

**LEARNING HUB**

**Software Requirement Specification**

– Hanoi, May 2023 –

**Table of Contents**

[I. Overview 3](#_heading=h.gjdgxs)

[1. Introduction 3](#_heading=h.30j0zll)

[2. System Functions 4](#_heading=h.1fob9te)

[a. Screen Flow 4](#_heading=h.plfrjvewxp2y)

[b. Screen Details 4](#_heading=h.tr5zzxsuklco)

[c. User Authorization 6](#_heading=h.ag5v92oc5zc3)

[d. Non-Screen Functions 8](#_heading=h.itp5ukoo4hvc)

[3. Common Requirements 1](#_heading=h.3znysh7)

[II. Functional Requirements 1](#_heading=h.2et92p0)

[1. User authentication and authorization 1](#_heading=h.qrqmmmdr0kww)

[2. Feature management 1](#_heading=h.xb3n7l1r24bz)

[3. User Registration 1](#_heading=h.mo7n0o84zjei)

[4. User Management 1](#_heading=h.6g6o4qj2pegj)

[5. Flashcard 1](#_heading=h.fnpu6nr97scc)

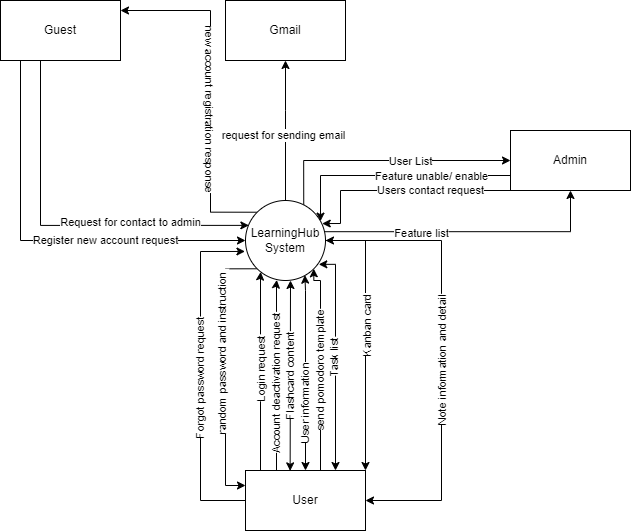
[6. Contact 1](#_heading=h.oj41251l75k9)

[7. Task management 1](#_heading=h.2m94fo4o63b7)

# I. Overview

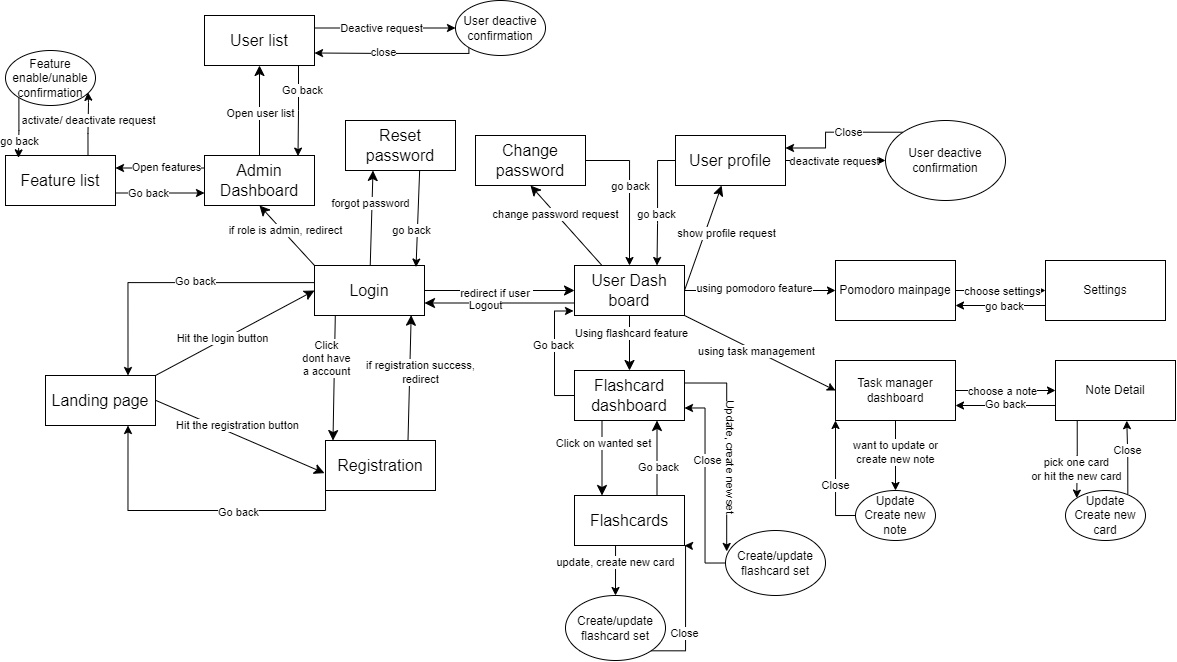
## 1. Introduction

Introducing LearningHub, the ultimate productivity app that combines flashcards, task management in one platform. We aim to help you streamline your productivity by reducing the number of apps you need to manage. With LearningHub, you can easily organize your study routine, stay on top of tasks, and boost your focus, all in one place. Say goodbye to juggling multiple apps and hello to maximum productivity with LearningHub.



## 2. System Functions

### a. Screen Flow



### b. Screen Details

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Feature** | **Screen** | **Description** |
| 1 | Login | User Login | Screen will have a form with two fields username and password. if the username and password are valid, the user will be sent a redirect to the user dashboard |
| 2 | Sign up | Registration | Screen will have a form with multiple fields  (email, password, first name, last name, phone number). when a user hit to create an account button. the system will check all the fields are valid (not null, valid template) and then save the new user to the database |
| 3 | Contact Us | Landing page | This page will briefly introduce the app. What the app can do to help users. It will also have a contact us session when everyone wants to contact the team. Users can leave emails and messages and send them to the team. |
| 4 | Reset Password | Forgot Password | In case users forgot their password, users will access this page and input their username (email) and hit the button to send a request to reset their password to email |
| 5 | Change Password | Change Password | User must enter a new password and confirm new password to change current password |
| 6 | Show user profile | User Profile | List all personal information of a current user. there will be a button for the user if s/he wants to change her profile. However, s/he can not change his or her email. user can also change the active status of that account (after a confirmation pop-up, if the account is set to deactivate, users will be sent back to a login page) |
| 7 | Show current work and redirect them to wanted feature | User dashboard | Navigate bar that has links to core features. The user dashboard will also show the list of notes and list of flashcards set in a specified order |
| 8 | Error handle | Error page | In case of the application caught an unexpected error (resources not found, external server error,..). The application must send the user to this error page with no trouble. |
| 9 | Feature activation | Feature list | This screen will list all core features and services this screen, the admin can see its description and also have a button in order to activate or deactivate one or more features. Users can only use activated features. This feature is used for admin in case of maintenance or no longer support. |
| 10 | User management | User list | This screen will list all accounts registered into this application and provide some information (phone number, email, user name). There will also include a button that allows the admin to set active or deactivated users (deactivate users will no longer be access and use the app). |
| 11 | Show all flashcard set | Flashcard dashboard | This screen will be shown first when the user wants to use this flashcard feature. On this screen, a list of flashcard sets will be shown in the last accessed order. User can click on one of those set to redirect to the flashcard list reference to that set. |
| 12 | Create, update flashcard set | Flashcard set create/update | A pop-up will appear when the user hit the modify button at the card set or hits the create a new card set in the flashcard dashboard. If the user hit the already had set, the information of the set will appear (title, description) and users can update it. If the user hit the create a new flash card set. a clear form (with no pre-value) will be shown and the user can fill out those fields to create a new set. |
| 13 | Show all flashcards | Flashcards List | When the user chooses a set that the user wants. this screen will be redirected. This screen will be divided into 2 main parts: whole cards and card lists. in the whole cards part, it will be rendered like a real flashcard (have 2 sides for definitions and terms) and the animation will mimic as likely as the real card does |
| 14 | Create, update flashcard set | Flashcard create/update | A pop-up will appear when the user hit the modify button at the card or hit the create a new card in the flashcard list. If the user hit the already had card, the information of the set will appear (terms, definition) and users can update it. If the user hit the create a new flashcard. a clear form (with no pre-value) will be shown and the user can fill out those fields to create a new card and add it to that set. |

### c. User Authorization

### 

| **Screen** | **Guest** | **User** | **Admin** |
| --- | --- | --- | --- |
| Landing Page | X | X | X |
| Login | X | X | X |
| Registration | X | X | X |
| Change password |  | X | X |
| Forgot Password | X | X | X |
| User profile |  | X | X |
| User deactivation confirm |  | X | X |
| Flashcard dashboard |  | X | X |
| Create new set |  | X | X |
| Update set |  | X | X |
| Delete set |  | X | X |
| Flashcard list |  | X | X |
| Card list view |  | X | X |
| Whole card view |  | X | X |
| Create new card |  | X | X |
| Update card |  | X | X |
| Delete card |  | X | X |
| Task management |  | X | X |
| Task management dash board |  | X | X |
| Create new Note |  | X | X |
| Update Note |  | X | X |
| Delete Note |  | X | X |
| Note screen |  | X | X |
| Update note information |  | X | X |
| Add new column to Kanban board |  | X | X |
| archive column to Kanban board |  | X | X |
| Add card to each column |  | X | X |
| Update card information |  | X | X |
| Delete card |  | X | X |
| Count card on each column |  | X | X |
| Move card to each column |  | X | X |
| Add label to card |  | X | X |
| Create new label |  | X | X |
| Update label information |  | X | X |
| Delete label |  | X | X |

### 

In which:

* Guest
* User
* Admin

### d. Non-Screen Functions

[Provide the descriptions for the non-screen system functions, i.e batch/cron job, service, API, etc.]

| **#** | **Feature** | **System Function** | **Description** |
| --- | --- | --- | --- |
| 1 | User validation and authorization | reset the password | if the user forgets the password(by clicking the forgot password option). The password of that user will be set to a server-generated password. A new password will be sent to users via email so they can access the account and change their key. |
| 2 | User manager | deactivate | user account who is deactivated will be unable to access the account nor use the app’s feature. Two ways will cause the user to be deactive(self-deactive, admin force to deactivate) |
| 3 | Security | Account creation | System may require activation to create accounts to access applications that store information and profile. A security system typically grants access to accounts when users enter the correct username and password |
| 4 | Alaram | Pomodoro timer | When the timer is set to 0, an alarm sound will be actuated. |
| 5 | Deadline check | Taskmanager | the start and due date of each card will be recorded and be represented as a line in the timeline chart |
| 6 | Input valid | login/ registration | check if all the required fields are filled and some fields (ex: phone number, email) are in the right form |
| 7 | Task management | move card | card can only be moved from column to another. this movement must be smoothly with minimized lag. The position of the card inside each column must be saved. |
| 8 | Task management | archive column | user must be pass the confirmation test in order to use this action. all card inside a column will also be deleted. |
| 9 | Task management | board data update and render | Each time when user interacted with board in front end, the data will be sended to backend to update in the database |
| 10 |  |  |  |

## 

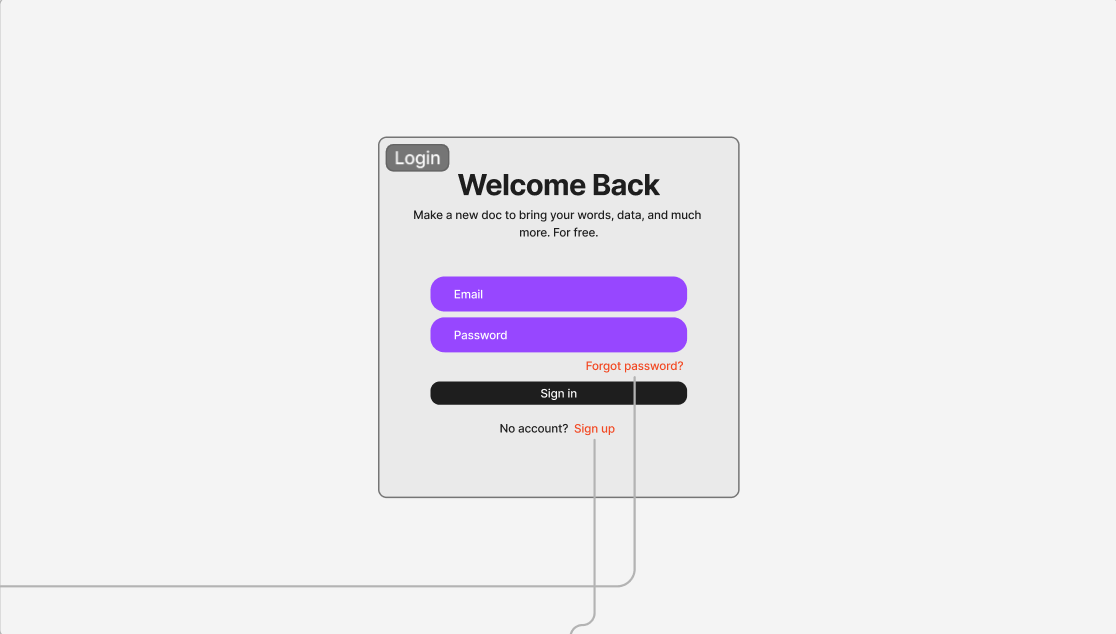
## 3. Common Requirements

1. The web page must be rendered smoothly without (less) delay in the ideal environment.
2. The web page must be displayed in English. However, the card or task can be used in any language using a UTF-8 encoder
3. The server side and client side must communicate with each other by using API (RESTful web application)
4. User data still remain in the database after deleting permanently
5. Every input field must be valid in type (email, phone number, password,...)
6. The application must be able to use in a variety of browsers
7. User-friendly design
8. User email must be valid and real in order to use the forget password feature
9. Only users and admins can use all the functionalities in this application.
10. Error handling mechanisms allow developers to identify and troubleshoot errors, improving the stability and improve user experience.

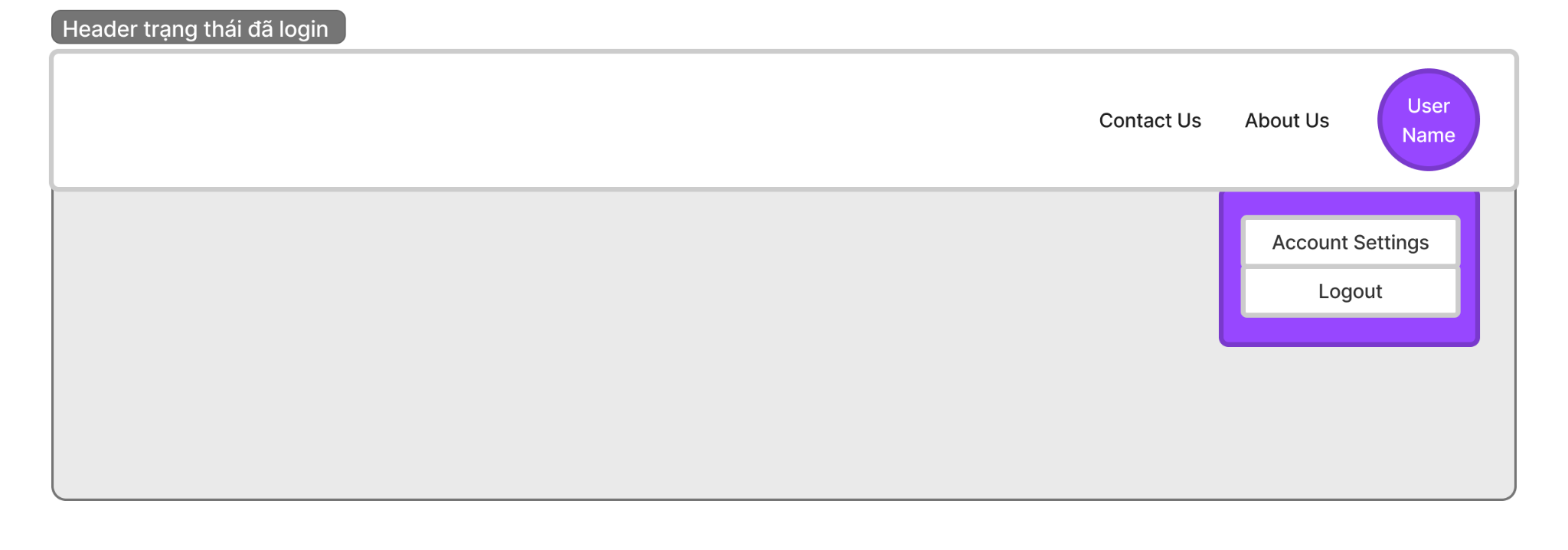
# II. Functional Requirements

## User authentication and authorization

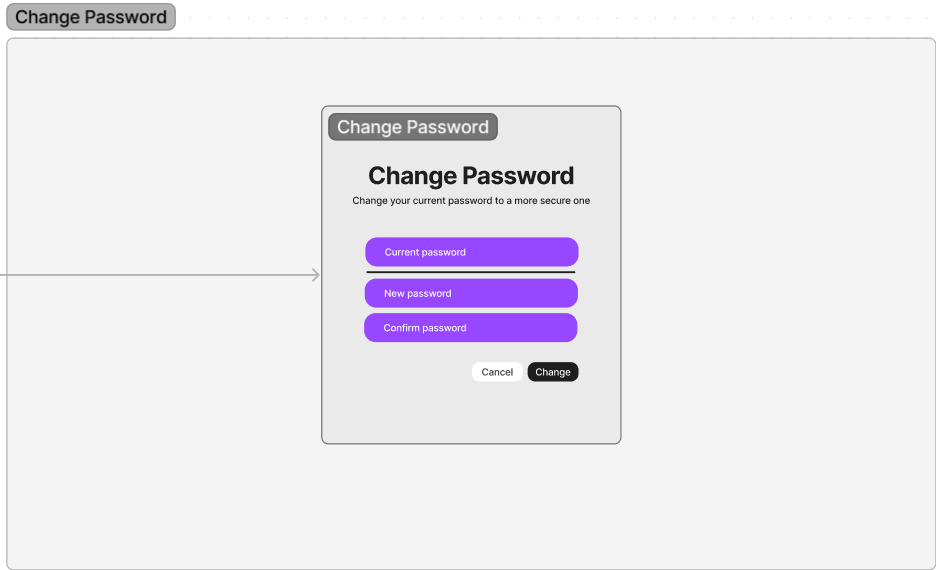
* 1. **Login**
* Function trigger: When the user hit the login button on the landing page or logout of the user dashboard. This page will also be redirected if the user completed the step of forgetting the password.
* Function description: Users and admins have to log in to use the full functionalities of the application.
* Screen layout:



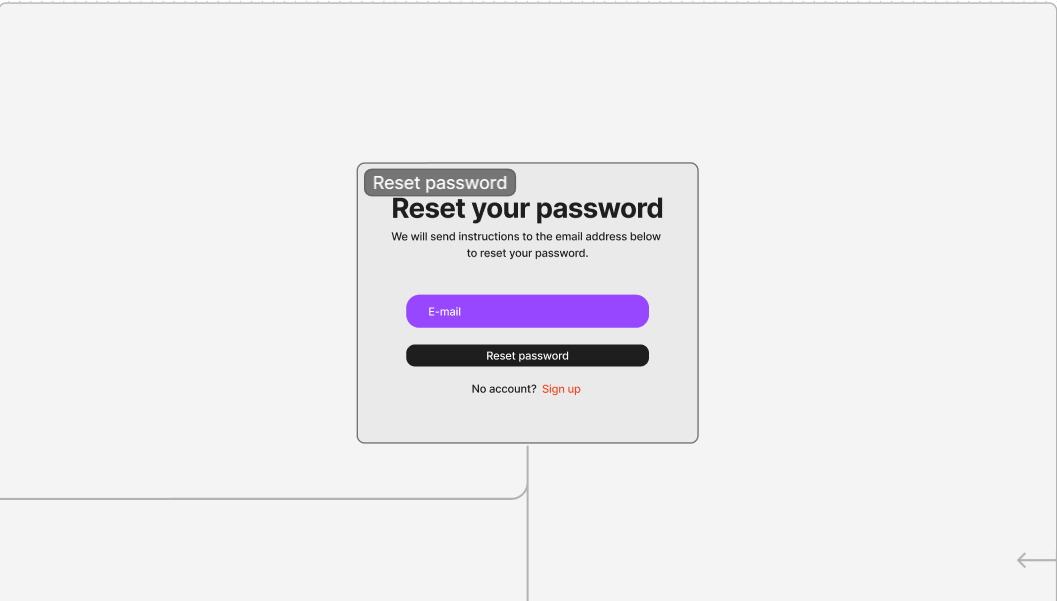
* Function Details: Anyone who wants to use the function of the application must have an account and login it using this login screen. The user must provide an email and password to the application. The server will check whether there is any account using this email address and whether the password is correct. If the email and password user entered correctly, the user will be redirected to the user dashboard (for the user) or admin dashboard (for the admin).
  1. **Logout**
* Function trigger: user hits the Logout button in the drop-down that displays after clicking the avatar
* Function description: the user will be logged out from the system
* Screen layout:



* Function Details: when the user wants to change the account or the account session is expired. Users then can’t access the system until they log in again
  1. **Change password**
* Function trigger: request changing password was sent.
* Function description: when the user (admin) wants to change the password. They can choose the change password option on the dashboard. The user must fill in 3 fields to use this function (current password, new password, and confirm new password)
* Screen layout:



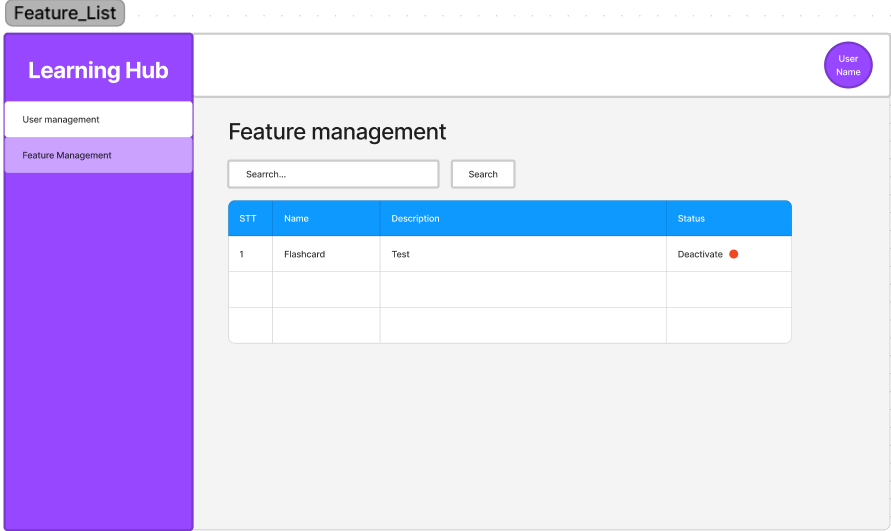
* Function Details: in case the user/admin want to change their password for security reason. The server must check the current password to see that the user knows what to do and they wish to change their password. After another confirmation by comparing the new password and the confirmation password. the server will change the password information of that account to a new password that they want.
  1. **Forgot password**
* Function trigger: when a user is unable to remember their password and requests the server in resetting it. Users can use this function through a “Forgot password” on the login screen.
* Function description: User and admin who have forgotten their password and needs to reset it. User will send their email address to the server to receive a new random password via email. The user will that password to login to the account.
* Screen layout:



* Function Details: the system will check if the provided email address exists in the user database. If a matching record is found, the server will replace that account’s password with a self-gen random password. That password will be sent to that email address. The User will be redirected to the login page to continue with the login process. In case the email address is invalid, the server will send a message to the page. If mail delivery Failure, the user will be redirected to an error page with code (500)
  1. **Authorization**
* Function trigger: This function will always be triggered when the user wants to access the application’s features. Some features of the application can only be accessed by a specified role only.
* Function description: Based on the current user’s role. the application will allow users to use some specific functions. The User will be able to see the user profile and use the Pomodoro feature, flashcard feature, and task management feature.
* Function Details: The server will check the role of the user logged in and based on that role will allow the user to use specific functions. In case the user still trying to use functions that the user does not have permission will be redirected to the Error page (unauthorization)

## Feature management

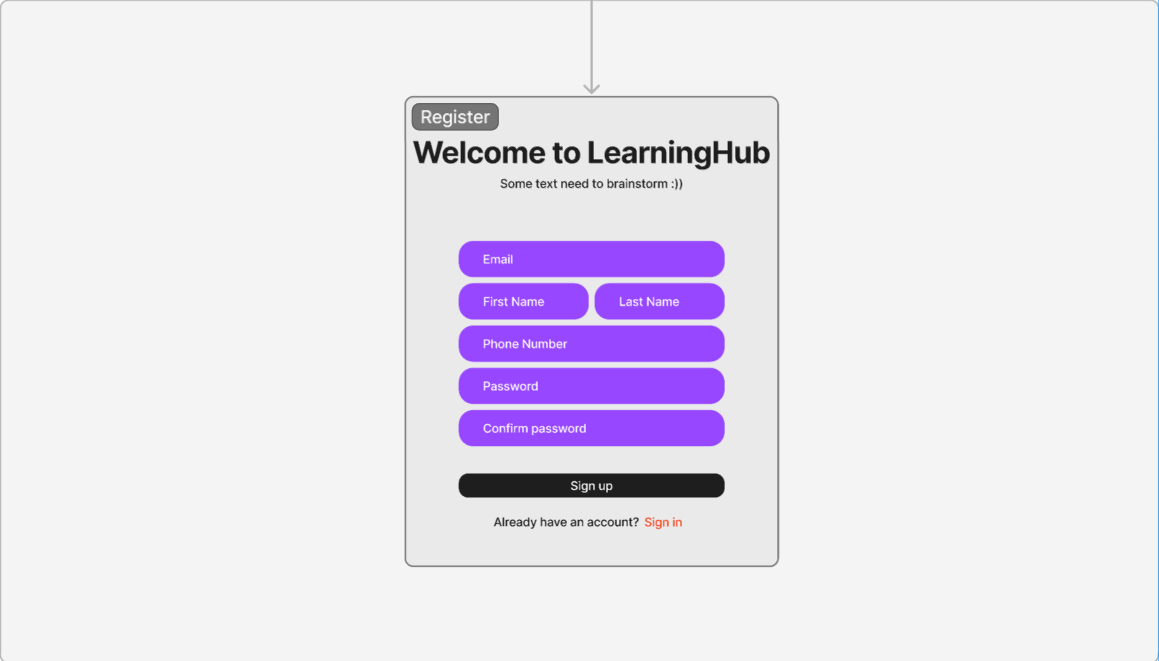
* 1. **Feature List**
* Function trigger: when an administrator wants to see the list of features that the application supported and set the active status of that function in case of maintenance or no longer support.
* Function description: Admin will be able to see a list of features (by clicking the feature list button). this list will contain the information of the feature which is name, description, and status. Admin can set the action of each feature and set the note of the reason why the feature is deactivated (this note will be shown to the user if the user still tries to access the feature.
* Screen layout:



* Function Details: The user can only use the feature if that feature set the status to active. In case the feature is deactivated by the admin when a user tries to access the features. They will be redirected to the Error page which will show the message written by the admin explaining why the feature is closed.

## User Registration

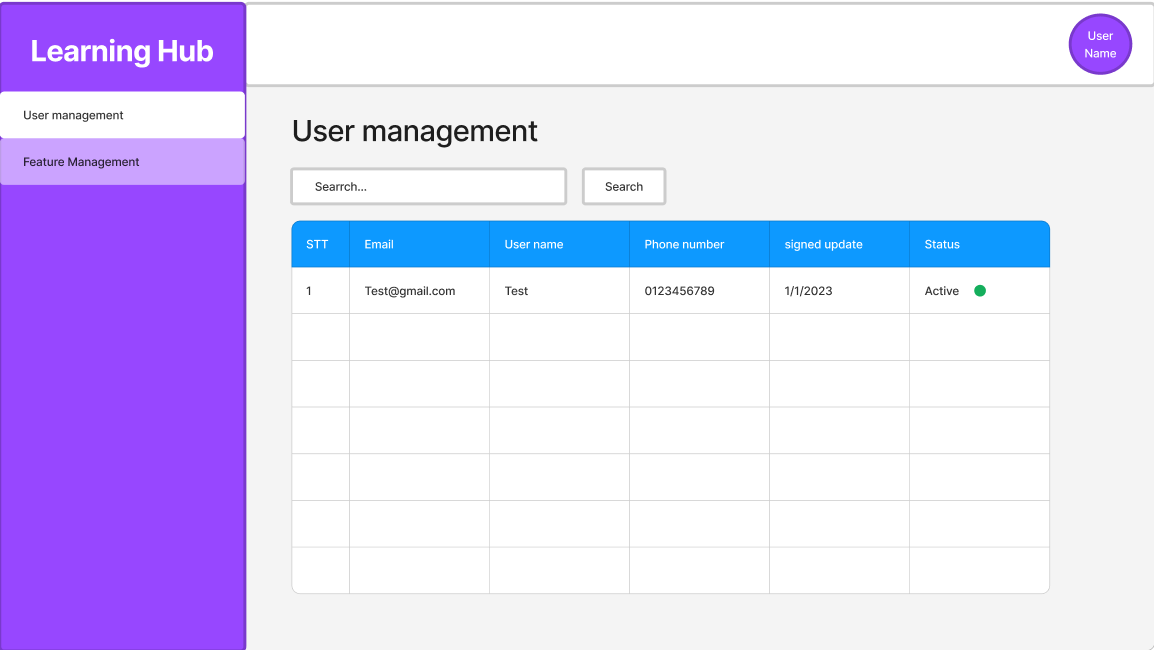
* Function trigger: new user intends to create an account on the system. This can be initiated through a Signup button on the landing page or login screen.
* Function description: User who wants to create an account on the system. The purpose of this function is to allow users to create a new account and gain access to the system’s features and services. This registration screen will involve a registration form and submit button. After the input validation (required fields are filled correctly) the system will create new user information in the database.
* Screen layout:



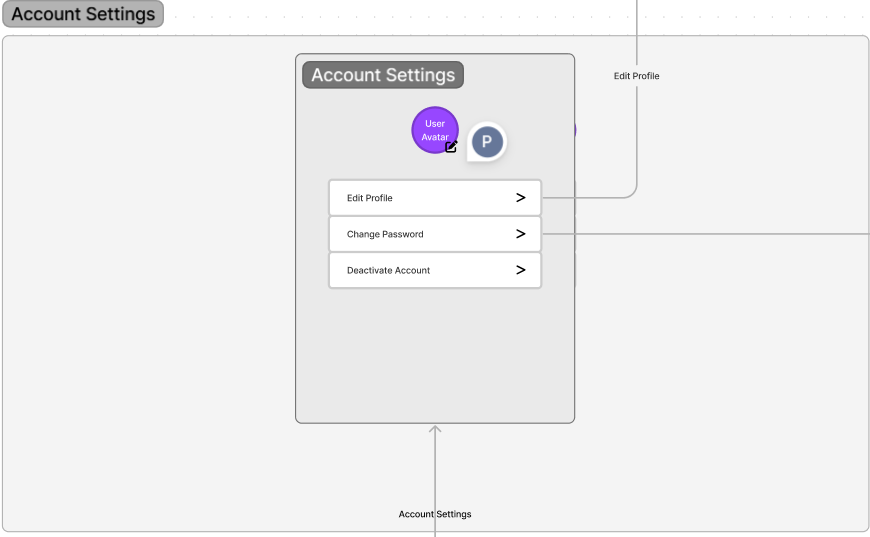
* Function Details: The system will check all required fields in the registration form are filled. Validation is performed on each field to ensure correct formatting (email address, phone number). If any input fields are invalid, appropriate error messages are displayed, indicating the specific fields that need to be corrected. If the input validation is successful, the system will create a new user account, storing new user information in the system database, and the role will be set default to the user. In case of an unexpected error, the server will redirect the user to an error page.

## User Management

* 1. **User list**
* Function trigger: When the admin want to see the list of user and he/she wants to see some information about users and set deactivate user
* Function description: admin can search users by email by the search filter. in the user list, the admin can deactivate the user account by clicking the status box
* Screen layout:



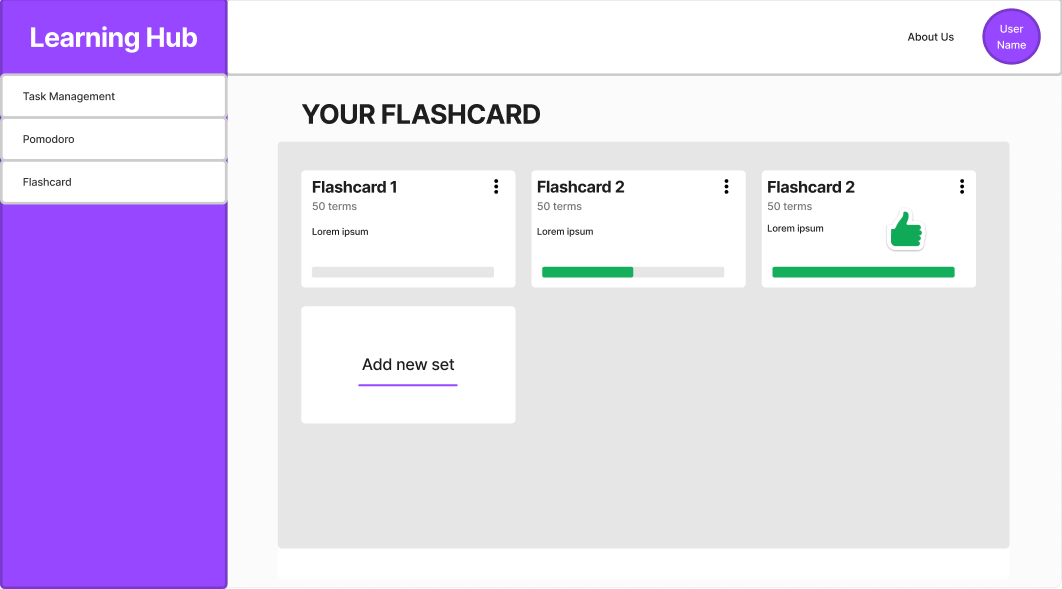
* Function Details: The function will provide the admin user list and give the admin permission to deactivate the user.
  1. **Deactivation**
* Function trigger: when the user wants to deactivate his/her account. This can be initiated through an account deactivated option in user settings. Admin can also be able to use this function to deactivate the user account through the user deactivate option in the user list screen.
* Function Description: User who wants to deactivate his/her account and no longer wants to use this account again. Admin can use this function to deactivate user account in case of user violate the regulation,...
* Screen layout:



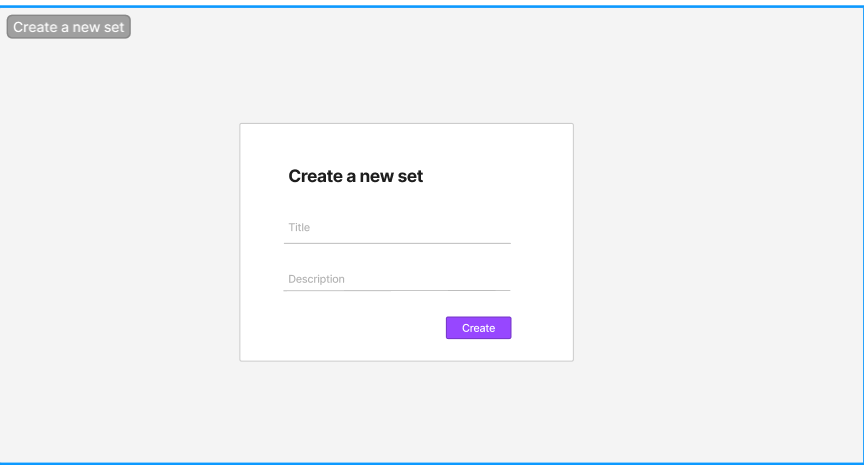
* Function Details: The user/admin when clicking the deactivate account button. after confirming in a popup, this user account will be found in the user database and set the status to deactivate. The user will then be redirected to the login page (in the user role).

## Flashcard

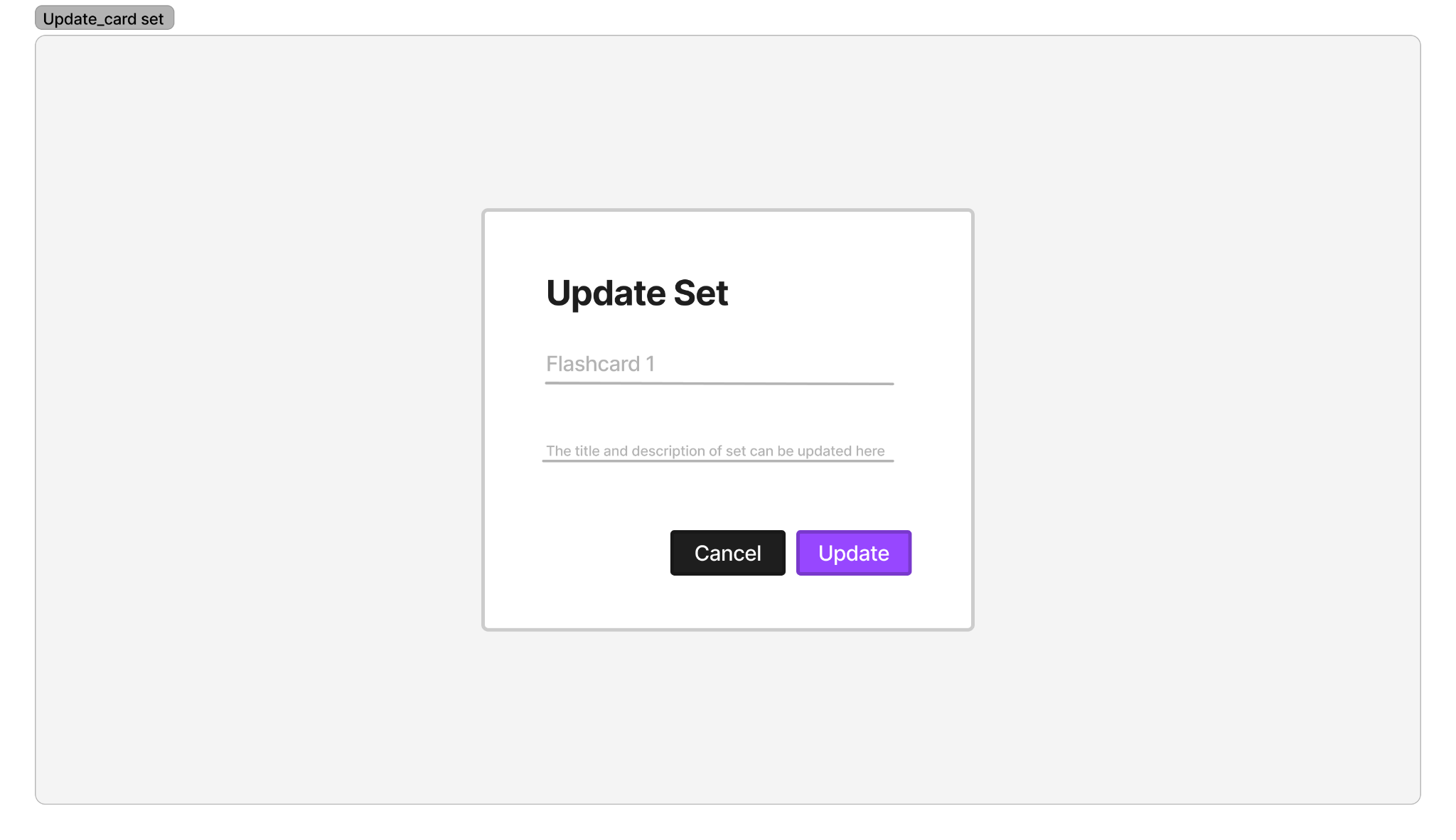
* 1. **Flashcard dashboard**
* Function trigger: when a user accesses the flashcard feature within the application. This can be initiated through a navigation path, such as clicking on a “Flashcards” tab or selecting a flashcard option from the side menu.
* Function description: The user who interacts with the flashcard dashboard to study and manage flashcards. This screen also shows a list of sets that the user created. This screen will also have a section for the user to create a new card set ( a pop-up will appear). each set will be represented as a block containing a Title, description, number of cards, and a progress bar. When the progress bar reaches 100%, a green tick will appear. The system processes user input to create a new set by filling in some information or updating, or deleting the set.



* Function Details: This screen will also have a section for the user to create a new card set ( a pop-up will appear). each set will be represented as a block containing a Title, and description. this screen will not only show the list of sets but also give the user permission to delete, add, and update set
  1. **Add new set**
* Function trigger: This function will be triggered when the user hit the add new set button in the flashcard dashboard.
* Function description: The user after clicking add new set button will see a pop-up, here, the user provided the set title and set description to create a new set. the system will create a new set when receiving user fields
* Screen layout:

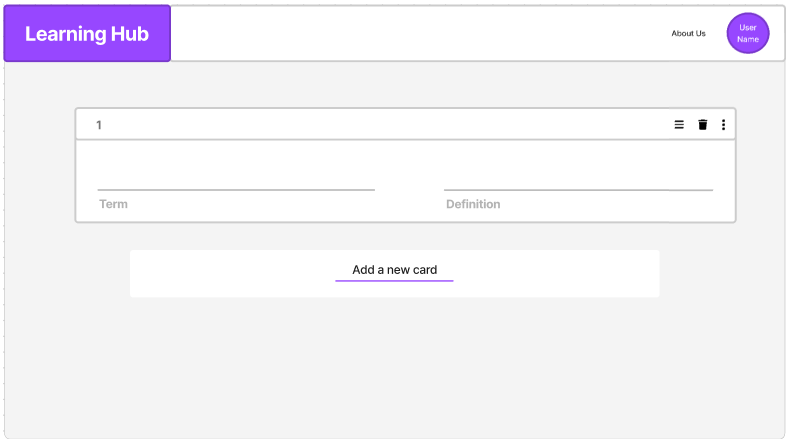


* Function Details: This is a pop-up involving title and description, the system will check all those fields if they are not empty. if validation was successful, the system will create a new set of information in the database. if not, the system will redirect the user to the flashcard dashboard with nothing change.
  1. **Update Set**
* Function trigger: This function will be triggered when the user pressed the button Edit set in the drop-down list
* Function description: update set title and description
* Screen layout:

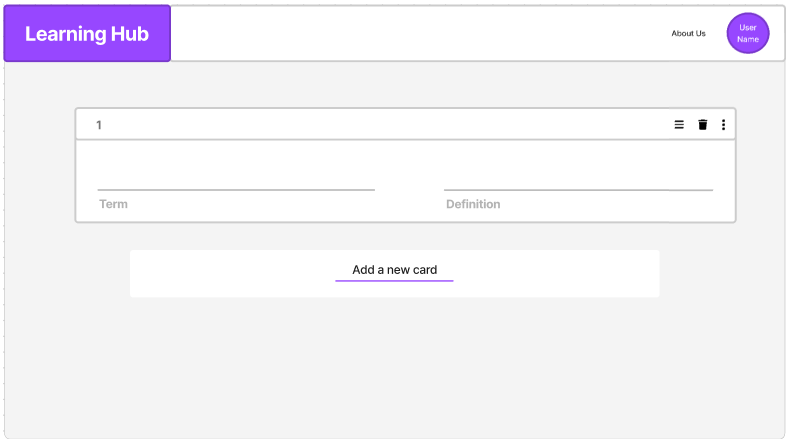
****

* Function Details: This is a pop-up involving title and description, the system will check all those fields if they are not empty. if validation was successful, the system will create a new set of information in the database. if not, the system will redirect the user to the flashcard dashboard with nothing change.
  1. **Delete flashcard set**
* Function trigger: This function will be triggered when the user pressed the button Delete set in the drop-down list
* Function description: delete set from your workspace
* Function Details: when the user press at this button, there will be a pop-up where user confirm that they really want to delete this set or they can cancel the action
  1. **Flashcard screen**
* Function trigger: when the user selects a specific flashcard to view or access the flashcard list within the Flashcard Dashboard. This function is typically initiated through user interaction by clicking one set in the dashboard
* Function description: user who interacts with the flashcard screen to view flashcards and manage their collection. The Flashcard screen function provides users with a comprehensive view of individual flashcards and a list of their flashcard collections for studying and management. The screen will be divided into 2 main parts: The card view and the card list. The card view is a dedicated part that displays the content of an individual card, including the term and definition side. The card list view is a section that presents an organized list or grid of flashcards, allowing the user to browse and manage their collection.





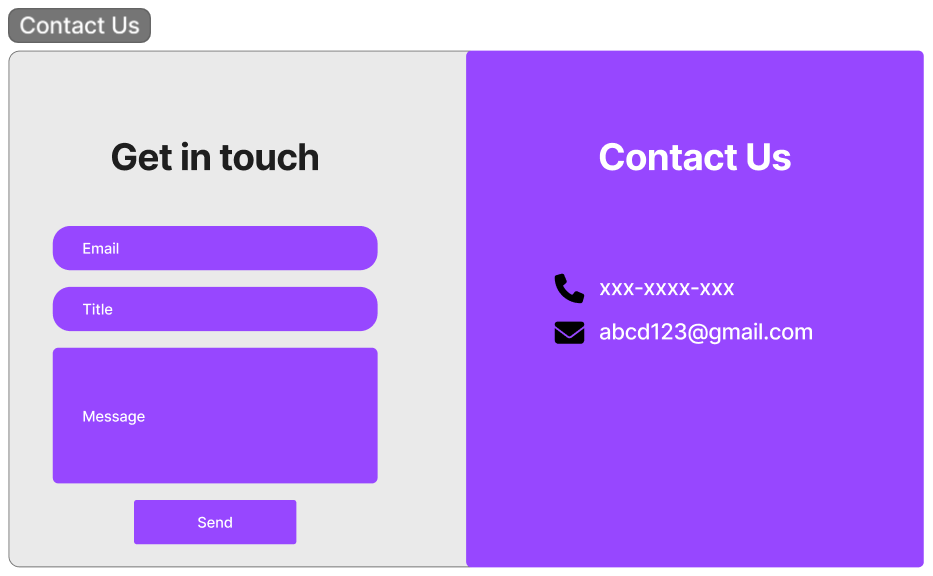
* Function Details: For the card view interface, by default, the first card will be shown first. There will have 2 buttons for the user to navigate the card. The front side of the flash card is initially shown, and the user can flip the card to reveal the backside. There will be a progress bar that will check every card is retrieved. the bar will reach full if the user retrieves all cards in this view. For the card list interface. this interface will be presented an organized view of the user’s flashcard collection, allowing the user to browse and manage their card. the flashcard will be displayed as a list of card items. Each item will show all 2 fields of the card (term and definition). controls are provided for actions such as editing, deleting, and creating a new card. In case the user tries to edit a card that doesn’t exists (via link), the system will redirect the user back to their flashcard view with nothing changed. in case the flashcard set is empty. the flashcard list will at first show a black card in card view and only add a new card button in a card list. in case the server can not reach the set or card, the system will redirect the user to an error page (resources not found)
  1. **Create a new Flashcard, delete and update the flashcard**
* Function trigger: can be triggered when a user wants to create a new flashcard or modify an existing one. Users can add a new card by clicking add new card button or change the information on the card by changing directly to the card list. Users can also delete cards with just one click
* Function description: User who interacts with the flashcard feature to create and update flashcards. The purpose of create and update flashcard function is to enable users to add new flashcards to their collection and make changes to existing flashcards. In each card item, there will have a button for the user to delete that card.
* Screen layout:



* Function Details: For flashcard creation. Users can access a flashcard creation form by clicking on add a new card button. The blank card item will be generated at the end of the card list (including term, definition). Users can fill out those fields to create a new card. For flashcard updating: the user can select existing flashcards from the flashcard dashboard for editing. the system will save new content into the database. For flashcard deletion, the user will click a button on each card item, the system will know what card the user wants to delete and remove from the database

## Contact

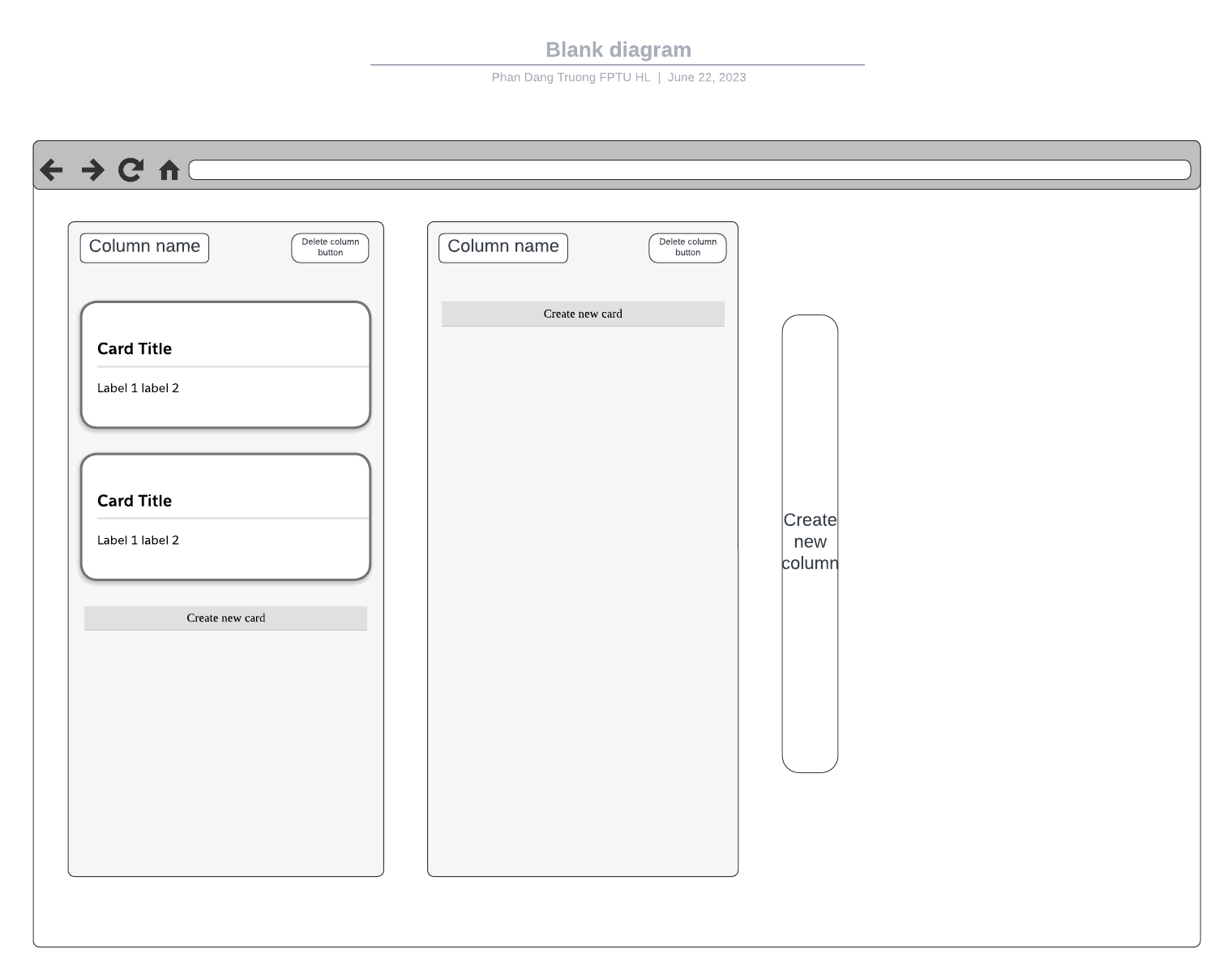
* 1. **Contact us**
* Function trigger: when they need to communicate with the system’s support or customer service team. This function is accessible through a contact us section on the landing page.
* Function description: The purpose of this function is to provide users with a means to reach out to the system’s team for assistance, inquiries, and feedback. The contact form includes email, title, and detail and sends button



* Function Details: The system will check that all fields are filled. Validation is performed on each field to ensure the correct formatting (email field). This information is used to send email to the team via Gmail.

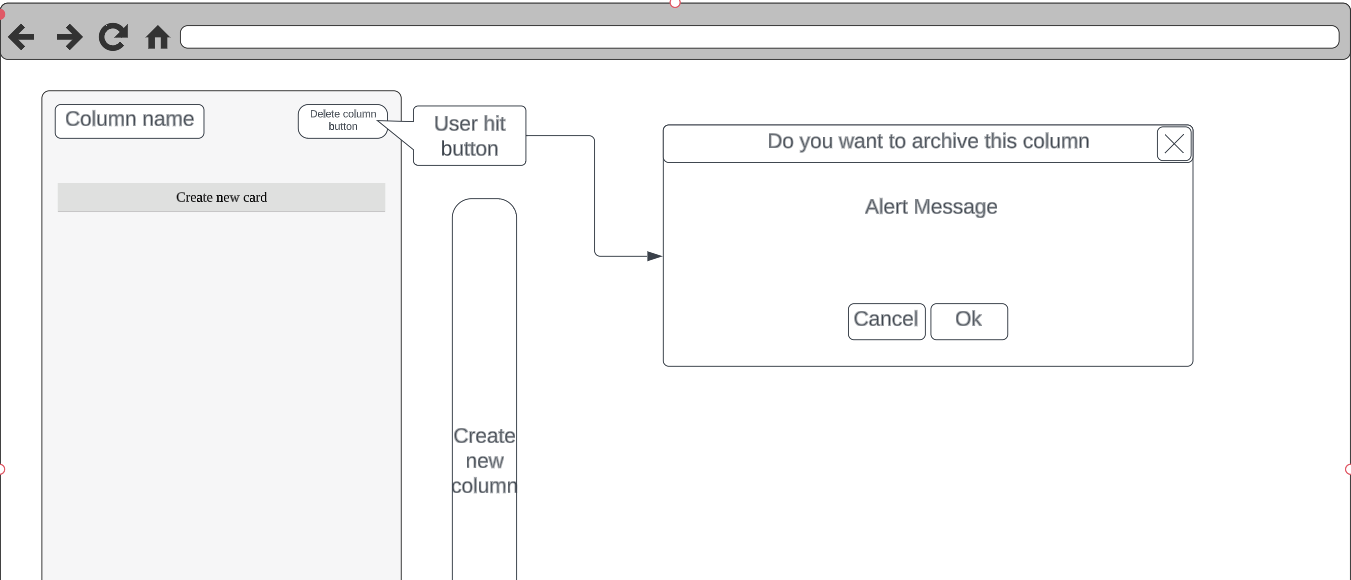
## Task management

1. **Board data update and render**

* **Function trigger**: When users go to detail one of their notes. This board will appear in the note details screen. when user reload, access to detail note page or interact with the kanban board (this includes: moving card, add column, archive column, delete card, update card information)
* **Function description**: This function will make the kanban board on the note screen stay up-to-date by user interaction. when the user comes back or reloads the page, all the modified information on the Kanban board will be saved and rendered exactly when they left.
* **Screen layout**
* **Function detail**: when the user access to detail note page (we call it the note screen). this function will fetch board data from the backend based on the board id given by the front-end side request API. The backend will return to the front-end side JSON data which has a unique data structure (column id: column title, card list(card name, label list: label name and color)). based on data obtained from the server, the front-end side will be rendered a kanban board with detail of columns, and cards in a board-liked format with some function buttons (add new card button, add new column button, archive column button). this kanban board rendered for the user will allow the user to move the card smoothly between columns or reorder the position of cards inside a column. when the user interacted with the Kanban board by some of these actions (add a new card, delete card, add a new column, delete a column, update a label, add/remove a label from one card, add a new label, moving the card to a different destination). the front-end side will update that JSON data(given by the server-side before)based on user actions and send it back to the backend to update and store new data in the database.
* **Abnormal cases**
  + In case the board does not contain any column: the back end will return nothing and in the front end will be render only add a new column button
  + In case there are not any cards in any column: the front end will be render only columns. user can add a new card inside each column
  + In case a card are moving around and moving back to it previous position: nothing will change, the data will not be updated to the database via backend api request
  + In case a card is moving and placed to a not-a-column destination: the card will be placed back to it previous destination. board data will not be updated or modified.

1. **Archive column**

* **Function trigger**: When the user is inside a note screen, in the Kanban board section. In each column, there is a button to delete that column. this function will be actuated when the user hit that button.
* **Function description**: This function will set the active status of the chosen column to false and hide it from the user. Cards inside archived column will also be lost. Confirmation of doing this action will be required.
* **Screen layout**



* **Function detail**: When the user hit the archived column button in a specific column, a pop-up will be displayed. This pop-up will ask the user to confirm their action. If the user hit the cancel button, the action will not be processed (nothing will be changed). In case the user hit the ok button. A delete request with the chosen column’s id will be sent to the server side from the front end. The server side will set the active status of the column with the given id to false and update its data in the database. The server will send a success message to the front-end side. The entire page will be reloaded in order for the frontend side to fetch new updated Kanban board data from the server. In fact, the column is not deleted from the database but hide it to the user. Because the archived columns will be hidden from the user. All cards in those columns will also be hidden from the user.
* **Abnormal cases**
  + In case the board does not contain any column: the user will not be able to use this function
  + In case the user hits everywhere else except buttons in the pop-up: the pop-up will disappear and nothing will be changed
  + In case server fails to archive the column: nothing will change, the note screen will be reloaded.

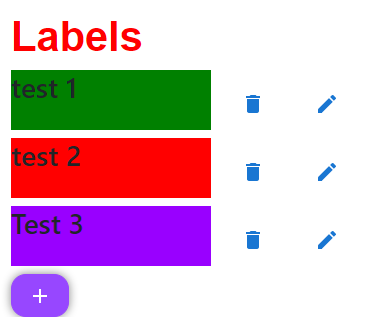
1. Quick Add note from dashboard
   1. Function trigger: When the user is inside Task Management Dashboard, there will be a button ‘+’ to add note
   2. Function description: This function will add new note to user’s workspace
   3. Screen layout
   4. Function details: this function will
   5. Abnormal cases
2. Edit note
   1. Function trigger: When the user
   2. Function description: This function will
   3. Screen layout
   4. Function details
   5. Abnormal cases
3. Archive note
   1. Function trigger: When the user
   2. Function description: This function will
   3. Screen layout
   4. Function details
   5. Abnormal cases
4. Add card to column in Kanban board
   1. Function trigger: When the user
   2. Function description: This function will
   3. Screen layout
   4. Function details
   5. Abnormal cases

## Labels

1. **List All Label in board Labels**

* **Function trigger**: When the component is rendered or when the value of the "data1" state changes.
* **Function description:** The LabelsList component is responsible for rendering a list of labels associated with a board. It provides functionality for adding, editing, and deleting labels. The component fetches the labels from the backend API and renders them on the screen. It also handles the state and logic for editing labels and updating the labels in the backend.

**-Screen layout**



* **Function detail:** The component uses the useState hook to manage the state variables. It initializes the state variables such as labelsList, boardId, editLabel, showLabels, data1, and count. The useEffect hook is used to fetch the labels from the backend API when the component is rendered or when the value of the "data1" state changes. The fetchLabels function is responsible for making a GET request to the backend API to fetch the labels associated with the board. The handleAddLabels function navigates the user to the addlabel screen when the "Add Label" button is clicked. The deleteLabel function is responsible for making a DELETE request to the backend API to delete a specific label. The handleEditLabel function sets the editLabel state variable to the selected label for editing. The handleSaveLabel function is responsible for making a PUT request to the backend API to update the edited label. The handleCancelEdit function resets the editLabel state variable to null, canceling the editing operation. The handleShowLabels and handleHideLabels functions control the visibility of the labels on the screen. The component renders the list of labels using the map function on the labelsList state variable. When the editLabel state variable is not null, it renders the edit label form with input fields for name and color. When the editLabel state variable is null, it renders the label details with buttons for deleting and editing the label.
* **Abnormal cases:**

**-** If there is an error fetching the labels from the backend, an error message should be logged to the console.

- If the DELETE request to delete a label fails, an error message should be logged to the console.

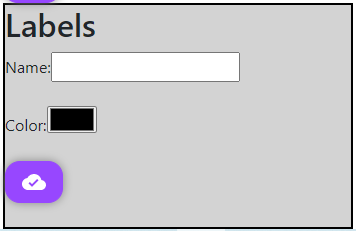
- If the PUT request to update a label fails, an error message should be logged to the console.

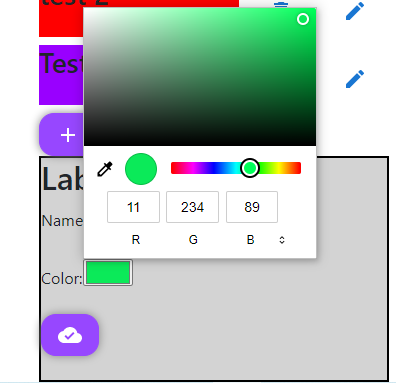
* 1. **Create new Label in board Labels**

**Function trigger**: When the component is rendered or when the "show" state variable changes.

* **Function description:** The Labels component is responsible for rendering a form to add new labels. It provides functionality to create a new label and send the data to the backend API. The component manages the state for the new label's name, color, and the visibility of the form.

**Screen layout:**

****

****

* **Function detail:** The component uses the useState hook to manage the state variables. It initializes the state variables such as show, newLabel, and count. The handleOnClick function toggles the visibility of the form when the "Add" button is clicked. The createLabel function is responsible for making a POST request to the backend API to create a new label. The function sends the newLabel data in the request body. If the POST request is successful, it logs a success message to the console and triggers the onDataChangechild function passed as a prop to notify the parent component of the data change. If the POST request fails, it logs an error message to the console. The component renders a button with an "Add" icon, and when clicked, it toggles the visibility of the label creation form. The form includes input fields for name and color, and a button with a "CloudDone" icon to submit the form. When the form is submitted, the handleOnClick function is called to create the label. The form also resets the input fields to empty values after successful submission.
* **Abnormal cases:** If there is an error creating the label and the POST request fails, an error message should be logged to the console. If the response from the backend API is not successful (status code other than 200), an error message should be logged to the console.