Requirements

**Functional Requirements**

1. User should be able to browse all the courses available in NTU

1.1. If needed, they can apply filters to their search

1.2. Filters include:

1.2.1. Year of Study

1.2.2. Course they are in

1.2.3. Semester Offered

1.2.4. Prerequisites to the course

2. User should be able to start a comment thread under a module

2.1. There will be a word limit of 500 words

3. User should be able to upload attachments to the application if needed

3.1. Restricted to image file type

4. User should be able to view all the comments under a particular module

4.1. Information can be sorted by the date posted, number of replies to a particular thread, number of upvotes

5. User should be able to register for a new account

5.1. Information to be included:

5.1.1. Name

5.1.2. Valid Gmail Address

5.1.3. Password (containing at least one uppercase and lowercase character)

6. User should be able to login with their credentials

6.1. System will validate that both fields (name & passwords have been filled up)

6.2. If login information is incorrect, the system will display an error message

7. Users should be able to pin the threads and modules that they are interested in

8. User should be able to reply to a comment thread

8.1. There will be a word limit of 500 words

**Non-Functional Requirements**

1. User should be able to navigate around the application within 5 minutes of using it

2. The app should load the details and comments regarding the module 30 seconds after clicking a specific module

3. User shall be prompted to provide their email for authentication before entering the application

4. The application must allow reversal of their actions

4.1. Users can bookmark / remove their pinned mods / threads

5. Upon login, the system will perform salt – hashing on the password for security requirements

6. The application should allow the user to search for a module

7. System must be able to maintain little or no downtime

8. The app should be capable to handle 50 000 users without affecting its performance

9. The application should allow the user to reset the password if the user forgets their password

| Functional | Non-Functional |
| --- | --- |
| User should be able to browse all the courses available in NTU. | Users should be able to navigate through the app after 5 minutes of seeing it. |
| User should be able to start a thread under a module(500 words) | The app should load the details and comments regarding the module 30 seconds after clicking a specific module. |
| User should be able to upload image file types of attachments. | Users shall be prompted to provide their email for authentication before entering the app. |
| Users should be able view all the threads under a module  Information can be sorted by the date posted | The app must allow reversal of their actions, user can bookmark / remove their pinned mods / threads |
| User should be able to register a new account.  Information to be included: name, valid email address, password | Upon login, the system will perform salt - hashing on the input password for security requirements |
| User should be able to login with their credentials.  System will validate that both fields ( name & password have been filled up )  If login information is incorrect, the system will display an error message | The app should allow user to search for the module. |
| User should be able to pin the threads and modules that they are interested in. | System must be able to maintain little or no downtime. |
| User should be able to comment on a thread(500 words). | The app should be capable enough to handle fifty thousand users without affecting its performance. |
|  | The app should allow the user to reset the password if the user forgot the password. |

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

Use Case Description

| Use Case ID: | 1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Login to app | | |
| 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 password is incorrect or email is invalid   1. An error message will be shown and prompt the user to input again. 2. The user returns to step 2 or can choose to reset the password if the user forgot the password. |
| Exceptions: | 1. If any of the login details do not match to any user account (user account does not exist), an error message is shown. 2. Users can re-enter their login details or choose to register an account. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 2 | | |
| --- | --- | --- | --- |
| Use Case Name: | Search for modules | | |
| 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 and 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: | Commenting to a thread | | |
| 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: |  |
| 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: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 7 | | |
| --- | --- | --- | --- |
| Use Case Name: | Pinned 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: |  |