Password" link. This redirects them to a password recovery process.  **6.3 Login with Google:**   * + The user clicks the "Login with Google" button.   + The system redirects the user to the Google authentication page.   + The user grants permission for the system to access their Google profile information.   + The system receives the user's Google profile information (email).   + **6.3.1 Existing Google-Linked Account:** If the email address is associated with a registered account and linked to Google, the system logs the user in and redirects them to the dashboard/home page.   + **6.3.2 Unlinked Google Account:** If the email address is associated with a registered account but *not* linked to Google, the system will link googleId with this email. | | |
| Exceptions: | **1.0.E1 System Down:**  If the system is unavailable during the sign-up attempt, the system displays a maintenance or error message and logs the incident. | | |
| Priority: | High | | |
| Frequency of Use: | The login use case is expected to be used frequently, as users must log in each time they want to access the system. The frequency will depend on the nature of the system and user activity patterns. | | |
| Business Rules: |  | | |
| Other Information: | none | | |
| Assumptions: | 1. The system has a secure login page with HTTPS encryption.  2. The system stores passwords securely using industry-standard hashing and salting techniques.  3. The system provides a mechanism for users to recover forgotten passwords.  4. The system has proper integration with Google's authentication API. | | |

A blue square with a black oval

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-03 Update Password** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 27/09/2024 |
| Primary Actor: | Registered User (Tenant, Host) | Secondary Actors: | None |
| Description: | This use case describes the process by which a registered user updates their existing password. The user navigates to their account settings and enters their new password twice for confirmation. If the user initially signed up using Google and has not yet set a password, they will not be required to enter a current password. The system checks the new password against any security requirements. If successful, the system updates the password and notifies the user. | | |
| Trigger: | The user selects the "Cập nhật mật khẩu" option within their security settings. | | |
| Preconditions: | PRE-1. The user is logged into the system. | | |
| Postconditions: | POST-1. The user's password is updated in the database.  POST-2. The user is notified of the successful password change. | | |
| Normal Flow: | **3.0 Update Password**   1. The user navigates to the "Account Settings" or "Profile" page. 2. The user selects the "Update Password" option. 3. **IF** the user has a password:    * a. The system displays a form requiring:      + i. Current Password      + ii. New Password      + iii. Confirm New Password    * b. The user enters their current password and the new password twice.    * c. The system validates the information:      + i. Verifies the current password matches the stored password.      + ii. Ensures the new password meets security requirements.      + iii. Checks that the new password and confirm new password fields match. 4. **ELSE** (user does not have a password - Google signup):    * a. The system displays a form requiring:      + i. New Password      + ii. Confirm New Password    * b. The user enters the new password twice.    * c. The system validates the information:      + i. Ensures the new password meets security requirements.      + ii. Checks that the new password and confirm new password fields match. 5. The system updates the password in the database. 6. The system displays a success message confirming the password change. | | |
| Alternative Flows: | **6.1 Incorrect Current Password** (If applicable): If the user enters an incorrect current password, the system displays an error message.  **6.2 New Password Doesn't Meet Requirements**: If the new password doesn't meet the security requirements, the system displays an error message.  6.3 New Passwords Don't Match: If the new password and confirm new password fields don't match, an error message is displayed. | | |
| Exceptions: | **1.0.E1 System Down:**  If the system is unavailable, an error message is displayed.  . | | |
| Priority: | Medium | | |
| Frequency of Use: | Low | | |
| Business Rules: |  | | |
| Other Information: | None | | |
| Assumptions: | The system securely stores passwords using hashing and salting. | | |

A screenshot of a computer screen

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-04 Update Email** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 27/09/2024 |
| Primary Actor: | Registered User (Tenant, Host) | Secondary Actors: |  |
| Description: | This use case describes how a registered user updates their email address associated with their account. The user provides their new email address, and the system sends a verification *code* to the new address. The user must enter this code back into the system to confirm the change. | | |
| Trigger: | The user selects the "Cập nhật email" option within their security settings. | | |
| Preconditions: | PRE-1. The user is logged into the system. | | |
| Postconditions: | POST-1. The user's email address is updated (after verification).:  POST-2. A verification email containing a code is sent to the new email address. | | |
| Normal Flow: | 1. **Update Email** 2. The user navigates to the "Account Settings" or "Profile" page. 3. The user selects the "Update Email" option. 4. The system displays a form requiring the new email address. 5. The user enters the new email address. 6. The system checks if the new email address is already in use. 7. The system generates a unique verification code. 8. The system sends an email to the new email address containing the verification code. 9. The system displays a form requiring the verification code. 10. The user enters the verification code received in the email. 11. The system verifies the code:     1. Checks if the code matches the generated code.     2. Checks if the code has expired. 12. If the code is valid, the system updates the user's email address in the database. 13. The system displays a success message confirming the email change | | |
| Alternative Flows: | 12.1 Email Already in Use: If the new email address is already registered, an error message is displayed.  12.2 Invalid Verification Code: If the user enters an incorrect or expired verification code, an error message is displayed, and the user is given the option to resend the code. | | |
| Exceptions: | **7.1 Email Delivery Failure:** If the verification email cannot be delivered, an error message is displayed, and the user is given options to resend the email or try a different address | | |
| Priority: | Medium | | |
| Frequency of Use: | Low | | |
| Business Rules: | **BR-01**: Email uniqueness should be enforced. Verification codes should have an expiration time. | | |
| Other Information: | none | | |
| Assumptions: | 1. Email uniqueness should be enforced. Verification codes should have an expiration time. | | |

A blue square with a black oval

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-05 Search for Housing** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 27/09/2024 |
| Primary Actor: | Registered User (Tenant, Host) | Secondary Actors: |  |
| Description: | This use case describes how a registered user searches for housing based on various criteria. The user can input search parameters such as location, price range, number of bedrooms, and other amenities. The system then displays a list of matching housing options. | | |
| Trigger: | The user initiates a search from the dedicated search page. | | |
| Preconditions: | PRE-1. The user is logged into the system (optional, searching can be done anonymously). | | |
| Postconditions: | POST-1. A list of housing options matching the search criteria is displayed | | |
| Normal Flow: | 1. **Search for Housing**   1. The user navigates to the search page or uses the main search bar.  2. The user enters search criteria (e.g., location, price range, number of bedrooms, amenities).  3. The system validates the input data.  4. The system queries the database for matching housing options.  5. The system displays the search results.  6. The user can refine the search results by applying filters or sorting. | | |
| Alternative Flows: | **6.1 No Results Found**: If no matching housing options are found, the system displays a message indicating this and suggests broadening the search criteria.  **6.2 Invalid Search Input**: If the user enters invalid search data (e.g., invalid date format, incorrect price range), the system displays an error message and prompts the user to correct the input. | | |
| Exceptions: | **4.1 Database Error:**  If there is an error during the database query, an error message is displayed, and the user is advised to try again later. | | |
| Priority: | High | | |
| Frequency of Use: | High | | |
| Business Rules: | **BR-01**: Search criteria should be validated before querying the database.  **BR-02**: Search results should be displayed in a clear and concise manner. | | |
| Other Information: | The search functionality should be optimized for performance to handle a large number of listings. Consider implementing autocomplete and suggested search features. | | |
| Assumptions: | 1. The system has a database of housing listings with relevant information. 2. Users can search using a variety of criteria | | |

A blue square with a black circle

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-06 Send message** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 30/10/2024 |
| Primary Actor: | Registered User (Tenant, Host) | Secondary Actors: |  |
| Description: | This use case describes how a registered user sends a text message in a chat session. The system allows real-time message exchange using WebSocket. | | |
| Trigger: | The user types a message and clicks "Send." | | |
| Preconditions: | PRE-1: The user is logged in.  PRE-2. A chat session exists with the recipient. | | |
| Postconditions: | POST-1: The message is delivered in real-time to the recipient. | | |
| Normal Flow: | 1. **Send message**   1. The user opens a chat window with a specific user.  2. The user types a message( may be file, image or video) and clicks the "Send" button.  3. The system sends the message via WebSocket.  4. The recipient receives the message instantly in the chat window. | | |
| Alternative Flows: | **4.1** If the recipient is offline, the message is stored and marked as "unread." | | |
| Exceptions: | **1.1** Connection Error: If there is a WebSocket connection error, the system displays an error message, and the message remains in the input field. | | |
| Priority: | High | | |
| Frequency of Use: | High | | |
| Business Rules: |  | | |
| Other Information: | None | | |
| Assumptions: | 1.The user has a stable internet connection for WebSocket communication.  2.The recipient is registered on the platform and has an active account. | | |

A blue square with a black oval

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-07 View Chat History** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 30/10/2024 |
| Primary Actor: | Registered User (Tenant, Host) | Secondary Actors: |  |
| Description: | This use case describes how a user views their chat history with another user. | | |
| Trigger: | The user opens a chat with a specific person. | | |
| Preconditions: | PRE-1: The user is logged in. | | |
| Postconditions: | POST-1: Chat history is displayed, showing recent messages with the other user. | | |
| Normal Flow: | 1. **View chat history**   1. The user selects a chat conversation to view.  2. The system retrieves previous messages from the database.  3. The system displays the chat history in reverse chronological order | | |
| Alternative Flows: | **2.1** No History Available: If there are no previous messages, a message stating "No previous messages" is displayed | | |
| Exceptions: | **2.1** Database Error: If there is an error retrieving chat history, an error message is displayed. | | |
| Priority: | High | | |
| Frequency of Use: | High | | |
| Business Rules: |  | | |
| Other Information: | Chat history should load quickly to improve user experience. | | |
| Assumptions: | 1. Chat history is stored securely and can be retrieved without delays. 2. The user is familiar with navigating through the chat history interface. | | |

A blue rectangular object with black text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-08 Block/Unblock User** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 30/10/2024 |
| Primary Actor: | Registered User (Tenant, Host) | Secondary Actors: |  |
| Description: | This use case describes how a user can block or unblock another user. | | |
| Trigger: | The user selects "Block" or "Unblock" in the chat options. | | |
| Preconditions: | PRE-1: The user is logged in. | | |
| Postconditions: | POST-1: Messages cannot be sent or received between blocked users. | | |
| Normal Flow: | 1. **Block/ Unblock User**   1. The user selects "Block" from the chat options.  2. The system updates the block status in the database.  3. The user receives a confirmation message that the user has been blocked | | |
| Alternative Flows: | **3.1** Unblock: If the user selects "Unblock," the system removes the block status, and a confirmation is shown. | | |
| Exceptions: | **2.1** Database Error: If there is an error updating the block status, an error message is displayed. | | |
| Priority: | Medium | | |
| Frequency of Use: | Low | | |
| Business Rules: |  | | |
| Other Information: | None | | |
| Assumptions: | 1. The user understands the implications of blocking and unblocking other users.  2. Blocked users will not receive notifications about the block. | | |

A diagram of a system

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| ID and Name: | **UC-09 View and Manage Notifications** | | |
| Created By: | Nguyen Cong Manh | Date Created: | 30/10/2024 |
| Primary Actor: | Registered User (Tenant, Host) | Secondary Actors: |  |
| Description: | This use case describes how a registered user views, deletes, reads, and marks notifications as read in the system. Users can see a list of their notifications and perform management actions on each notification. | | |
| Trigger: | The user initiates this use case by navigating to the notifications page or selecting the notifications icon. | | |
| Preconditions: | PRE-1: The user is logged in.  PRE-2. The system has existing notifications for the user. | | |
| Postconditions: | POST-1. The notifications page displays the user's list of notifications.  POST-2. If the user marks notifications as read, their status updates accordingly.  POST-3. If the user deletes notifications, they are removed from the user's notification list. | | |
| Normal Flow: | 1. **View and Manage Notifications**   1. The user navigates to the notifications page or clicks the notifications icon.  2. The system retrieves and displays a list of notifications, sorted by date (most recent first).  3. The user can read each notification by selecting it, which opens a detailed view if available.  4. If the user reads a new notification, the system updates its status to "read."  5. The user can mark a notification as "read" without opening it by selecting a "Mark as Read" option.  6. The user can delete a notification by selecting the "Delete" option. | | |
| Alternative Flows: | **6.1 No Notifications Available:**   * If the user has no notifications, the system displays a message indicating there are no notifications to view.   **6.2 Network Error During Load:**   * If there is a network error or the notifications cannot load, the system displays an error message and suggests trying again. | | |
| Exceptions: | **5.1 Error Marking Notification as Read:**   * If there is an error while marking a notification as read, the system displays an error message and retains the notification's original status.   **6.3 Error Deleting Notification:**   * If there is an error during deletion, the system displays an error message and keeps the notification in the list**.** | | |
| Priority: | Medium | | |
| Frequency of Use: | Low | | |
| Business Rules: | **BR-01:** Only registered users can view, read, mark, and delete notifications.  **BR-02:** Notifications marked as "read" should not appear in the "new" notifications count. | | |
| Other Information: | The notifications feature should be responsive and provide real-time updates where possible. | | |
| Assumptions: | 1. Notifications are stored in a format that allows retrieval and modification (e.g., updating read status).  2. The user has the required permissions to mark or delete their notifications.  3. The system provides feedback if there are errors in retrieving or managing notifications. | | |