**System Features and Functional Requirements**

**Login**

1. For returning users, they should be able to login
   1. Users should be able to freely sign in / out at their comfort
   2. System will validate the login credentials and display error message if login is unsuccessful
   3. If they forget their password, they should be able to reset their password using the registered email
2. Database will be able to return the correct data when required and determine if a login is successful or not

**Register**

1. For new users , they must sign up for a new account before being able to login
   1. The email provided by the user must be unique
   2. Password provided by the user must be unique
      1. Password must be at least 8 character long
      2. Password length should not exceed 20 characters long
      3. Password should contain at least one upper case character
      4. Password should contain at least one lower case character
2. Database must be able to store this new data into the existing database
   * 1. Hashing will be done to ensure their privacy is not compromised in any way

**Reset Password**

1. For existing user , they should be able to reset their password when they want to
   1. Database will be able to generate a new random password and send to the user email address when required to.
   2. Users will be allowed to change password to their own choice upon the first successful login after resetting password.
2. Database will be able to update the new password that the user keyed in
   1. New password must satisfy the same requirement as when creating a password for a new account

**Search**

1. Users should be able to browse all the courses available in NTU
   1. If needed, they can apply filters to their search
   2. Filters include:
      1. Year of Study
      2. Course they are in
      3. Semester Offered
      4. Prerequisites to the course
2. Database will be able to display the courses correctly when requested to
   1. If filters are applied, the results will be based on the filters
   2. If no filters are applied, result will be displayed in alphabetical order

**Creating New Thread**

1. For users , they should be able to start a new thread under any module
   1. The title section should not be empty
      1. Word limit restricted to 50 words
   2. The description section should not be empty
      1. Word limit restricted to 500 words
   3. If needed, user are able to upload attachments to the application if needed
      1. Restricted to image file type
2. Database should be able to store all these threads correctly under the correct module

**Commenting under a thread**

1. For users , they should be able to freely leave comments on existing threads under any modules
   1. Comment section should not be empty
      1. Word limit restricted to 500 words
   2. Database will be able to store the comments correctly

**Viewing of comments / threads**

1. Users should be able to view all the comments under a particular module
   1. Comments can be sorted by different filters
      1. Date posted
      2. Number of replies to a particular thread
      3. Number of upvotes

**Pinning of modules / threads**

1. Users should be able to pin the threads / modules that they are interested in
   1. These pinned threads / modules will be added to the homepage of the application

**Deleting of comments / threads**

1. Users should be able to delete their threads / comments if they want to
   1. User will click on the little cross at the right side of their thread / comment
      1. Display confirmation message to make sure that user will not accidentally delete their thread / comment
   2. After deletion of the comment / thread , the original content will be changed to [DELETED] , while any existing comments on it will remain unchanged
2. Databases should be able to make the required changes when they are tasked to.

**Other Nonfunctional Requirements**

**Performance Requirements**

1. The application shall load up within 5 seconds.
2. The home screen shall be displayed within 1 second of opening the application.
3. The module search will display results within 5 seconds from search.
4. The threads under each module will display results within 3 seconds from clicking into a module.
5. The comments under a particular thread shall take within 2 seconds to load
6. Posting of a new comment will take at most 2 seconds to do so.
7. Posting of a new thread under a module will take at most 5 seconds to do so.
8. All error messages will be displayed within 0.5 second
9. The database shall store at most 1000 users records
10. Application will be able to support at most 500 users using the application at the same time

**Safety Requirements**

1. The passwords key in by the user is hashed using salt hashing to ensure security and privacy
2. User’s login credentials are encrypted in the database to ensure security and privacy to each user

**Security Requirements**

1. If the user keys in the wrong password more than 10 times , his account will be locked
2. Software used to develop the application should be industry recognised with a certain amount of credibility.
3. User has to login successfully to post threads and comments.
4. Users will be prompted to re-login if he/she has not logged in to the application in a long period of time.

**Software Quality Attributes**

**Reliability**

1. The mobile application should run smoothly and should not crash frequently
2. Application should handle little or no downtime
3. There will be daily update of database to the database to ensure the reliability of the database
4. The application should be usable even when there is a maintenance of the backend by segregating the front-end of the application from the back-end of the application

**Maintainability**

1. The implementation , design of the application is done with consideration of the future maintenance fees in mind.
2. Documentation shall be done in a clear and concise manner to ensure future additions or maintenance can be done by anyone as long as they have the documentation in hand

**Reusability**

1. Implementation of the application is done so in modules to allow reusing.

**Portability**

1. The mobile app must be able to work on various sizes of the mobile screen.

**Data Dictionary**

| User Data | | | | |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Data Format | Description | Example |
| id | Int |  | Unique user ID assigned to upon account creation | 00142 |
| username | Text |  | Unique name for all users | JoeZhao |
| registered\_date | Date | DD/MM/YYYY | Date of user’s registration | 02/02/2022 |
| taken\_mods | List | id1, id2, … | List of module ids that the user has taken | CE2003, CE2006, CE3001 |
| pinned\_thread\_ids | List | id1, id2, … | List of thread ids that the user wants to follow | 00031, 00103, 00463, 00543 |
| pinned\_mods | List | id1, id2, … | List of module IDs that the user wants to follow | CE2006, CE3006 |
| posted\_thread\_ids | List | id1, id2, … | List of thread ids that the user has posted | 00031, 00103, 00463, 00543 |
| posted\_reply\_ids | List | id1, id2, … | List of reply ids that the user has posted | 00051, 00117, 00931 |
| Modules List | | | | |
| Field Name | Data Type | Data Format | Description | Example |
| mod\_ids | Text | id1, id2, … | Module ID (or list of IDs if there are multiples) | CE3005, CZ3006 |
| name | Text |  | Module Name | COMPUTER NETWORKS-Net-Centric Computing |
| thread\_ids | Text | id1, id2, … | List of thread ids under this module | 00031, 00103, 00463, 00543 |
| Threads List | | | | |
| Field Name | Data Type | Data Format | Description | Example |
| thread\_id | Int |  | Unique ID assigned to this thread upon creation | 08412 |
| thread\_name | Text |  | Name of the thread | How is CE2006? |
| submitted\_user\_id | Int |  | ID of the thread creator | 00142 |
| submitted\_time | Date and time | dd-MM-yyyyT HH:mm:ssZ | Thread submission time | 02-02-2022T-8:00:00Z |
| thread\_content | Text |  | Additional content for the thread (optional) | - |
| thread\_reply\_ids | List | id1, id2, … | List of reply ids under this module | 00051, 00117, 00931 |
| Replies List | | | | |
| Field Name | Data Type | Data Format | Description | Example |
| reply\_id | Int |  | Unique ID assigned to this reply upon creation | 00612 |
| reply\_content | Text |  | Content of the reply |  |
| submitted\_user\_id | Int |  | ID of the user that replied | 00142 |
| submitted\_time | Date and time | dd-MM-yyyyT HH:mm:ssZ |  | 02-02-2022T-8:00:00Z |

**Use Case Diagram**

<https://lucid.app/lucidchart/1c129cd4-7e8f-4883-b10b-fb4671ed1da6/edit?invitationId=inv_1d85ea09-0ad6-4ba5-b62b-064039b849f9>

**Use Case Description**

| Use Case ID: | 1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Login to user account | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Gmail API |
| --- | --- |
| Description: | Users need to login to use the app. |
| Preconditions: | 1. User has downloaded the app 2. User has registered an account |
| Postconditions: | 1. App homepage is loaded |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. The user is prompted with Google’s login page 2. User will enter it's email address and password to log in. 3. Application system will then verify the email address and password with Gmail API. 4. Gmail API will send a message to the system if the login is successful 5. Upon successful login, the homepage of the application will be shown |
| Alternative Flows: | AF-S4: If the user inputs incorrect password   1. If a user inputs the incorrect password, an error message will be shown. 2. The user can input password again. |
| Exceptions: | 1. The user can choose to forget the password or register an account (if the account does not exist). |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: | 1. Login is in english |
| Notes and Issues: |  |

| Use Case ID: | 2 | | |
| --- | --- | --- | --- |
| Use Case Name: | Search for modules of interest | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Module database |
| --- | --- |
| Description: | Users can search for a specific module by typing in into the search box. |
| Preconditions: | 1. User login successfully. |
| Postconditions: | 1. The module interface is loaded |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User enters the course code/name of the module into the search box. 2. System search for the particular module from its database. 3. User selects the correct module from the result obtained from the database. |
| Alternative Flows: | AF-S2: If the searched course code/name of the module is not found   1. Message “Module not found!” will be shown. Users will be prompted to input again. 2. The user returns to step 2. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 3 | | |
| --- | --- | --- | --- |
| Use Case Name: | Create new thread | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Thread API |
| --- | --- |
| Description: | User creates a new thread under the module that is selected. |
| Preconditions: | 1. User login successfully. 2. Module course code/name seached is valid. |
| Postconditions: | 1. Thread will be posted successfully under the module information. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User clicks on the “Create New Thread” button. 2. New Thread screen displayed. 3. User enters the title, description and also attaches any attachment if needed. 4. User clicks on the “Post thread” button. |
| Alternative Flows: | AF-S4 : If the user tries to post an empty thread   1. Users will be prompted to input something. 2. The user returns to step 3.   AF-S4 : If the user exceeds the word limit   1. Message “Word limit exceeded” will be shown 2. Users will be required to cut down on the word count. 3. User returns to step 4   AF-S4 : If the user tries to attached an unknown file type   1. Message “Invalid File Type” will be displayed 2. User will return to step 3 |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 4 | | |
| --- | --- | --- | --- |
| Use Case Name: | Add comment | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (initial actor), Thread API |
| --- | --- |
| Description: | Users can add comments to a thread under the module of interest. |
| Preconditions: | 1. User login successfully. 2. There has to be an active thread under module information. |
| Postconditions: | 1. Comment will be posted successfully under the thread. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User clicks on the “reply” button. 2. New Thread screen displayed. 3. User enters the description. 4. User clicks on the “Post comment” button. |
| Alternative Flows: | AF-S4 : If the user tries to post an empty thread   1. Users will be prompted to input something. 2. The user returns to step 4.   AF-S4 : If the user exceeds the word limit   1. Message “Word limit exceeded” will be shown 2. Users will be required to cut down on the word count. 3. User returns to step 4 |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 5 | | |
| --- | --- | --- | --- |
| Use Case Name: | Delete thread | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Thread API |
| --- | --- |
| Description: | Users can delete the thread that they created. |
| Preconditions: | 1. User login successfully. 2. User created a thread successfully under the module selected. |
| Postconditions: | 1. Thread will be deleted successfully. 2. Other users will not be able to view the thread anymore. 3. Comments under the thread will also be removed. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User taps on the “Delete Thread” button. 2. Confirmation message “Are you sure you want to delete the thread?” with two options: “Confirm” and “Cancel” 3. User taps the “Confirm” button. |
| Alternative Flows: |  |
| Exceptions: | 1. The user can choose not to delete the thread by pressing the “Cancel” button. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 6 | | |
| --- | --- | --- | --- |
| Use Case Name: | Delete Comment | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Thread API |
| --- | --- |
| Description: | Users can delete the comment they posted. |
| Preconditions: | 1. User login successfully. 2. User posted a comment successfully under the thread. |
| Postconditions: | 1. Comment will be deleted successfully. 2. Other users will not be able to view the comment anymore. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User taps on the “Delete Comment” button. 2. Confirmation message “Are you sure you want to delete the comment?” with two options: “Confirm” and “Cancel” 3. User taps the “Confirm” button. |
| Alternative Flows: |  |
| Exceptions: | 1. The user can choose not to delete the comment by pressing the “Cancel” button. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 7 | | |
| --- | --- | --- | --- |
| Use Case Name: | Pin Thread | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Thread API |
| --- | --- |
| Description: | Users can pin a thread of the module that they are interested in. |
| Preconditions: | 1. User login successfully. 2. There has to be an active thread under module information. |
| Postconditions: | 1. Thread will be pinned and displayed on the homepage of the user. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User selects the pin icon displayed beside the thread of interest. 2. A message “Thread pinned!” will be shown and the pin icon will be bolded. 3. Thread will now appear on the homepage of the user. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 8 | | |
| --- | --- | --- | --- |
| Use Case Name: | Register Account | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Gmail API |
| --- | --- |
| Description: | Users can register an account. |
| Preconditions: | 1. User has downloaded the app. 2. User does not has registered an account |
| Postconditions: | 1. App login page is loaded |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. The user is prompted with Google’s login page. 2. User taps on the “Register” button. 3. User will enter it's full name, email address and password to create an account. 4. Application system will then send the email address and password to Gmail API. 5. Gmail API will send a message to the system if the registration is successful 6. Upon successful registration, the login page of the application will be shown 7. User will then enter its email address and password to login |
| Alternative Flows: | AF-S3: If the user does not have a valid address   1. If the input address does not exist, an error message will be shown. 2. The user can choose to re-enter a valid address. 3. After re-entering email, the user will attempt to create the account. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 9 | | |
| --- | --- | --- | --- |
| Use Case Name: | View pinned modules/thread | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Thread API |
| --- | --- |
| Description: | Users can view the pinned modules/threads. |
| Preconditions: | 1. User login successfully. |
| Postconditions: | 1. Pinned threads/modules will be shown on the homepage of the user. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1.Upon logging in, a list of pinned threads/modules will appear on the homepage |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | Use case 4 |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 10 | | |
| --- | --- | --- | --- |
| Use Case Name: | View module information | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor), Database API |
| --- | --- |
| Description: | Users can view information regarding the module. |
| Preconditions: | 1. User login successfully. 2. Module course code/name seached is valid. |
| Postconditions: | 1. Module information page will be loaded. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1.User will click on the module name in the module page.   1. The module information will be shown. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 11 | | |
| --- | --- | --- | --- |
| Use Case Name: | View comments under module | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor),Database API |
| --- | --- |
| Description: | Users can view the comments under a selected module thread. |
| Preconditions: | 1.User login successfully.  2. There has to be an active thread under module information. |
| Postconditions: | 1. Page with the thread’s comments will be loaded |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1.Upon clicking the thread, the comments of the thread will be shown |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 12 | | |
| --- | --- | --- | --- |
| Use Case Name: | View threads under module | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User (Initiating actor),Database API |
| --- | --- |
| Description: | Users can view the threads under a selected module. |
| Preconditions: | 1.User login successfully.  2. Module course code/name seached is valid. |
| Postconditions: | 1.Module information page will be loaded. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1.Upon clicking the module, the threads of the module will be shown below the course information. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | Use case 4 |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

**Class diagram**

<https://lucid.app/lucidchart/5e18e2e9-110f-4114-8833-c71eab3cf31a/edit?invitationId=inv_2493186e-8bfd-4077-9837-4d4fcbecc2e3>

**Sequence diagram**

<https://lucid.app/lucidchart/521e0e5d-bcb7-4225-9423-655254665887/edit?invitationId=inv_44cea412-58e5-4e35-871b-66d4bf0832cd>

**Initial Dialog map**

<https://lucid.app/lucidchart/914b5d65-516f-42c6-96b2-4211841196db/edit?invitationId=inv_e2806258-dbe7-409c-9751-b3be0125640e>