BookLeaf

Use-Case Specification

Version <2.0>

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 9/11/2024 | 1.0 | Init the use-case specification | Truong Thanh Toan, Pham Thanh Vinh |
| 21/11/2024 | 2.0 | * Use-case 2.6.3, 2.6.4: redefine *reading goal* and use-cases related to reading goal. * Update the table of content. * Define the maximum number of tags and libraries a user can create. * Table of contents: divide use cases into groups related to a main feature (such as Account, Book, Tag, Library, Social,...) * Restructure use-case specification (section 2): separate use cases to 2 groups - group of general user/normal users and group of admin users. | Truong Thanh Toan |
| * Assign a priority for each use case. The use case that has the higher priority will be done first, and the lower afterwards. By doing so, the group of low priority use-cases could be ignored if the timeline is overrun. | Pham Tran Yen Quyen |
| * Add the use-case model for admin users. | Pham Thanh Vinh |

Table of Contents

[**1. Use-case Model 5**](#_heading=h.gjdgxs)

[**2. Use-case Specifications 10**](#_heading=h.30j0zll)

[**2.1 Use cases for general users 10**](#_heading=h.4epuownjj7ld)

[2.1.1 Use-case: Edit user profile (username) 10](#_heading=h.9wiilrgv2cm3)

[2.1.2 Use-case: Edit user profile (password) 11](#_heading=h.kas8d9uevkdz)

[2.1.3 Use-case: Edit user profile (email) 12](#_heading=h.pfa4dm610vv4)

[2.1.4 Use-case: Edit user profile (avatar) 13](#_heading=h.96d1nxdwdn4i)

[2.1.5 Use-case: Self-delete the account 13](#_heading=h.qssl1fzi2182)

[*2.1.6 Use-case: Search for another normal user 14*](#_heading=h.gbxtmpon9zh6)

[**2.2 Use cases for normal users 14**](#_heading=h.9cahtxll8wqs)

[*2.2.1 User’s account 14*](#_heading=h.qrbvj9v2ma8o)

[2.2.1.1 Use-case: Register a new normal user account 14](#_heading=h.2et92p0)

[2.2.1.2 Use-case: Log in to a normal user account 16](#_heading=h.2fahoxgoprak)

[*2.2.2 Books 16*](#_heading=h.v4wx6s6fg9qy)

[2.2.2.1 Use-case: Upload books 16](#_heading=h.8jleowecnhwl)

[2.2.2.2 Use-case: Download a book from cloud 17](#_heading=h.a8y0nrf4s9d9)

[2.2.2.3 Use-case: See the list of uploaded books 17](#_heading=h.69yxl980yevj)

[2.2.2.4 Use-case: Search books (by name/author/tags/categories) 18](#_heading=h.tb1m6wnuppf2)

[2.2.2.5 Use-case: Sort books (by name A - Z/name Z - A/reading progress ascendingly/reading progress descendingly) 18](#_heading=h.xmbxzkcpq367)

[2.2.2.6 Use-case: Delete a book 19](#_heading=h.mh7wz2wbc7b3)

[*2.2.3 Tags 20*](#_heading=h.x3kj77nlj6kl)

[2.2.3.1 Use-case: Create a tag 20](#_heading=h.p5slynuyiwt5)

[2.2.3.2 Use-case: Edit a tag name 20](#_heading=h.ejs55x1lq09o)

[2.2.3.3 Use-case: Delete a tag 21](#_heading=h.pwxslas75ul6)

[2.2.3.4 Use-case: Set tags for a book 21](#_heading=h.t26j9xai1eay)

[*2.2.4 Library 22*](#_heading=h.opqain44ap8r)

[2.2.4.1 Use-case: Create a library 22](#_heading=h.r0v3stravxcp)

[2.2.4.2 Use-case: See the list of created libraries 23](#_heading=h.8bvunsan2ocl)

[2.2.4.3 Use-case: Delete a library 23](#_heading=h.h44cmxkqd8ac)

[2.2.4.4 Use-case: Edit the library (name) 23](#_heading=h.b4cowm1n05fw)

[2.2.4.5 Use-case: Edit the library (visibility) 24](#_heading=h.mutrxuboq25u)

[2.2.4.6 Use-case: Edit the library (add books) 24](#_heading=h.8bw186jweh5d)

[2.2.4.7 Use-case: Edit the library (remove a book) 25](#_heading=h.rgwisj4thksc)

[*2.2.5 Reading features 25*](#_heading=h.7ya302pvfvlq)

[2.2.5.1 Use-case: Read book 25](#_heading=h.e7djxsmxq8n5)

[2.2.5.2 Use-case: Add highlight 26](#_heading=h.1jw52c5tn6x1)

[2.2.5.3 Use-case: Remove highlight 27](#_heading=h.36v1dfqfb5bw)

[2.2.5.4 Use-case: Add note 27](#_heading=h.vzn887p4xw6n)

[2.2.5.5 Use-case: Edit note 28](#_heading=h.xlfxupmfplr7)

[2.2.5.6 Use-case: Remove note 29](#_heading=h.uzipslgh294e)

[2.2.5.7 Use-case: Add bookmark 29](#_heading=h.m0s3r9797qgz)

[2.2.5.8 Use-case: Remove bookmark 30](#_heading=h.vztf8rhizr5v)

[2.2.5.9 Use-case: Change font 31](#_heading=h.tv77egs9ljsr)

[2.2.5.10 Use-case: Change font size 31](#_heading=h.gyqycn2cbwiz)

[2.2.5.11 Use-case: Adjust line spacing 32](#_heading=h.wfjl8w6535ex)

[2.2.5.12 Use-case: Change reading theme 32](#_heading=h.ak1mjnihsjey)

[*2.2.6 Self analytics 32*](#_heading=h.gb9kbt1m7rb2)

[2.2.6.1 Use-case: View reading analysis of a book 32](#_heading=h.xz3fnuuqu6t3)

[2.2.6.2 Use-case: Track time spent 33](#_heading=h.87zctnyhj3x1)

[2.2.6.3 Use-case: Set/Edit a reading goal 33](#_heading=h.ff5m8ycjurq4)

[2.2.6.4 Use-case: Delete a reading goal 34](#_heading=h.2us6fdenhzwk)

[*2.2.7 Social 34*](#_heading=h.51s1m410zhud)

[2.2.7.1 Friend request 34](#_heading=h.k88485yvlkmm)

[2.2.7.1.1 Use-case: Send a friend request to another normal user 34](#_heading=h.nsjhkgklqwc)

[2.2.7.1.2 Use-case: Cancel a friend request to another normal user 34](#_heading=h.owl3rz4a73j)

[2.2.7.1.3 Use-case: Accept a friend request from another normal user 35](#_heading=h.aa51re6av6vh)

[2.2.7.1.4 Use-case: Decline a friend request from another normal user 35](#_heading=h.9ovjmtjfdhb0)

[2.2.7.2 Relationship management 36](#_heading=h.u4hddp815s62)

[2.2.7.2.1 Use-case: Remove a friend 36](#_heading=h.c02g9f7ojy1)

[2.2.7.2.2 Use-case: Normal user blocks another normal user 36](#_heading=h.5r8ii05u67nq)

[2.2.7.2.3 Use-case: Normal user unblocks another normal user 37](#_heading=h.x7rhnsdd6ena)

[**2.3 Use cases for admin users 37**](#_heading=h.1w7x4nxd36xt)

[2.3.1 Use-case: Register a new admin account 37](#_heading=h.bd7y1tvb7lsq)

[2.3.2 Use-case: Log in to an admin account 39](#_heading=h.2exb0b1i88fm)

[2.3.3 Use-case: Delete a normal user’s account 39](#_heading=h.3lxf9k2dm02u)

[2.3.4 Use-case: See all other user’s books/libraries 40](#_heading=h.xvab3ao218lx)

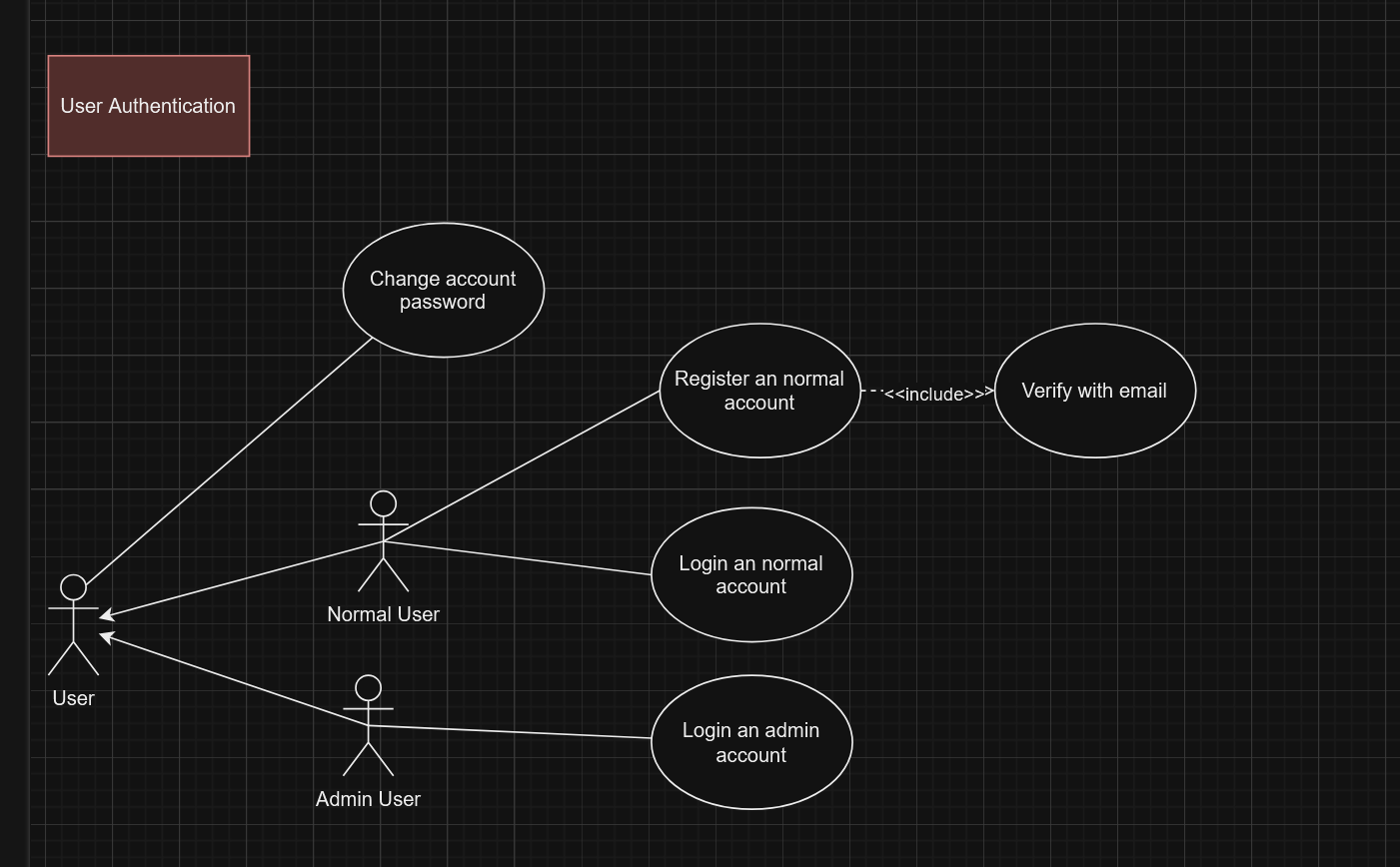
[2.3.5 Use-case: Delete a book by admin 40](#_heading=h.hl9luehtodz8)

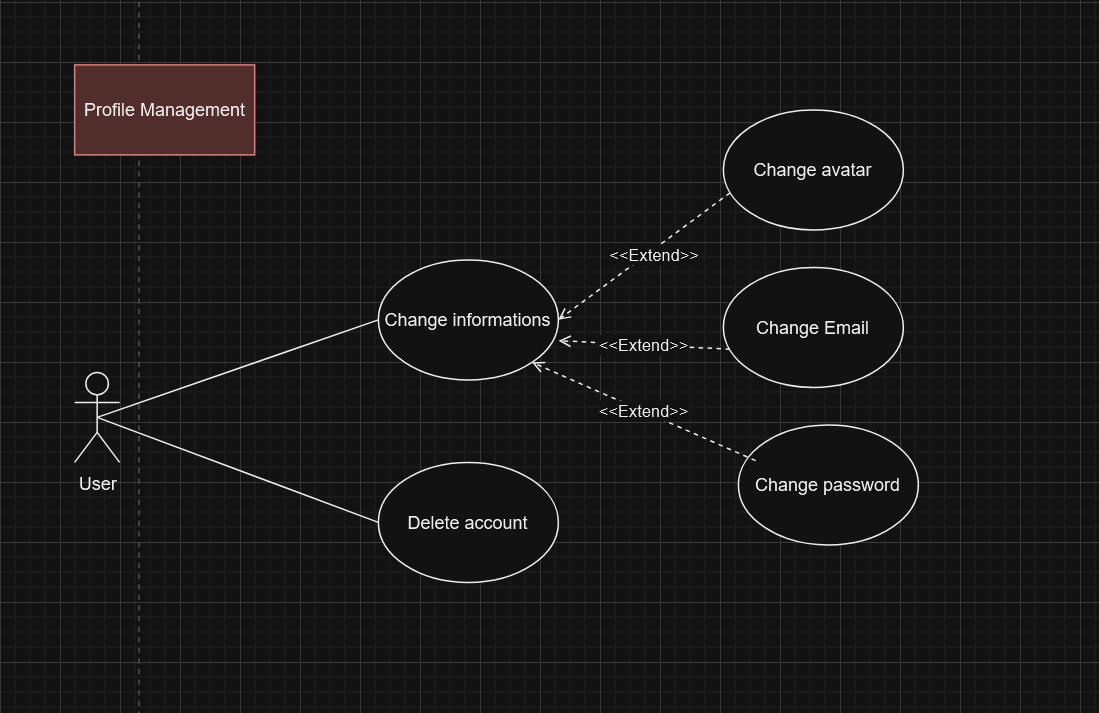
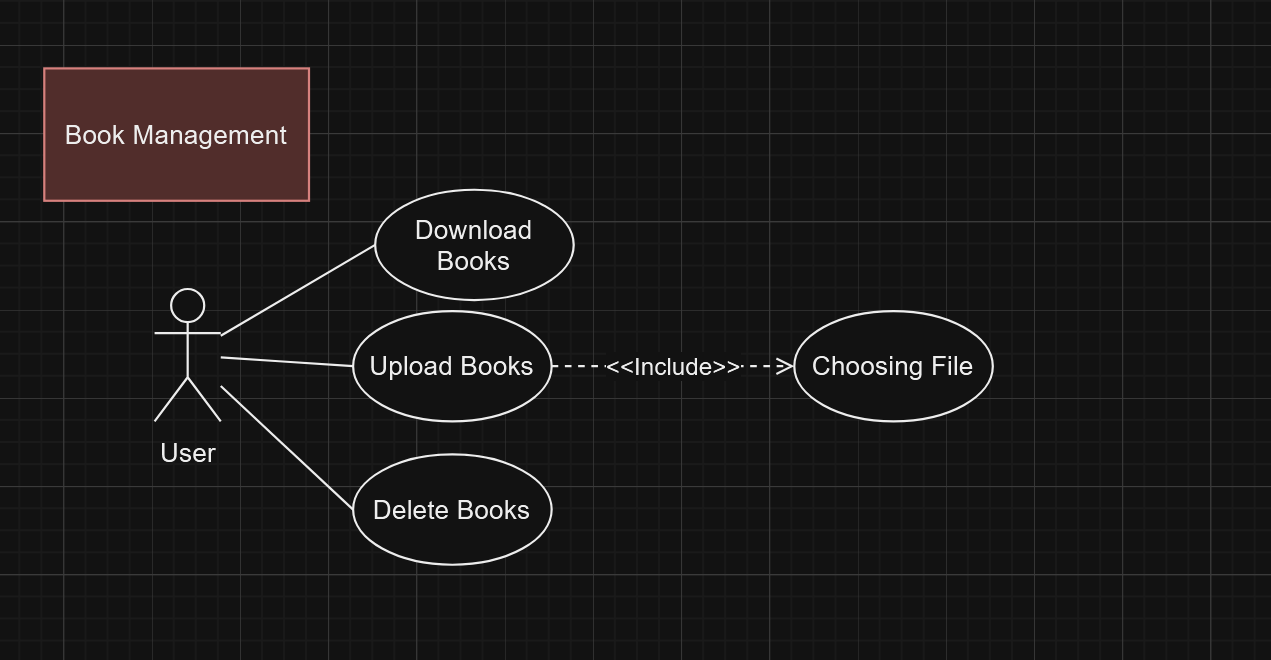
[2.3.6 Use-case: Overview analytics 40](#_heading=h.hm1yecc1wqfu)

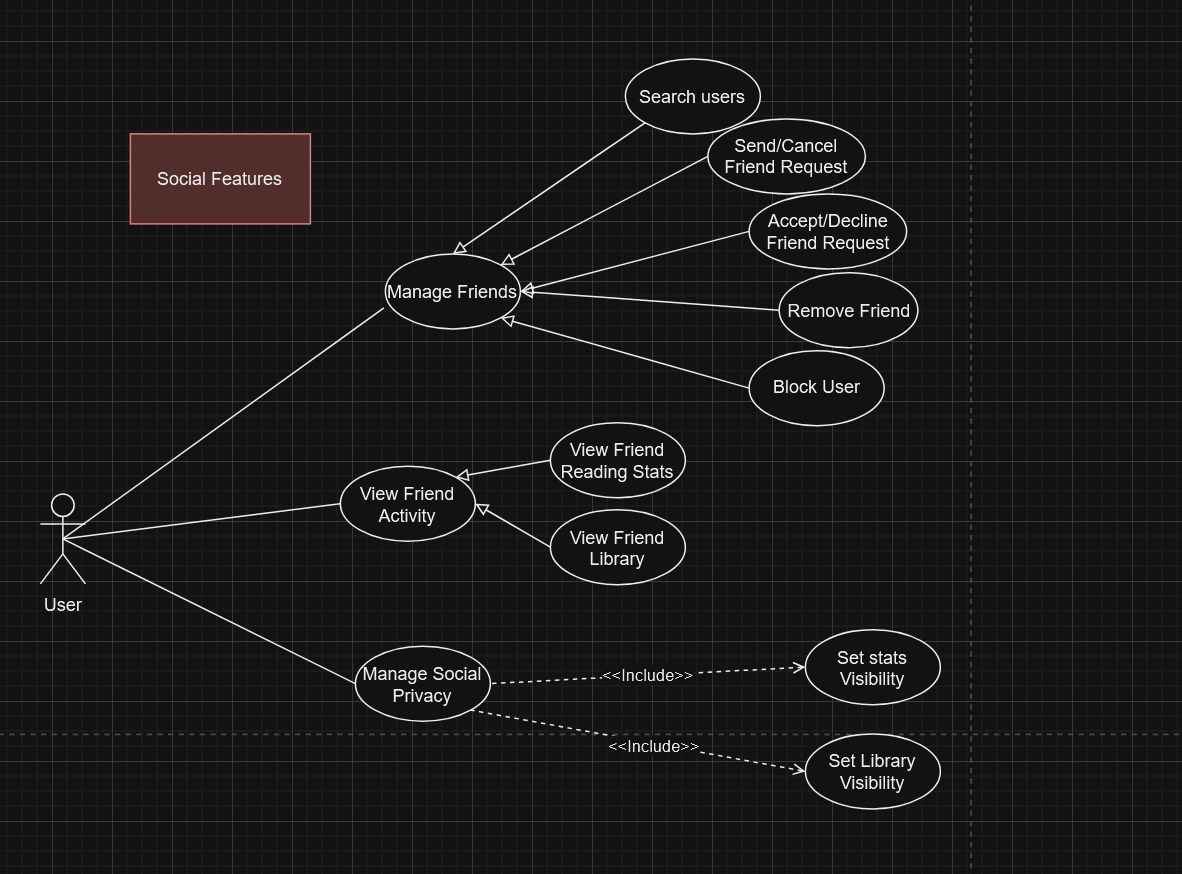
[2.3.7 Use-case: Monitor reading progress 41](#_heading=h.c3osz34nz9w3)

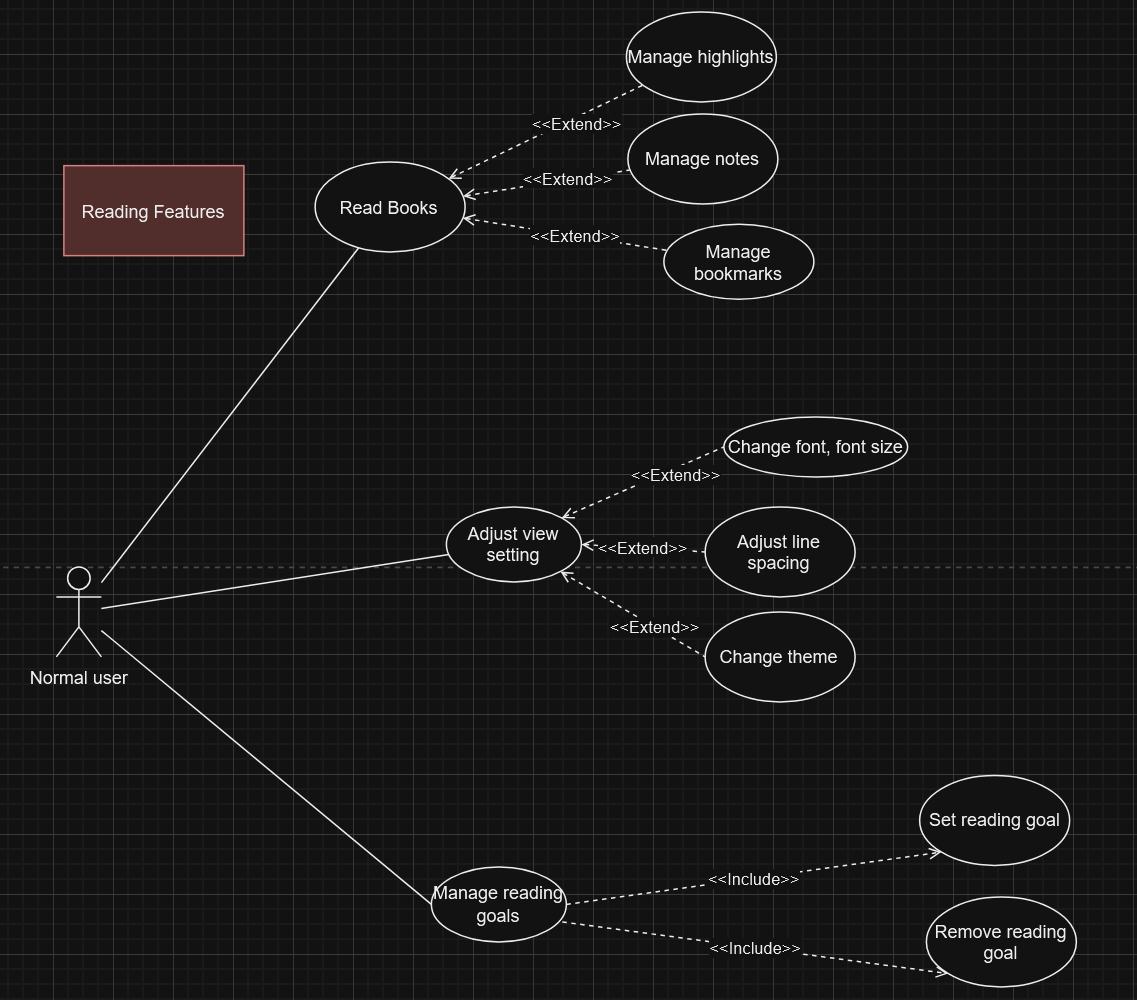
[2.3.8 Use-case: Monitor user session 41](#_heading=h.49vh4jvqlwyt)

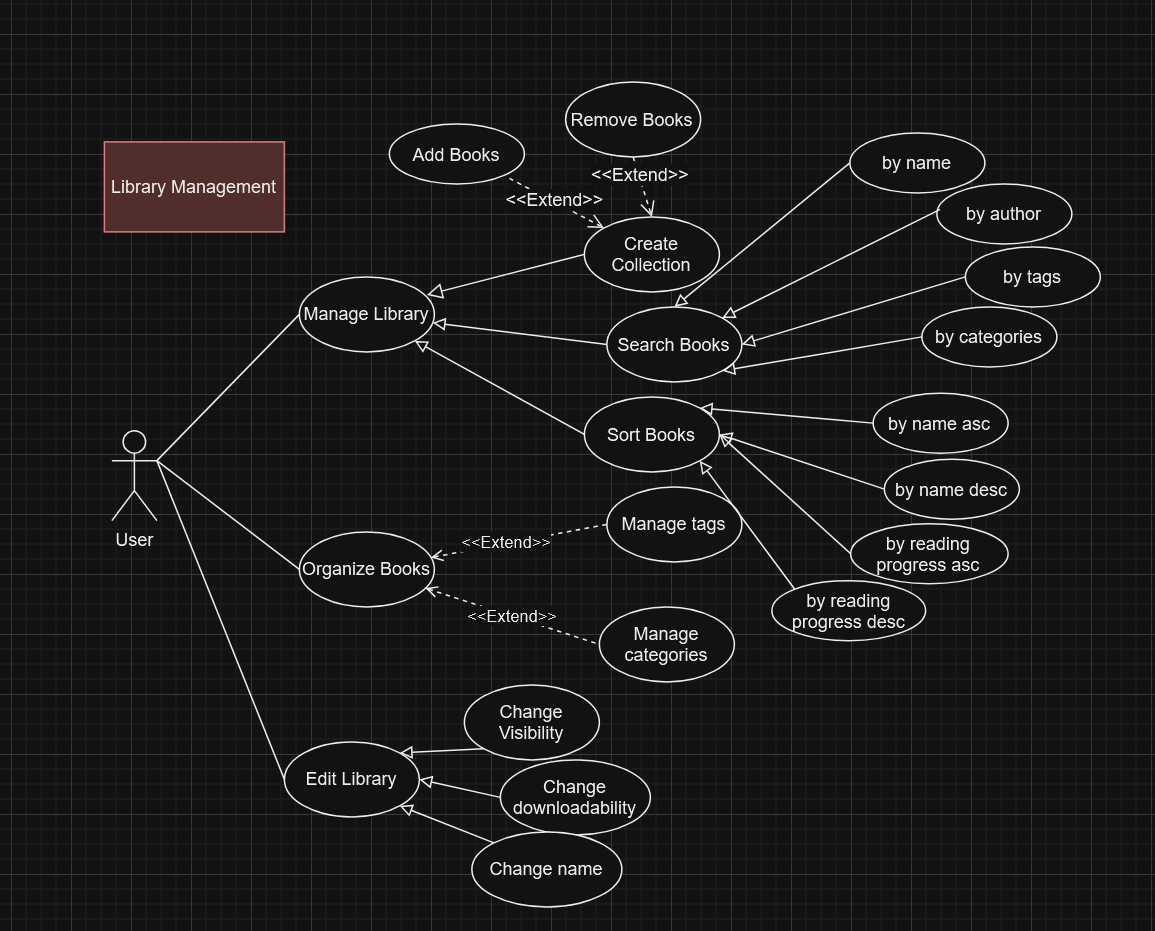
# Use-case Model











# 

# Use-case Specifications

## Use cases for general users

## Use-case: Edit user profile (username)

| Use case Name | Edit account’s username |
| --- | --- |
| Brief description | The use case describes how a user (in general) can edit their username. |
| Actors | User. |
| Basic Flow | 1. User navigates to their user profile. 2. User clicks the “edit” icon to start editing. 3. System allows the user to change their username. 4. User enters the new username: shall not contain any whitespaces; only contains a-z, A-Z, 0-9, period(s) “.”, and underscores “\_”. 5. User confirms their changes. 6. System disallows the user to enter a new username (disable the input textbox). 7. System displays the notification “Username changed”. |
| Alternative Flows | **Alternative flow 1: The new username has existed in the database.**   1. From #7 of the basic flow, the system displays a notification “Username used”. 2. Continue from #4 of the basic flow.   **Alternative flow 2: Username is invalid.**   1. From #7 of the basic flow, the system raises a notification: “Username is invalid”. 2. Continue from #4 of the basic flow.   **Alternative flow 3: User cancels the action.**   1. From the step the user cancels the action, the system halts and the username remains. |
| Pre-conditions | The user must have logged in. |
| Post-conditions | User successfully changes their username. |
| Priority | **High** |

## Use-case: Edit user profile (password)

| Use case Name | Edit account’s password |
| --- | --- |
| Brief description | The use case describes how a user (in general) can edit their password. |
| Actors | User. |
| Basic Flow | 1. User navigates to their user profile. 2. User clicks the “edit” icon to start editing. 3. System displays a form for editing the password (including old password, new password, and confirm new password). 4. User inputs all the fields. 5. User confirms their changes. 6. System displays the notification “Password changed”. |
| Alternative Flows | **Alternative flow 1: Any fields were left blanked.**   1. From #6 of the basic flow, system notifies the user that “All fields must be filled in”. 2. Continue from #4 of the basic flow.   **Alternative flow 2: Old password is incorrect.**   1. From #6 of the basic flow, system notifies the user that “Old password is invalid”. 2. Continue from #4 of the basic flow.   **Alternative flow 3: New password is *not* strong enough.**   1. From #6 of the basic flow, system notifies the user that “Password must be strong” (a strong password contains at least 8 characters including lowercase *a-z*, uppercase letters *A-Z*, digits *0-9*, and some of the following special characters *!#\*+-*). 2. Continue from #4 of the basic flow.   **Alternative flow 4: Confirm password and password are not matched.**   1. From #6 of the basic flow, system notifies the user that “Passwords are not matched”. 2. Continue from #4 of the basic flow.   **Alternative flow 5: User cancels the action.**   1. From the step the user cancels the action, the system halts and the old password remains. |
| Pre-conditions | The user must have logged in. |
| Post-conditions | User successfully changes their password. |
| Priority | **High** |

## Use-case: Edit user profile (email)

| Use case Name | Edit account’s email |
| --- | --- |
| Brief description | The use case describes how a user (in general) can edit their email. |
| Actors | User. |
| Basic Flow | 1. User navigates to their user profile. 2. User clicks the “edit” icon to start editing. 3. System allows the user to change their email. 4. User enters the new email: no white spaces, pattern “[example@provider.topdomain](mailto:example@provider.topdomain)”; check email existence. 5. User confirms their changes. 6. System disallows the user to reenter email (disable the input textbox). 7. System sends OTP to the user to verify the new email.   ***\* OTP will expire after 15 minutes***   1. User enters the OTP code for verification. 2. System displays the notification “Email changed”. 3. System sends an email to the new email address provided by the user to announce that it has been linked to the account. |
| Alternative Flows | **Alternative flow 1: The new email has existed in the database.**   1. From #7 of the basic flow, the system displays a notification “Email address used”. 2. Continue from #3 of the basic flow.   **Alternative flow 2: New email is in an invalid format.**   1. From #7 of the basic flow, the system displays a notification “Invalid email”. 2. Continue from #3 of the basic flow.   **Alternative flow 3: User enters an incorrect code.**   1. From #9 of the basic flow, the system displays a “Wrong verification code.” message. 2. Continue from #8 of the basic flow.   **Alternative flow 4: User enters an expired code.**  (behaves as if the code is incorrect).  **Alternative flow 5: User cancels the action.**   1. From the step the user cancels the action, the system halts and the email remains. |
| Pre-conditions | The user must have logged in. |
| Post-conditions | User successfully changes their email. |
| Priority | **High** |

## Use-case: Edit user profile (avatar)

| Use case Name | Edit account’s avatar |
| --- | --- |
| Brief description | The use case describes how a user (in general) can edit their avatar. |
| Actors | User. |
| Basic Flow | 1. User navigates to their user profile. 2. User clicks the “edit” icon to start editing. 3. System allows the user to change their avatar. 4. User chooses an image and confirms their change. 5. System updates the new avatar and displays the notification “Avatar changed”. |
| Alternative Flows | **Alternative flow 1: User cancels the action.**   1. From the step the user cancels the action, the system halts and the library remains on the list. |
| Pre-conditions | The user must have logged in. |
| Post-conditions | User successfully changes their avatar. |
| Priority | **Low** |

## Use-case: Self-delete the account

| Use case Name | Self-delete the account. |
| --- | --- |
| Brief description | The use case describes how a user can remove their own account. |
| Actors | User. |
| Basic Flow | 1. User clicks the button “Delete account” on their profile. 2. System asks the user to confirm they want to delete the account by entering “CONFIRM DELETE <username>” and clicking the button “Confirm”. 3. System deletes the account and removes the user from all the friend lists they are in. 4. System notifies the user that they has removed their account. 5. System sends an email to the user’s email address that the account linked to the email address has been removed. 6. Log out the user. |
| Alternative Flows | **Alternative flow 1: User inputs the confirm phrase incorrectly.**   1. From #4 of the basic flow, the system displays a notification “Please confirm again”. 2. Continue from #2 of the basic flow.   **Alternative flow 2: User cancels the action.**   1. From the step the user cancels the action, the system halts and the account remains. |
| Pre-conditions | The user must have logged in.  User must navigate to their user profile. |
| Post-conditions | User successfully removes their account. |
| Priority | **Medium** |

## Use-case: Search for another normal user

| Use case Name | **Search for another normal user** |
| --- | --- |
| Brief description | Allows a user to search for another normal user within the application by entering their username. |
| Actors | User. |
| Basic Flow | 1. User navigates to the “Search user” feature in the application. 2. The user enters the name or username of the person they wish to find. 3. The system retrieves and displays a list of users matching the search criteria. |
| Alternative Flows | **Alternative flow 1: No matching users are found.**   1. From #3 of the basic flow, the system displays nothing in the searching result area.   **Alternative flow 2: The user has been blocked by the one they are looking for.**  (behaves as if no matching users are found)   1. From #3 of the basic flow, the system displays nothing in the searching result area. |
| Pre-conditions | The user must be logged in. |
| Post-conditions | The user sees the search results, or nothing if no users are found. |
| Priority | **High** |

## Use cases for normal users

## User’s account

## Use-case: Register a new normal user account

| Use case Name | Register a new normal user account. |
| --- | --- |
| Brief description | The use case describes the flow of normal account registration for both normal end-users and administrators. |
| Actors | Normal user. |
| Basic Flow | 1. System displays the registration form. 2. User enters username, email, password, confirm password (all fields are required). 3. User clicks to the “Register” button (submit). 4. System validates the inputs. Constraints are:    1. No empty field.    2. Username and email have not been used (including email address variations using periods and plus signs).    3. Email must be in a valid form: no white spaces, pattern “[example@provider.topdomain](mailto:example@provider.topdomain)”.    4. All fields contain only printable characters.    5. The password must be strong (at least 8 characters including lowercase *a-z*, uppercase letters *A-Z*, digits *0-9*, and some of the following special characters *!#\*+-*).    6. Username shall not contain any whitespaces; only contains a-z, A-Z, 0-9, period(s) “.”, and underscores “\_”.    7. Confirm password and password must be identical. 5. System notifies the user that there is a verification code sent to the email user provided, and displays an input so that the user can enter the code.   ***The verification code is 6-digit length.***  ***The code will expire after 15 minutes.***   1. User enters the code to the input displayed on the screen. 2. System verifies the code, creates a new account and shows the “Account has been successfully registered” message. |
| Alternative Flows | **Alternative flow 1: Username or email has been used.**   1. From #5 of the basic flow, the system raises a notification: “Username or email has been used”. 2. Continue from #1 of the basic flow.   **Alternative flow 2: Password is not sufficiently strong.**   1. From #5 of the basic flow, the system raises a notification: “Password is not strong enough”. 2. Continue from #1 of the basic flow.   **Alternative flow 3: Username is invalid.**   1. From #5 of the basic flow, the system raises a notification: “Username is invalid”. 2. Continue from #1 of the basic flow.   **Alternative flow 4: Invalid email.**   1. From #5 of the basic flow, the system raises a notification: “Email is invalid”. 2. Continue from #1 of the basic flow.   **Alternative flow 5: Any field is empty.**   1. From #5 of the basic flow, the system raises a notification: “All fields are required”. 2. Continue from #1 of the basic flow.   **Alternative flow 6: Any fields contain non-printable characters.**   1. From #5 of the basic flow, the system raises a notification: “Must use printable Latin character”. 2. Continue from #1 of the basic flow.   **Alternative flow 7: Confirm password and password are not identical.**   1. From #5 of the basic flow, the system raises a notification: “Passwords do not match”. 2. Continue from #1 of the basic flow.   **Alternative flow 8: User enters an incorrect code.**   1. From #7 of the basic flow, the system displays a “Wrong verification code.” message. 2. Continue from #6 of the basic flow.   **Alternative flow 9: User enters an expired code.**  (behaves as if the code is incorrect). |
| Pre-conditions | User navigates the register page of the website ***/register***, either directly or via the login page. |
| Post-conditions | User successfully creates a new account if all the constraints and pre-conditions are met. |
| Priority | **High** |

## Use-case: Log in to a normal user account

| Use case Name | Log in to a normal user account. |
| --- | --- |
| Brief description | The use case describes the flow of logging in to a normal user account. |
| Actors | Normal user. |
| Basic Flow | 1. System displays the login form. 2. User fills in all the fields:  * Username/Email. * Password.  1. User submits the login form. 2. The system informs that the user has successfully signed in and redirects the user to the main page from which he/she can start their journey. |
| Alternative Flows | **Alternative flow 1: Incorrect username/email or password.**   1. From #4 of the basic flow, the system raises a notification: “Username or password is incorrect”. 2. Continue from #1 of the basic flow. |
| Pre-conditions | User navigates to the login page **/** or **/login** of the website. |
| Post-conditions | User successfully logs in to their account if it exists. |
| Priority | **High** |

## Books

## Use-case: Upload books

| Use case Name | Upload books. |
| --- | --- |
| Brief description | A normal user can upload their books to the system database. |
| Actors | Normal user. |
| Basic Flow | 1. User clicks the button “Upload a book”. 2. System shows the list of files on a cloud provider (initially Google Drive) that will be chosen (initially empty). Moreover, it shall go with a button by clicking which a file explorer window will appear. 3. User clicks the “Choose file” button. 4. System allows user to choose files on their device using a file explorer window. 5. User chooses:    1. File(s) on their device: only .epub and .pdf files are allowed.    2. Tags for each book.    3. Categories for each book. 6. System retrieves and processes the file:    1. Format validation.    2. Metadata extraction. 7. System displays the notification: “Books successfully uploaded”. 8. System closes the file explorer window, and new books will be displayed on the screen at the main page. |
| Alternative Flows | **Alternative flow 1: User chooses an invalid file type.**   1. From #4 of the basic flow, the system displays a notification “Only .epub and .pdf files are supported”. 2. Continue from #2 of the basic flow.   **Alternative flow 2: User cancels the action.**   1. From the step the user cancels the action, the system halts and the user’s book collection remains the same |
| Pre-conditions | * User must have logged in. * User navigates the main page ***/books*** displayed after they have just logged in. |
| Post-conditions | * User successfully uploads a book. * New books are displayed in the personal gallery. * New added books are set to private and are unable to be downloaded by other users. |
| Priority | **High** |

## Use-case: Download a book from cloud

| Use case Name | Download a book from cloud. |
| --- | --- |
| Brief description | A normal user can download a book, which allows the user to download, from the cloud storage. |
| Actors | Normal user, Book. |
| Basic Flow | 1. User chooses the download icon on the book. 2. System allows the user to download the book, and the book will be downloaded to the user’s device. |
| Alternative Flows | **Alternative flow 1: User is not allowed to download the book.**   1. From #2 of the basic flow, the system displays a notification “Access denied”. 2. Halt. |
| Pre-conditions | * User must have logged in. * User navigates to a site where they can see the book. * User must have the right to access the book. Accessible books are:   + Books that the user uploaded themselves.   + Books of another user that they share and allow other users to see (visibility) and download. |
| Post-conditions | User successfully downloads the accessible book. |
| Priority | **High** |

## Use-case: See the list of uploaded books

| Use case Name | See the list of uploaded books |
| --- | --- |
| Brief description | A normal user can see the list of their uploaded books. |
| Actors | Normal user, Book. |
| Basic Flow | 1. User navigates to the site ***/books***. 2. System displays all books of the user. |
| Alternative Flows | None. |
| Pre-conditions | User must have been logged in. |
| Post-conditions | User can see the list of their books. |
| Priority | **Medium** |

## Use-case: Search books (by name/author/tags/categories)

| Use case Name | Search books. |
| --- | --- |
| Brief description | User can search in their library some books based on one of the following criteria: books’ name, author, tags, and categories. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates the searching feature on the main page after the user successfully logged in. 2. User chooses a criteria to be based on (initially be “Name”), enters the text and submit the search. 3. System retrieves the books that match and displays the result to the user (if any). |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and everything remains. Note that the system does not delete the text that user entered to the search bar.   **Alternative flow 2: System finds no matching results found.**   1. System displays a message stating: “No books match your search criteria.” 2. User can either return to #2 to modify the search term or criterion, clear the search, or return to the main library view. |
| Pre-conditions | * User is logged in. * User has books in their library that include metadata for name, author, tags, and categories. |
| Post-conditions | Search results matching the specified criteria are displayed in a list format, allowing the user to view, access, or further refine the search. |
| Priority | **High** |

## Use-case: Sort books (by name A - Z/name Z - A/reading progress ascendingly/reading progress descendingly)

| Use case Name | Sort books. |
| --- | --- |
| Brief description | Allows a user to sort the books in their library based on criteria such as alphabetical order or reading progress, enabling easier browsing. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to their library page using the nav bar. 2. User identifies the “Sort” option located at the top of the library list and clicks on it to initiate sorting. 3. System displays sorting options in a dropdown menu, including “Name A-Z,” “Name Z-A,” “Reading Progress Ascending,” and “Reading Progress Descending.” 4. User selects a sorting criterion by clicking one of the options. 5. System reorders the books in the library view based on the selected criterion:    1. Name A-Z: Orders books alphabetically from A to Z.    2. Name Z-A: Orders books alphabetically from Z to A.    3. Reading Progress Ascending: Orders books based on the percentage of reading progress, from least to most completed.    4. Reading Progress Descending: Orders books based on the percentage of reading progress, from most to least completed. 6. System updates the library view to reflect the selected sorting order. |
| Alternative Flows | **Alternative flow 1: User cancels the sorting action without selecting a criterion.**   1. System retains the current sorting order and closes the dropdown menu without changing the display. |
| Pre-conditions | * User is logged in. * User has books in their library with names and reading progress data. |
| Post-conditions | The library view is updated to reflect the selected sorting order, allowing the user to browse their books more conveniently. |
| Priority | **Medium** |

## Use-case: Delete a book

| Use case Name | Delete a book. |
| --- | --- |
| Brief description | The use case describes how a normal user deletes one of their books. |
| Actors | Normal user. |
| Basic Flow | 1. User chooses the book they want to delete. 2. User chooses the option “Delete”. 3. User confirms their choice. 4. System update the book list. |
| Alternative Flows | **Alternative flow 1: Book cannot be found or user cannot access the book.**   1. From #4 of the basic flow, the system displays a notification “Access denied”. 2. Halt.   **Alternative flow 2: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and the library remains on the list. |
| Pre-conditions | * User is logged in. * User navigates to the site ***/books***. |
| Post-conditions | User successfully deletes their book. |
| Priority | **High** |

## Tags

## Use-case: Create a tag

| Use case Name | Create a tag. |
| --- | --- |
| Brief description | Allow user to create a tag so that they can manage their books systematically. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to the tag feature. 2. System displays the list of tags created by the user accompanied by an “Add” icon. 3. User chooses “Add”. 4. System allows user to create a new tag: enter tag name. 5. User chooses “Save”. 6. System verifies for duplication and updates the tag name. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box (without saving).**   1. From the step the user cancels the action, the system halts and the book remains untagged.   **Alternative flow 2: Tag name duplicated.**   1. From #6 of the basic flow, the system prompts the user that “Tag name already exists”. 2. Continue from #4 of the basic flow.   **Alternative flow 3: Maximum number of tags reached (20 tags).**   1. From #4 of the basic flow, the system prompts the user that “Maximum number of tags”. 2. Halt. |
| Pre-conditions | User is logged in. |
| Post-conditions | The tag is successfully added to a book. |
| Priority | **High** |

## Use-case: Edit a tag name

| Use case Name | Edit a tag name. |
| --- | --- |
| Brief description | Allows user to edit the name of the tag they created. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to the tag feature. 2. System displays the list of tags created by the user accompanied by an “Edit” and “Delete” icon for each tag. 3. User chooses “Edit”. 4. System allows user to change the tag name. 5. User chooses “Save”. 6. System updates the tag name. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box (without saving).**   1. From the step the user cancels the action, the system halts and the tag name remains unchanged. |
| Pre-conditions | * User is logged in. * The tag exists. |
| Post-conditions | Tag name successfully changed. |
| Priority | **Medium** |

## Use-case: Delete a tag

| Use case Name | Delete a tag. |
| --- | --- |
| Brief description | Allows a user to delete the tag they created. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to the tag feature. 2. System displays the list of tags created by the user accompanied by an “Edit” and “Delete” icon for each tag. 3. User chooses “Delete”. 4. System prompts the user to confirm their deletion. 5. User chooses “Confirm”. 6. System delete the tag. |
| Alternative Flows | **Alternative flow 1: The user doesn’t click “Confirm”.**   1. From #6 of the basic flow, system halts the process.   **Alternative flow 2: User cancels the action by clicking outside the options menu box.**   1. From the step user cancels the action, the system halts the process. No tags were deleted. |
| Pre-conditions | * User is logged in. * The tag exists. |
| Post-conditions | The tag is unlinked from the book and removed from the database. |
| Priority | **Medium** |

## Use-case: Set tags for a book

| Use case Name | Set tags for a book. |
| --- | --- |
| Brief description | Allows a user to assign one or more predefined tags to a book, helping to organize and classify the book in their library. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to their library using the nav bar. 2. User selects a book by clicking on its cover or title to access the book’s settings. 3. System opens the book details page showing the book’s metadata, including current categories. 4. User locates the “Set Tags” option within the details page and clicks on it. 5. System displays a dropdown list or checkboxes with predefined tags, such as “Fiction,” “Non-fiction,” “Science Fiction,” “Biography,” etc. 6. User selects one or more tags by clicking on the corresponding checkboxes or dropdown options. 7. User confirms the selection by clicking the “Save” button. 8. System saves the selected tags to the database and associates them with the book. 9. System displays a confirmation message, such as “Tags updated successfully,” and updates the book details to reflect the new tags. |
| Alternative Flows | **Alternative flow 1: User cancels the action before clicking “Save”.**   1. The system does not save any category changes, and the book’s existing tags remain unchanged. |
| Pre-conditions | * User is logged in. * User has at least one book uploaded in their library. |
| Post-conditions | The selected tags are saved and associated with the book. |
| Priority | **High** |

## Library

## Use-case: Create a library

| Use case Name | Create a library of books. |
| --- | --- |
| Brief description | A normal user can create a library to organize their books. |
| Actors | Normal user. |
| Basic Flow | 1. User chooses “Create library”. 2. System displays a form for creating a new library, including:    1. Library name: a *mandatory* field.    2. Visibility (public/private): a *mandatory* option include "Public" (visible to friends or all users) or "Private" (visible only to the user).    3. Downloadability: a *mandatory* option to allow or restrict other users from downloading the books in the library.    4. Books: an *optional* option of books to add to the library (a book can belong to multiple libraries). 3. User fills in all the fields and submits the form. 4. System verifies the input. 5. When verified, the system creates the library and displays a “New library created” message. |
| Alternative Flows | **Alternative flow 1: User (magically) access to a book they do not own.**   1. From #6 of the basic flow, the application shows a message “Access denied.” 2. Halt.   **Alternative flow 2: Required field(s) is/are empty.**   1. From #4 of the basic flow, the system prompts user to fill in mandatory fields. 2. Continue from #2 of the main flow. |
| Pre-conditions | * User navigates the main page ***/books***. * User is logged in and has access to the "Libraries" section. |
| Post-conditions | Library is created, saved with the specified name, visibility, and download permissions, and is shown in the user's libraries list. |
| Priority | **High** |

## Use-case: See the list of created libraries

| Use case Name | See the list of created libraries. |
| --- | --- |
| Brief description | Enables a user to view all libraries they have created. |
| Actors | Normal user. |
| Basic Flow | 1. The user navigates to the "Libraries" section. 2. The system displays a list of all libraries created by the user; or displays nothing if no libraries are available. |
| Alternative Flows | None. |
| Pre-conditions | The user is logged in and has accessed the "Collections" section. |
| Post-conditions | The user sees a list of their libraries or nothing if none are created. |
| Priority | **High** |

## Use-case: Delete a library

| Use case Name | Delete a library. |
| --- | --- |
| Brief description | Allows a user to delete a library they have previously created. |
| Actors | Normal user. |
| Basic Flow | 1. The user selects a library they wish to delete. 2. The system asks for confirmation. 3. The user confirms the deletion. 4. The system removes the library from the database and displays a message: "Library deleted." |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and the library remains on the list. |
| Pre-conditions | The user is logged in and has navigated to their libraries list. |
| Post-conditions | The library is deleted from the user’s list. |
| Priority | **High** |

## Use-case: Edit the library (name)

| Use case Name | Edit library name |
| --- | --- |
| Brief description | Allows a user to edit the name of an existing library. |
| Actors | Normal user. |
| Basic Flow | 1. The user selects a library and chooses to edit the name. 2. The system displays the current name in an editable field. 3. The user enters a new name and confirms. 4. The system saves the new name and updates the library. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and the library remains on the list.   **Alternative flow 2: New name is left blank.**   1. From #4 of the basic flow, the system displays a prompt: "Library name is required." 2. Continue from #2 of the basic flow. |
| Pre-conditions | User is logged in and viewing their libraries. |
| Post-conditions | The library name is updated in the database. |
| Priority | **Medium** |

## Use-case: Edit the library (visibility)

| Use case Name | Edit library visibility |
| --- | --- |
| Brief description | Allows a user to change the visibility setting of a library. |
| Actors | Normal user. |
| Basic Flow | 1. The user selects a library to edit. 2. The system displays the current visibility setting. 3. The user changes the visibility to either "Public" or "Private" and confirms. 4. The system saves the new visibility setting. |
| Alternative Flows | **Alternative flow 1: User cancels the action.**   1. From the step the user cancels the action, the system halts and the library remains on the list. |
| Pre-conditions | User is logged in and viewing their libraries. |
| Post-conditions | The visibility setting of the library is updated in the database. |
| Priority | **High** |

## Use-case: Edit the library (add books)

| Use case Name | Add a book to the library. |
| --- | --- |
| Brief description | Allows a user to add books to an existing library. |
| Actors | Normal user. |
| Basic Flow | 1. User selects a library to edit. 2. The system displays the books currently in the library. 3. User selects books from the list of uploaded books to add to the library. 4. User confirms their changes. 5. The system updates the library and displays a confirmation message. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and the library remains on the list.   **Alternative flow 2: User (magically) tries to add books they do not own or books which do not exist in the database.**   1. From #5 of the basic flow, the system displays "Access Denied.". |
| Pre-conditions | * User is logged in. * User owns the library. |
| Post-conditions | The library’s list of books is updated in the database. |
| Priority | **High** |

## Use-case: Edit the library (remove a book)

| Use case Name | Remove a book from the library. |
| --- | --- |
| Brief description | This use case describes how a normal user can remove one of the books from their library (not remove the book from the storage; the book just no longer relates to the library). |
| Actors | Normal user. |
| Basic Flow | 1. User selects a library to edit. 2. The system displays the books currently in the library. 3. User chooses “Remove books”. 4. User chooses books to be removed from the library and confirms their changes. 5. The system updates the library and displays a confirmation message. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and the library remains on the list.   **Alternative flow 2: User (magically) tries to remove books they do not own, books which do not exist in the database, or books which are not in the library.**   1. From #5 of the basic flow, the system displays "Access Denied.". |
| Pre-conditions | * User is logged in. * User owns the library. |
| Post-conditions | The library’s list of books is updated in the database. |
| Priority | **High** |

## Reading features

## Use-case: Read book

| Use case Name | Read a book. |
| --- | --- |
| Brief description | Allows a user to open a book and read it within the application. |
| Actors | Normal user. |
| Basic Flow | 1. User selects the book from their library list on the main interface. 2. System retrieves the book content and the user’s last saved reading position. 3. System displays the book in the reading interface at the last read position or the first page if no saved position exists (user’s first time access to the book). 4. User navigates through the book using the "Next" and "Previous" buttons, page scroll, clicking right or left of the pages, or by selecting a chapter from the table of contents. 5. User finishes reading and closes the book. 6. System saves the lastest read position. |
| Alternative Flows | **Alternative flow 1: System cannot retrieve the book’s content.**   1. From #3 of the basic flow, the browser keeps loading until time is out. 2. Continue from #1 of the basic flow.   **Alternative flow 2: System cannot retrieve the book’s related features (all the highlights, notes, bookmarks).**   1. From #3 of the basic flow, the book’s content will be displayed as usual but the related features will not. 2. Continue from #4 of the basic flow. |
| Pre-conditions | * User is logged in. * User has access to the selected book. * User navigates to the main interface of the user. |
| Post-conditions | The last read position is saved in the database with any in-progress notes, highlights, or bookmarks are also saved. |
| Priority | **High** |

## Use-case: Add highlight

| Use case Name | Add a highlight. |
| --- | --- |
| Brief description | Allows a user to highlight a specific text passage within a book. |
| Actors | Normal user. |
| Basic Flow | 1. User selects text within the opened book by clicking and dragging. 2. User right-clicks on the selected text. 3. System displays a menu with options like “Highlight,” “Add Note,” and “Copy”. 4. User hovers over the “Highlight” upon doing so a smaller color options menu beside it:    1. User chooses 1 of 4 color options: “Red”, “Yellow”, “Blue", “Green”.    2. If the user does not click on a color and just clicks on the “Highlight” button, the default color is applied (“Yellow”) if there has not been any previous highlight color chosen, else the previous chosen color will be chosen. 5. System applies the highlight with the chosen or the default color to the selected text, visually marking it. 6. User presses Ctrl + S on the keyboard to save all the changes they made. 7. System saves the highlights in the database. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and does not save any highlights, the text remains unmarked.   **Alternative flow 2: User stops reading the book without saving the changes.**   1. From the step the user ends the reading process, the system asks the user whether they want to save their changes. 2. User chooses:    1. *Yes*: The system saves all the changes. Continue from step #7 of the basic flow.    2. *No*: The system discards all the changes. Halt. |
| Pre-conditions | * User is logged in. * User is reading the book. * The book belongs to the user. * User is allowed to select texts in the file. |
| Post-conditions | The highlight is saved. |
| Priority | **High** |

## Use-case: Remove highlight

| Use case Name | Remove a highlight. |
| --- | --- |
| Brief description | Allows a user to remove a highlight from the text, restoring it to normal text. |
| Actors | Normal user. |
| Basic Flow | 1. User clicks on highlighted text within the book. 2. System displays a context menu with options such as “Remove Highlight”, “Add Note”, “Copy”. 3. User selects “Remove Highlight”. 4. System removes the highlight from the text. 5. User presses Ctrl + S on the keyboard to save all the changes they made. 6. System updates the database, deleting the highlight record for this selection. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and the highlight area remains unchanged.   **Alternative flow 2: User stops reading the book without saving the changes.**   1. From step #5 of the basic flow, all the removed highlighted texts in the reading session that were stored locally will remain unchanged.   **Alternative flow 3: User stops reading the book without saving the changes.**   1. From the step the user ends the reading process, the system asks the user whether they want to save their changes. 2. User chooses:    1. *Yes*: The system saves all the changes. Continue from step #6 of the basic flow.    2. *No*: The system discards all the changes. Halt. |
| Pre-conditions | * User is logged in. * User has previously highlighted text in the book. * User is reading a book. |
| Post-conditions | The highlight is unlinked and removed from the database. |
| Priority | **High** |

## Use-case: Add note

| Use case Name | Add a note. |
| --- | --- |
| Brief description | Allows a user to attach a personal note to a specific passage or section in the book. |
| Actors | Normal user. |
| Basic Flow | 1. User selects text within the opened book by clicking and dragging. 2. User right-clicks on the selected text. 3. System displays a menu with options for the selected text. 4. User selects “Add Note” which opens a text input box. 5. User enters note text in the box and clicks “Save”. 6. System displays the “note” icon at the end of the chosen text. 7. User presses Ctrl + S on the keyboard to save all the changes they made. 8. System saves the note in the database. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step user cancels the action, the system halts the process. It does not save the note, and no changes are made to the book.   **Alternative flow 2: User stops reading the book without saving the changes.**   1. From the step the user ends the reading process, the system asks the user whether they want to save their changes. 2. User chooses:    1. *Yes*: The system saves all the changes. Continue from step #8 of the basic flow.    2. *No*: The system discards all the changes. Halt. |
| Pre-conditions | * User is logged in. * User is reading a book where notes are permitted. |
| Post-conditions | The note is saved and can be accessed at the specified location in the book. |
| Priority | **Medium** |

## Use-case: Edit note

| Use case Name | Add a note. |
| --- | --- |
| Brief description | Allows a user to edit a note they created earlier. |
| Actors | Normal user. |
| Basic Flow | 1. User double-clicks on the note icon of the note they desire to edit. 2. System displays a modal including the note’s content and an “Edit” icon. 3. User selects the “Edit” icon. 4. System allows the user to edit the note’s content. 5. User edits the note and saves it. 6. System saves the edited note. 7. User presses Ctrl + S on the keyboard to save all the changes they made. 8. System saves the edited note to the database. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the modal.**   1. From the step user cancels the action, the system halts the process. It does not save the note, and no changes are made to the book.   **Alternative flow 2:** **User stops reading the book without saving the changes.**   1. From the step the user ends the reading process, the system asks the user whether they want to save their changes. 2. User chooses:    1. *Yes*: The system saves all the changes. Continue from step #8 of the basic flow.    2. *No*: The system discards all the changes. Halt. |
| Pre-conditions | * User is logged in. * User is reading a book where notes are permitted. |
| Post-conditions | The note is saved and can be accessed at the specified location in the book. |
| Priority | **Medium** |

## Use-case: Remove note

| Use case Name | Remove a note. |
| --- | --- |
| Brief description | Allows a user to delete a previously saved note. |
| Actors | Normal user. |
| Basic Flow | 1. User double-clicks on the note icon of the note they desire to edit. 2. System displays a modal including the note’s content and a “Delete Note” icon. 3. User selects “Delete Note”. 4. System prompts for confirmation “Delete note?”. 5. User selects the options “YES” within the prompt box. 6. User presses Ctrl + S on the keyboard to save all the changes they made. 7. The system deletes the note from the database. |
| Alternative Flows | **Alternative flow 1: User cancels the action by clicking outside the options menu box.**   1. From the step user cancels the action, the system halts the process. The note remains saved and attached to the text. No changes were made in this case.   **Alternative flow 2: User stops reading the book without saving the changes.**   1. From the step the user ends the reading process, the system asks the user whether they want to save their changes. 2. User chooses:    1. *Yes*: The system saves all the changes. Continue from step #8 of the basic flow.    2. *No*: The system discards all the changes. Halt.   **Alternative flow 3: User chooses “NO” at step #5 of the basic flow.**   1. From #6 of the basic flow, the system halts the process. No changes were made. |
| Pre-conditions | * User is logged in. * User has previously added a note to the text. * User is reading a book. |
| Post-conditions | The note is unlinked from the book and removed from the database. |
| Priority | **Medium** |

## Use-case: Add bookmark

| Use case Name | Add a bookmark. |
| --- | --- |
| Brief description | Allows a user to bookmark a specific page for quick access in the future. |
| Actors | Normal user. |
| Basic Flow | 1. User selects the “Bookmark” → “Create bookmark” option from the interface menu. 2. System displays the modal so that the user can input the page number they want to bookmark. 3. System saves the bookmark locally. 4. User presses Ctrl + S on the keyboard to save all the changes they made. 5. System saves the bookmark to the database. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the options menu box.**   1. From the step user cancels the action, the system halts the process. No bookmarks were created.   **Alternative flow 2: User stops reading the book without saving the changes.**   1. From the step the user ends the reading process, the system asks the user whether they want to save their changes. 2. User chooses:    1. *Yes*: The system saves all the changes. Continue from step #8 of the basic flow.    2. *No*: The system discards all the changes. Halt. |
| Pre-conditions | * User is logged in. * User is reading a book they own or have access to. |
| Post-conditions | The bookmark is saved in the database, linked to the user’s profile. |
| Priority | **High** |

## Use-case: Remove bookmark

| Use case Name | Remove a bookmark. |
| --- | --- |
| Brief description | Allows a user to delete a saved bookmark. |
| Actors | Normal user. |
| Basic Flow | 1. User selects the “Bookmark” option to navigate to the list of bookmarks created. 2. System displays “Remove” icons on the corresponding bookmark. 3. User chooses the “Remove” icon on the bookmark. 4. System prompts the user to confirm their change. 5. User confirms. 6. System removes the bookmark locally. 7. User presses Ctrl + S on the keyboard to save all the changes they made. 8. The system deletes the bookmark from the database. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts the process and the bookmark remains saved.   **Alternative flow 2: User does not confirm the change.**   1. From #6 of the basic flow, system halts the process.   **Alternative flow 3: User stops reading the book without saving the changes.**   1. From the step the user ends the reading process, the system asks the user whether they want to save their changes. 2. User chooses:    1. *Yes*: The system saves all the changes. Continue from step #8 of the basic flow.    2. *No*: The system discards all the changes. Halt. |
| Pre-conditions | * User is logged in. * User has a bookmark saved in the book. * User is reading a book. |
| Post-conditions | The bookmark is removed from the database. |
| Priority | **High** |

## Use-case: Change font

| Use case Name | Change font or adjust font size. |
| --- | --- |
| Brief description | Allows a user to change the font in the book reader interface for a personalized reading experience. |
| Actors | Normal user. |
| Basic Flow | 1. User selects “View” in the reader interface. 2. System displays the view panel including “Font”. 3. User chooses the font they like (initial font will be chosen later). 4. System applies the font on the screen. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts the process and font settings remain unchanged. |
| Pre-conditions | * User is logged in. * User is reading the book. |
| Post-conditions | The font will be applied to other reading processes. |
| Priority | **High** |

## Use-case: Change font size

| Use case Name | Adjust font size. |
| --- | --- |
| Brief description | Allows a user to change the font size in the book reader interface for a personalized reading experience. |
| Actors | Normal user. |
| Basic Flow | 1. User selects “View” in the reader interface. 2. System displays the view panel including “Font size”. 3. User chooses the font size they like (initial font size will be chosen later). 4. System applies the font size on the screen. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts the process and font settings remain unchanged. |
| Pre-conditions | * User is logged in. * User is reading the book. |
| Post-conditions | The font size will be applied to other reading processes. |
| Priority | **High** |

## Use-case: Adjust line spacing

| Use case Name | Adjust line spacing. |
| --- | --- |
| Brief description | Allows a user to change the line spacing in the book reader interface for a personalized reading experience. |
| Actors | Normal user. |
| Basic Flow | 1. User selects “View” in the reader interface. 2. System displays the view panel including “Line spacing”. 3. User chooses the font size they like (initial line spacing will be chosen later). 4. System applies the line spacing on the screen. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts the process and font settings remain unchanged. |
| Pre-conditions | * User is logged in. * User is reading the book. |
| Post-conditions | The line spacing will be applied to other reading processes. |
| Priority | **Medium** |

## Use-case: Change reading theme

| Use case Name | Change reading theme. |
| --- | --- |
| Brief description | Allows a user to change the theme (background color and text color) in the book reader interface for a personalized reading experience. |
| Actors | Normal user. |
| Basic Flow | 1. User selects “View” in the reader interface. 2. System displays the view panel including “Theme”. 3. User chooses the theme they like (initial theme will be chosen later). 4. System applies the theme to the screen. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts the process and font settings remain unchanged. |
| Pre-conditions | * User is logged in. * User is reading the book. |
| Post-conditions | The theme will be applied to other reading processes. |
| Priority | **High** |

## Self analytics

## Use-case: View reading analysis of a book

| Use case Name | View reading analysis of a book. |
| --- | --- |
| Brief description | Allows the user to see the reading analysis of all the books they uploaded. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to anywhere they can see their books. 2. System displays the reading analysis on the book:    1. Reading progress.    2. Reading speed.    3. Page count. |
| Alternative Flows | None. |
| Pre-conditions | User is logged in and has permission to view the specified book. |
| Post-conditions | The user sees their reading analysis in percentage. |
| Priority | **High** |

## Use-case: Track time spent

| Use case Name | Track time spent. |
| --- | --- |
| Brief description | Allows the user to track how much time they spent on the site to read. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to the ‘social’ user profile. 2. System displays time spent on reading. |
| Alternative Flows | None. |
| Pre-conditions | User is logged in. |
| Post-conditions | User can see the amount of time they spent on reading. |
| Priority | **Medium** |

## Use-case: Set/Edit a reading goal

| Use case Name | Set/Edit a reading goal. |
| --- | --- |
| Brief description | Allows user to set or edit a reading goal (set by the number of pages to be completed). |
| Actors | Normal user. |
| Basic Flow | 1. User choose “Reading goal” on the left hand side menu on the main page. 2. System displays 2 options: “Set reading goal” and “Delete reading goal”. 3. User chooses “Set reading goal”. 4. System displays the current goal if it exists, or an empty input. 5. User inputs the number of pages to read and chooses the “Set goal” option. 6. System saves and displays the new reading goal. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the “Set goal” menu box.**   1. From the step the user cancels the action, the system halts the process and saves/updates nothing. |
| Pre-conditions | * User is logged in. * User navigates to the main page ***/books***. |
| Post-conditions | A reading goal with a specific number of pages to be completed is set for the selected book. |
| Priority | **High** |

## Use-case: Delete a reading goal

| Use case Name | Delete a reading goal. |
| --- | --- |
| Brief description | Allows user to delete a reading goal created for a book. |
| Actors | Normal user. |
| Basic Flow | 1. User choose “Reading goal” on the left hand side menu on the main page. 2. System displays 2 options: “Set reading goal” and “Delete reading goal”. 3. User chooses “Delete reading goal”. 4. System removes the earlier goal. |
| Alternative Flows | **Alternative flow 1:** **User cancels the action by clicking outside the menu box.**   1. From the step the user cancels the action, the system halts the process and the goal remains set.   **Alternative flow 2: There is no goal set.**   1. From #4 of basic flow, nothing happens. |
| Pre-conditions | * User is logged in. * User navigates to the main page ***/books***. |
| Post-conditions | User successfully deletes the goal. |
| Priority | **High** |

## Social

## Friend request

## Use-case: Send a friend request to another normal user

| Use case Name | Send a friend request to another normal user. |
| --- | --- |
| Brief description | Allows a normal user to send a friend request to another normal user within the application. |
| Actors | Normal user 1: the user who sent the friend request.  Normal user 2 (passive): the user who receives the friend request. |
| Basic Flow | 1. User 1 navigates to User 2’s profile via the searching feature. 2. User 1 selects the “Add friend” option. 3. The system sends a friend request to User 2, and flips the previous “Add friend” option to ”Cancel request”. |
| Alternative Flows | **Alternative flow 1: User sends a friend request more than once.**   1. The system behaves as if nothing happens. |
| Pre-conditions | User 1 must be logged in and have located User 2’s profile. |
| Post-conditions | User 2 receives a friend request notification. |
| Priority | **High** |

## Use-case: Cancel a friend request to another normal user

| Use case Name | Cancel a friend request to another normal user. |
| --- | --- |
| Brief description | Allows a normal user to cancel a friend request they sent to another normal user within the application. |
| Actors | Normal user 1: the user who cancels the friend request.  Normal user 2 (passive): the user who received the friend request. |
| Basic Flow | 1. User 1 navigates to User 2’s profile via the searching feature. 2. User 1 selects the “Cancel request” option. 3. The system sends a friend request to User 2/removes the friend request from user 2, and flips the previous “Cancel request” option to “Add friend”. |
| Alternative Flows | **Alternative flow 1: User cancels a friend request more than once.**   1. The system behaves as if nothing happens.   **Alternative flow 2: User cancels an accepted friend request.**   1. After #3 of the basic flow, user 1 and user 2 unfriend each other. |
| Pre-conditions | User 1 must be logged in and have located User 2’s profile. |
| Post-conditions | User 2 will not see the canceled friend request. |
| Priority | **Medium** |

## Use-case: Accept a friend request from another normal user

| Use case Name | Accept a friend request from another normal user. |
| --- | --- |
| Brief description | Allows a user to accept a friend request from another user to establish a connection between the two. |
| Actors | Normal user 1: the user receiving and accepting/declining the friend request.  Normal user 2 (passive): the user who sent the friend request. |
| Basic Flow | 1. User 1 navigates to the “Friend requests” feature to see the list of friend requests they received. 2. User 1 chooses “Accept”. 3. The system adds both users to each other’s friend lists and notifies both users of the connection. |
| Alternative Flows | **Alternative flow 1: User 1 accepts a canceled friend request.**  (this happens when user 1 accepts the friend request before user 2 cancels it)   1. After #3 from the basic flow, user 1 and user 2 unfriend each other.   **Alternative flow 2: User accepts a friend request more than once.**   1. The system behaves as if nothing happens. |
| Pre-conditions | * User 1 is logged in. * User 1 has a pending friend request from User 2. |
| Post-conditions | User 1 and User 2 are friends. |
| Priority | **High** |

## Use-case: Decline a friend request from another normal user

| Use case Name | Decline a friend request from another normal user. |
| --- | --- |
| Brief description | Allows a user to decline a friend request they received from another user. |
| Actors | Normal user 1: the user receiving and accepting/declining the friend request.  Normal user 2 (passive): the user who sent the friend request. |
| Basic Flow | 1. User 1 navigates to the “Friend requests” feature to see the list of friend requests they received. 2. User 1 chooses “Decline”. 3. The system removes the friend request. |
| Alternative Flows | **Alternative flow 1: User 1 declines a canceled friend request.**   1. The system behaves as if nothing happens.   **Alternative flow 2: User accepts a friend request more than once.**   1. The system behaves as if nothing happens. |
| Pre-conditions | * User 1 is logged in. * User 1 has a pending friend request from User 2. |
| Post-conditions | * The friend request will no longer appear on the side of user 1. * When user 1 navigates to user 2’s profile, they see the option “Add friend” (as if no friend request was sent). |
| Priority | **High** |

## Relationship management

## Use-case: Remove a friend

| Use case Name | Remove a friend. |
| --- | --- |
| Brief description | Allows a user to remove an existing friend, breaking the connection between them. |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to their friends list or the friend’s user profile. 2. User selects the option to remove a friend. 3. The system asks user 1 to confirm the removal. 4. User confirms their choice. 5. The system updates both users’ friends lists. |
| Alternative Flows | **Alternative flow 1: User 1 removes a friend more than once.**   1. The system behaves as if nothing happens. |
| Pre-conditions | * The user is logged in. * The friend to be removed is user 1’s friend. |
| Post-conditions | The selected friend is removed from the user's friend list. |
| Priority | **High** |

## Use-case: Normal user blocks another normal user

| Use case Name | Block another normal user(s). |
| --- | --- |
| Brief description | This use case describes how a normal user can block other normal user(s). |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to the user profile of the user to be blocked. 2. User chooses the “Block” option. 3. System displays a prompt box asking confirmation “Block this user” where the user can either choose “YES” or “CANCEL”. 4. The user chooses “YES”, the System returns a pop-up prompt “Blocked Users”. 5. The system removes each of them from the other friend list. |
| Alternative Flows | **Alternative flow 1: User cancels the action.**   1. From the step the user cancels the action, the system halts and the library remains on the list.   **Alternative flow 2: Users are not friends.**   1. From step #5 of the basic flow, the system halts the process. |
| Pre-conditions | The user is logged in and can access the other user’s profile. |
| Post-conditions | The blocked user(s) not being allowed to view the profile / library / books of the user(s) who block them as well as auto-unfriending the blocked user(s) if the users were of friends status beforehand. |
| Priority | **Medium** |

## Use-case: Normal user unblocks another normal user

| Use case Name | Block another normal user(s). |
| --- | --- |
| Brief description | This use case describes how a normal user can block other normal user(s). |
| Actors | Normal user. |
| Basic Flow | 1. User navigates to the user profile of the user to be blocked. 2. User chooses the “Block” option. 3. System displays a prompt box asking confirmation “Block this user” where the user can either choose “YES” or “CANCEL”. 4. The user chooses “YES”, the System returns a pop-up prompt “Blocked Users”. 5. The system removes each of them from the other friend list. |
| Alternative Flows | **Alternative flow 1: User cancels the action.**   1. From the step the user cancels the action, the system halts and the library remains on the list.   **Alternative flow 2: Users are not friends.**   1. From step #5 of the basic flow, the system halts the process. |
| Pre-conditions | The user is logged in and can access the other user’s profile. |
| Post-conditions | The blocked user(s) not being allowed to view the profile / library / books of the user(s) who block them as well as auto-unfriending the blocked user(s) if the users were of friends status beforehand. |
| Priority | **Medium** |

## Use cases for admin users

## Use-case: Register a new admin account

| Use case Name | Register a new admin account. |
| --- | --- |
| Brief description | The use case describes the flow of creating a new admin account inside an existing admin account. |
| Actors | Admin user. |
| Basic Flow | 1. System displays the registration form. 2. Admin enters username, email, password, confirm password (all fields are required). 3. Admin clicks to the “Register” button (submit). 4. System validates the inputs. Constraints are:    1. No empty field.    2. Username and email have not been used (including email address variations using periods and plus signs).    3. Email must be in a valid form: no white spaces, pattern “[example@provider.topdomain](mailto:example@provider.topdomain)”.    4. The email must exist.    5. All fields contain only printable characters.    6. The password must be strong (at least 8 characters including lowercase *a-z*, uppercase letters *A-Z*, digits *0-9*, and some of the following special characters *!#\*+-*).    7. Username shall not contain any whitespaces; only contains a-z, A-Z, 0-9, period(s) “.”, and underscores “\_”.    8. Confirm password and password must be identical. 5. System notifies the admin that the account has been successfully registered. 6. System sends an email to the email address provided by the user. |
| Alternative Flows | **Alternative flow 1: Username or email has been used.**   1. From #5 of the basic flow, the system raises a notification: “Username or email has been used”. 2. Continue from #1 of the basic flow.   **Alternative flow 2: Password is not sufficiently strong.**   1. From #5 of the basic flow, the system raises a notification: “Password is not strong enough”. 2. Continue from #1 of the basic flow.   **Alternative flow 3: Username is invalid**   1. From #5 of the basic flow, the system raises a notification: “Username is invalid”. 2. Continue from #1 of the basic flow.   **Alternative flow 4: Invalid email.**   1. From #5 of the basic flow, the system raises a notification: “Email is invalid”. 2. Continue from #1 of the basic flow.   **Alternative flow 5: Any field is empty.**   1. From #5 of the basic flow, the system raises a notification: “All fields are required”. 2. Continue from #1 of the basic flow.   **Alternative flow 6: Any fields contain non-printable characters.**   1. From #5 of the basic flow, the system raises a notification: “Must use printable Latin character”. 2. Continue from #1 of the basic flow.   **Alternative flow 7: Confirm password and password are not identical.**   1. From #5 of the basic flow, the system raises a notification: “Passwords are not matched”. 2. Continue from #1 of the basic flow. |
| Pre-conditions | * The user who has the ability to create a new admin account must also be an admin. * The admin user logs in with their admin account, then navigates to the “Create admin account” section. * Initially, there must be an admin account named “admin” in the system. This initial admin will be used to create another admin account. |
| Post-conditions | User successfully creates a new admin account. |
| Priority | **High** |

## Use-case: Log in to an admin account

| Use case Name | Log in to an admin account. |
| --- | --- |
| Brief description | The use case describes the flow of logging in to an admin account. |
| Actors | Admin user. |
| Basic Flow | 1. System displays the login form. 2. Admin fills in all the fields:  * Username/Email. * Password.  1. Admin submits the login form. 2. The system informs that the admin has successfully signed in and redirects the admin to the main page from which he/she can start their journey. |
| Alternative Flows | **Alternative flow 1: Incorrect username/email or password.**   1. From #4 of the basic flow, the system raises a notification: “Username or password is incorrect”. 2. Continue from #1 of the basic flow. |
| Pre-conditions | User navigates to the admin login page **/admin-login** of the website. |
| Post-conditions | Admin successfully logs in to their account if it exists. |
| Priority | **High** |

## Use-case: Delete a normal user’s account

| Use case Name | Deletes a normal user’s account. |
| --- | --- |
| Brief description | The use case describes how an admin can delete the account of a normal user. |
| Actors | Admin user. |
| Basic Flow | 1. Admin chooses “Delete account” on the user’s profile. 2. System asks the user to confirm they want to delete the account by entering “CONFIRM DELETE <username>” and clicking the button “Confirm”. 3. System notifies the user that they has removed the account. 4. System sends an email to the user’s email address that the account linked to the email address has been removed. 5. Log out the user. |
| Alternative Flows | **Alternative flow 1: Admin inputs the confirm phrase incorrectly.**   1. From #4 of the basic flow, the system displays a notification “Please confirm again”. 2. Continue from #2 of the basic flow.   **Alternative flow 2: User cancels the action.**   1. From the step the user cancels the action, the system halts and the library remains on the list. |
| Pre-conditions | The admin must have logged in.  The admin must navigate to the chosen user’s profile. |
| Post-conditions | Admin successfully removes the chosen normal user’s account. |
| Priority | **High** |

## Use-case: See all other user’s books/libraries

| Use case Name | See all other user’s books. |
| --- | --- |
| Brief description | The use case describes how an admin can see all the books and libraries of a normal user. |
| Actors | Admin user. |
| Basic Flow | 1. User navigates to the normal user’s profile. 2. System displays all the books and libraries the user has, even those that are private. |
| Alternative Flows | **Alternative flow 1: There is no book or library.**   1. From #2 of the basic flow, the system displays nothing. |
| Pre-conditions | User is logged in. |
| Post-conditions | User successfully see all the books and libraries of the normal user. |
| Priority | **High** |

## Use-case: Delete a book by admin

| Use case Name | Delete a book by admin. |
| --- | --- |
| Brief description | The use case describes how an admin deletes one of a normal user’s books. |
| Actors | Admin user. |
| Basic Flow | 1. User navigates to the book list of the normal user. 2. User chooses the book they want to delete. 3. User chooses the option “Delete”. 4. User confirms their choice. 5. System update the book list. |
| Alternative Flows | **Alternative flow 1: Book cannot be found.**   1. From #4 of the basic flow, the system displays a notification “Access denied”. 2. Halt.   **Alternative flow 2: User cancels the action by clicking outside the options menu box.**   1. From the step the user cancels the action, the system halts and the library remains on the list. |
| Pre-conditions | User is logged in. |
| Post-conditions | User successfully deletes the book. |
| Priority | **High** |

## Use-case: Overview analytics

| Use case Name | Overview analytics. |
| --- | --- |
| Brief description | Allows the admin user to see overall statistics: number of real-time active users, average reading time of all the users. |
| Actors | Admin user. |
| Basic Flow | 1. The admin user navigates to the analytics or dashboard section intended for administrative overview. 2. The system retrieves and displays relevant real-time and historical analytics, which includes:    1. Number of active users currently online.    2. Other key reading metrics, such as average time spent per session, total books uploaded. 3. The admin user can view, filter, and analyze the data as needed to monitor platform engagement. |
| Alternative Flows | None. |
| Pre-conditions | The admin user is logged in with the necessary permissions to view analytics data. |
| Post-conditions | The system provides an analytics overview, enabling the admin user to monitor key metrics related to user activity and reading engagement on the platform. |
| Priority | **High** |

## Use-case: Monitor reading progress

| Use case Name | Reading progress monitoring. |
| --- | --- |
| Brief description | Allow the admin to monitor user’s reading progress on each book that they had uploaded on the platform (measured in percentage). |
| Actors | Admin user. |
| Basic Flow | 1. Admin navigates to a normal user’s profile. 2. System displays the list of books that the normal user uploaded, and reading progress of each book is displayed on the corresponding book. |
| Alternative Flows | None. |
| Pre-conditions | The admin user is logged in with the necessary permissions to view all user's reading progress data. |
| Post-conditions | Admin can see the reading progress of each of all the books of every user. |
| Priority | **High** |

## Use-case: Monitor user session

| Use case Name | User session monitoring. |
| --- | --- |
| Brief description | Allows the admin to monitor the time a normal user spent on reading in the application (measured in hh:mm). |
| Actors | Admin user. |
| Basic Flow | 1. Admin navigates to a normal user’s profile. 2. System displays the user’s reading sessions AKA the total time spent reading on the platform. 3. The admin user can view and analyze this data to understand reading patterns, track engagement, or monitor specific users’ reading habits. |
| Alternative Flows | None. |
| Pre-conditions | The admin user is logged in with the necessary permissions to view all user's session data. |
| Post-conditions | Admin can see the time the user spent on reading. |
| Priority | **High** |

## 