2DV609

Växjö Volunteers
Requirements Specification

Author 1: Atakan Coban
Author 2: Albert Henmyr
Author 3: Alija Levic
Author 4: Kateryna Melnyk
Author 5: Richard Oelschlager

Content

1. Introduction

There are currently no mobile applications available that allow users to volunteer or seek assistance in the Växjö city region. Nowadays, volunteering/activism efforts are typically displayed on official charity and volunteering organizations' websites as worldwide or wide-area campaigns, distributed throughout subject groups and forums in various social networks and other online sources, or encompass the whole Swedish territory. This document will establish, specify and explain the requirements for a mobile application that aims at solving the problem aforementioned.

2. System-wide Requirements

2.1. System Stakeholders

Event organizers: The application allows organizers to find volunteers at a much more convenient and efficient rate.

Event volunteers (Users of the application): The application allows volunteers to find events to attend at a much more convenient and efficient rate.

Development team: The ongoing development of the application will be held responsible by this team and its success will determine the credibility of them as well.

Server team: The accessibility, integrity and security along with many others will be held responsible by this team.

Moderators (Forum moderators): The social order is required to be constantly maintained by this team.

Administrators (Technical administrator, e.g. DB): They will be entrusted with impactful privileges and themselves entrust upon others more privileges.

Managers (Owner and executives): These stakeholders are responsible for anything below their chain of command in terms of the development and maintenance of all services revolving around the application.

App store platform owners: Interest in the application will bring interest to the app store platforms as they are the bridge between the user and their access to the application.

External dependency maintainers: The use of dependency by successful parties increases the credibility and responsibility of maintenance for the dependencies and their maintainers.

2.2. Functional Requirements

BKN-001: The system must allow a user to register and authenticate through Google.
Rationale: Having a well established and trusted registration method will by extension increase trust in the security of this application. Furthermore, outsourcing the authentication process lowers database integrity requirements.
Related: **USR-001**

BKN-002: The system must store data about users and posts and not lose it if it reboots.
Rationale: The system must not rely on a computer being perfectly stable and available.

BKN-003: The application must run on Android and iOS. All USR tests must pass on both platforms.
Rationale: As a mobile application, it should work with these two most popular operating systems.

BKN-004: The system should allow optional notifications on the client device. These are no notifications, notifications on all new posts, and notifications on new posts of the specific category.
Rationale: So that very eager users don't need to open the app to look for news.

BKN-007: Available post categories are: Animals, Nature & Environment, Refugees & Immigrants, People with disabilities, Day-to-day help.
Rationale: These are the most suitable categories of volunteer help we could think of.

MOD-001: A new post or changes to an existing post must successfully pass through an automated moderation filter to be approved for public view.
Rationale: The app relies on formal requests, not casual chatting. Unserious and irrelevant posts should not make it through.
Related: **MOD-005**, **MOD-007**, **MOD-008**, **MOD-009**
Constrained by: **MOD-010**

MOD-002: A moderator should be able to delete posts by other users or moderators from public view.
Rationale: Moderators are humans and not infallible. If something inappropriate slips through, they need to be able to remove it.
Related: **MOD-003**, **MOD-005**, **MOD-007**, **MOD-008**, **MOD-009**

MOD-003: Rejected posts should only be visible to the author and moderators.
Rationale: So that posts that violate the moderation rules of the application are not publicly visible.
Related: **MOD-002**, **MOD-007**, **MOD-008**, **MOD-009**

MOD-005: A user can request manual approval of a post that was rejected by the automated filter.
Rationale: The filter is just an algorithm. A human may notice that there's nothing wrong with the post.
Related: **MOD-001**, **MOD-007**, **MOD-008**, **MOD-009**

MOD-006: Moderators can issue and lift bans on user accounts.
Rationale: To keep out anyone who's misusing the app.

MOD-007: A post can have one of the statuses Accepted, Rejected and Pending Moderation.
Rationale: There are three states for a post. Either the moderation system (automated and/or manual) has deemed it acceptable, unacceptable or simply not yet decided.
Related: **MOD-001**, **MOD-002**, **MOD-005**, **MOD-008**, **MOD-009**

MOD-008: In case of a rejected post or declined changes to an existing post, the system should provide the reason for such a decision to the user.
Rationale: So that users know what they did wrong.
Related: **MOD-001**, **MOD-002**, **MOD-003**, **MOD-005**, **MOD-007**

MOD-009: A post that has been manually rejected must also be manually approved if edited.
Rationale: If it had to be manually rejected, the filter wasn't good enough.
Related: **MOD-001**, **MOD-002**, **MOD-003**, **MOD-005**, **MOD-007**

USR-001: When a user registers, a profile is created and populated with data from Google.
Rationale: A default solution, so that a profile exists.
Related: **BKN-001**

USR-002: A user shall be able to customize their profile with additional information in an "About me" field. It shall be at most 1,000 characters long.
Rationale: Some users wish to let others know more about them. Mentioning what skills you can provide as a volunteer saves you from having to repeat yourself every time you offer to help. A more complete profile is also generally trusted more when seen in comparison to an empty profile with just a name.

USR-003: A user shall be able to list all volunteer requests..
Rationale: The goal is that anyone looking to help can find people in need of help.

USR-004: A user shall be able to filter volunteer requests by category.
Rationale: So that a volunteer can limit their search to things they can actually help with.
Related: **BKN-007**

USR-005: A user shall be able to submit a volunteer request. Its body shall be at most 1,000 characters long.
Rationale: So that people can ask for help.

Related: **USR-007**
Constrained by: **USR-015**

USR-007: A user shall be able to select a category from a list when submitting a volunteer request.
Rationale: To make it easier for volunteers to know if they are able to help.
Related: **USR-005**
Specified by: **USR-017**

USR-008: A user shall be able to list/view their own volunteer requests.
Rationale: So they know what they have posted and the state of their request
Related: **USR-009**, **USR-010**

USR-009: A user shall be able to delete their own volunteer requests.
Rationale: In case they no longer require aid.
Related: **USR-008**, **USR-010**

USR-010: A user shall be able to edit their own volunteer requests. Changes must go through the moderation system.
Rationale: Situation changed or just proofreading.
Related: **USR-008**, **USR-009**

USR-011: The system must allow a user to bookmark other users' posts.
Rationale: So that users can easily keep track of posts they find interesting.
Related: **USR-012**, **USR-013**

USR-012: A user shall be able to bookmark other users' posts.
Rationale: So that users can easily keep track of posts they find interesting.
Related: **USR-011**, **USR-013**

USR-013: A user shall be able to remove bookmarked requests from their bookmarks.
Rationale: In case they are no longer relevant.
Related: **USR-011**, **USR-012**

USR-014: A user shall be able to report other users' inappropriate posts.
Rationale: So that users can help moderators with their tasks.

2.3. Non-functional Requirements

MOD-010: Processing a post for automated moderation should take no more than 3 seconds combined of queuing and computation.
Rationale: This is a reasonable compromise between user enjoyment and resource demands.

BKN-008: The system response time should be kept to a minimum of 3 seconds.

Rationale: This is a reasonable compromise between user enjoyment and resource demands.

BKN-009: A login request should be processed in less than 0.5 seconds.
Rationale: This is a reasonable compromise between user enjoyment and resource demands.

BKN-010: A database request should be processed in less than 0.5 seconds.
Rationale: This is a reasonable compromise between user enjoyment and resource demands.

2.4. Requirements Classification

The following categories exist for classification:

- Administration: Hosting the app, managing user access levels and global settings.
- Backend: Functionality whose implementation involves a lot of logic and very little front-end.
- Database: Saving and loading data.
- Front-end: Functionality that mostly involves the user interface.
- Maintenance: Fixing and improving the app post-release.
- Moderation: Approving/denying posts, deleting posts that don't belong, possibly issuing/lifting bans on users.
- Security: Ensuring integrity.
- Support: Troubleshooting problems with the app.
- User account: Relates to user identification and authentication.

2.4.1 Functional requirements

BKN-001: The system must allow a user to register and authenticate through Google.
Backend, Database, Security, User Account

BKN-002: The system must store data about users and posts and not lose it if it reboots.
Backend, Database

BKN-003: The application will be designed for both iOS and Android. All USR tests must pass on both platforms.
Backend, Maintenance

BKN-004: The system should allow optional notifications on the client device. These are no notifications, notifications on all new posts, and notifications on new posts of the specific category.
Backend

BKN-007: Available post categories are: Animals, Nature & Environment, Refugees & Immigrants, People with disabilities, Day-to-day help.
Backend, Database

MOD-001: A new post or changes to an existing post must successfully pass through an automated moderation filter to be approved for public view.
Moderation

MOD-002: A moderator should be able to delete posts by other users or moderators from public view.
Moderation

MOD-003: Rejected posts should only be visible to the author and moderators.
Backend, Moderation

MOD-005: A user can request manual approval of a post that was rejected by the automated filter.
Moderation

MOD-006: Moderators can issue and lift bans on user accounts.
Database, Moderation

MOD-007: A post can have one of the statuses Accepted, Rejected and Pending Moderation.
Database, Moderation

MOD-008: In case of a rejected post or declined changes to an existing post, the system should provide the reason for such a decision to the user.
Database, Moderation

MOD-009: A post that has been manually rejected must also be manually approved if edited.
Database, Moderation

USR-001: When a user registers, a profile is created. When this profile is viewed, the user's Google profile data is pulled to populate it. The "About me" field from USR-002, on the other hand, is saved by the application.
Database, Front-end

USR-002: A user shall be able to customize their profile with additional information in an "About me" field. It shall be at most 1,000 characters long.
Database, Front-end

USR-003: A user shall be able to list all volunteer requests.
Database, Front-end

USR-004: A user shall be able to filter volunteer requests by category.
Database, Front-end

USR-005: A user shall be able to submit a volunteer request. It shall be at most 1,000 characters long.
Database, Front-end

USR-007: A user shall be able to select a category from a list when submitting a volunteer request.
Database, Front-end

USR-008: A user shall be able to list/view their own volunteer requests.
Database, Front-end

USR-009: A user shall be able to delete their own volunteer requests.
Database, Front-end

USR-010: A user shall be able to edit their own volunteer requests. Changes must go through the moderation system.
Database, Front-end, Moderation

USR-011: A user shall be able to bookmark other users' posts.
Database, Front-end

USR-012: A user shall be able to view their list of bookmarked requests.
Database, Front-end

USR-013: A user shall be able to remove bookmarked requests from their bookmarks.
Database, Front-end

USR-014: A user shall be able to report other users' inappropriate posts.
Front-end, Moderation

2.4.2 Non-functional requirements

MOD-010: The automated moderation filter should not take more than 3 seconds to process a post.
Backend, Moderation

BKN-008: The system response time should be kept to a minimum of 3 seconds.
Backend, Database

BKN-009: A login request should be processed in less than 0.5 seconds.
Backend, Database

BKN-010: A database request should be processed in less than 0.5 seconds.
Backend, Database

2.5. Requirements Analysis

All requirements were analyzed for the following flaws:

- Premature design
- Combined requirement
- Unnecessary requirement
- Requiring non-standard hardware
- Goal non-conformance
- Ambiguity
- Being unrealistic
- Being untestable

BKN-001: This may appear to be a combined requirement, and it is, but it's for the better. A previous iteration had this split into four different requirements, but they were very thin and couldn't be independently tested.

BKN-002, **BKN-004**, **USR-002** and **USR-005** can also each be considered a combined requirement, but this comes with the benefit of testability. If they were split, the two new requirements would have to be tested together anyway.

2.6. Risk Assessment

All requirements were analyzed for the following risks:

- High performance impact
- Safety and security
- Unusual development process
- Unfamiliar implementation technology
- Non-standard data to be stored in a database
- Schedule (time-consuming)
- Involving external contractors
- Being unstable and subject to change

In cases where the risk was found to be low and obviously so, no mention is made.

BKN-001: This relies on using Google for registration and authentication. Even though the service is free, this may be considered an external contractor. However, Google is seen to be stable and reliable, so we consider this risk low.

A similar case can be made for the security aspect. We do not fear that involving Google will compromise our app's safety. On the contrary, we are happy to shift the authentication responsibility onto them, lowering our concerns about integrity and lowering users' concerns about our trustworthiness. This risk too is low.

There may be some (medium) risk of difficult implementation. This can be partly or entirely alleviated by using a technology stack for which a solution already exists for Google interaction.

BKN-003: This carries several risks. Performance, safety and the schedule itself may all be at risk (medium to high) when trying to develop an application for multiple platforms. Again, the right technology stack will severely mitigate these concerns.

MOD-001: It could be quite time-consuming (medium) to design a filter from scratch, and this filter would also be subject to change (high). If an existing solution can be implemented, these risks are both heavily mitigated, to the point that the only real danger is that this library is discontinued. This would mean that all we have to worry about is a low risk due to involving an external contractor.

BKN-007: There may be some (medium) risk of having to change the categories, but the development impact of editing an enumerated list is trivial.

2.7. Systematic Validation

All requirements were validated for the following criteria:

- Completeness
- Consistency
- Comprehensibility
- Ambiguity
- Structure
- Traceability
- Standards conformance

At an overall level, all requirements are traceable with rationale and follow the same standards. We have achieved a high quality in this iteration.

2.8. Test cases

2.8.1 Functional

BKN-001: The system must allow a user to register and authenticate through Google.
Test: Register through Google. Verify that the system knows about this Google account. Verify that the system can be accessed by logging in with this Google account.

BKN-002: The system must store data about users and posts and not lose it if it reboots.
Test: Submit and approve a post, then reboot the system and verify the post is still there. Update an "About me" field, then reboot the system and verify the change is still there.

BKN-003: The application will be designed for both iOS and Android. All USR tests must pass on both platforms.
Test: Verify that all USR tests pass on both an Android and an iOS device.

BKN-004: The system should allow optional notifications on the client device. These are no notifications, notifications on all new posts, and notifications on new posts of the specific category.
Test: Change the notification preferences and verify that they catch/ignore updates accordingly.

BKN-007: Available post categories are: Animals, Nature & Environment, Refugees & Immigrants, People with disabilities, Day-to-day help.
Test: Verify that when attempting to make a new post, these categories are available to describe it.

MOD-001: A new post or changes to an existing post must successfully pass through an automated moderation filter to be approved for public view.
Test: Make a post that shouldn't pass the filter and verify that it doesn't become publicly visible. Then, make one that should pass the filter and verify that it does.

MOD-002: A moderator should be able to delete posts by other users or moderators from public view.
Test: Verify that if a moderator deletes a post, it's no longer visible to others than moderators and the author of the post.

MOD-003: Rejected posts should only be visible to the author and moderators.
Test: Verify that if a moderator rejects a post, it's only visible to moderators and the author of the post.

MOD-005: A user can request manual approval of a post that was rejected by the automated filter.
Test: Verify that if a user requests manual approval, the post appears for moderators to approve/reject.

MOD-006: Moderators can issue and lift bans on user accounts.
Test: Verify that if a moderator bans a user, that user can no longer view, edit or post. Verify that if the ban is lifted, these privileges are restored.

MOD-007: A post can have one of the statuses Accepted, Rejected and Pending moderation.
Test: Verify that one cannot assign a custom or no status to a post.

MOD-008: In case of a rejected post or declined changes to an existing post, the system should provide the reason for such a decision to the user.
Test: Verify that when either automated or manual moderation rejects/declines a post, the user sees the same reason as the algorithm/moderator provided.

MOD-009: A post that has been manually rejected must also be manually approved if edited.
Test: Verify that an edited manually rejected post goes directly to manual moderation when resubmitted.

USR-001: When a user registers, a profile is created. When this profile is viewed, the user's Google profile data is pulled to populate it. The "About me" field from **USR-002**, on the other hand, is saved by the application.
Test: Verify that when viewing a user's profile, the information (except About me) matches that of their Google profile. Changes to the Google profile after registration should still be reflected.

USR-002: A user shall be able to customize their profile with additional information in an "About me" field. It shall be at most 1,000 characters long.
Test: Verify that the "About me" field is visible to other users and that changes to it are received, while refusing a string longer than 1,000 characters.

USR-003: A user shall be able to list all volunteer requests.
Test: Verify that an arbitrary user sees every publicly visible post in the system.

USR-004: A user shall be able to filter volunteer requests by category.
Test: Verify that ascending/descending sorting of requests results in the corresponding alphabetical/chronological ordering.

USR-005: A user shall be able to submit a volunteer request. It shall be at most 1,000 characters long.
Test: Verify that when a user submits a post, the moderation system receives it, while refusing a string longer than 1,000 characters.

USR-007: A user shall be able to select a category from a list when submitting a volunteer request.
Test: Verify that when making a post, the user is presented with the options from **BKN-007** to categorize it.

USR-008: A user shall be able to list/view their own volunteer requests.
Test: Create three users. Have each of them post three requests. Verify that for each of these users, it's possible to filter the posts so that only their own are visible.

USR-009: A user shall be able to delete their own volunteer requests.
Test: Verify that when a user deletes a post, it's removed from the database and no longer visible to anyone.

USR-010: A user shall be able to edit their own volunteer requests. Changes must go through the moderation system.
Test: Verify that when a user edits a post, the moderation system receives the new version, and if approved, the new version of the post is visible to everyone.

USR-011: A user shall be able to bookmark other users' posts.
USR-012: A user shall be able to view their list of bookmarked requests.
Test: Have User A bookmark a post by User B. User A then stops viewing the post and instead views their list of bookmarked posts. The post should be there and selecting it should take User A to the post again.

USR-013: A user shall be able to remove bookmarked requests from their bookmarks.
Test: Verify that when a user removes a bookmark, the post is removed from their list of bookmarked requests.

USR-014: A user shall be able to report other users' inappropriate posts.
Test: Verify that when a user reports a post, moderators are made aware of which post they should look at and why.

2.8.2 Non-functional

MOD-010: Processing a post for automated moderation should take no more than 3 seconds combined of queuing and computation.
Test: Simulate the system.

BKN-008: The system response time should be kept to a minimum of 3 seconds.
Test: Simulate the system.

BKN-009: A login request should be processed in less than 0.5 seconds.
Test: Simulate the system.

BKN-010: A database request should be processed in less than 0.5 seconds.
Test: Simulate the system.

3. System Interfaces

3.1. User Interfaces

3.1.1. Look & Feel

3.1.1.1. The general style of the GUI

The application GUI should follow the Material Design guidelines that are the default choice when creating Flutter applications (the usage of the Flutter framework is mentioned in Section 5. System Constraints). This decision implies the rendering of the user interface components/widgets in the Material Design context that is a native style for Android OS users but is still user friendly for the iOS users.

3.1.1.2. Colors

LAF-001: The system should allow a user to choose between light and dark themes for the client application.

LAF-002: The Material Design color system should be applied to the 2 possible GUI themes (light and dark).

LAF-003: The primary color should be picked as the one that is most frequently displayed across the GUI components.

LAF-004: Shades of GUI component colors should be used to give the illusion of a third dimension where the lighter shades represent objects that are floating higher than the darker ones.

LAF-005: Interactive components such as squares representing posts should appear to hover above lesser interactive components, such as the background.

LAF-006: Secondary color should be picked and applied to the buttons and selection controls.

3.1.1.3. Shapes

In order to visually emphasize the importance of a particular GUI component and draw the user's attention (apart from applying the color scheme), the design should adhere to the following requirements applicable to the component's shape:

LAF-008: Interactive components such as squares representing posts in the UI should have rounded edges.

3.1.1.5. Typefaces

LAF-009: The textual content should be represented using one of the sans serif fonts with the kerning spacing.
Rationale: In order to create a visually pleasing experience by interacting with the application GUI and improve the readability of the text.

LAF-010: The textual content should be represented using bold or italics highlighting.
Rationale: In order to draw the user's attention to the particular elements.

3.1.2. Layout and Navigation Requirements

3.1.2.1 General

The Material Design principles provide a path that, when followed correctly, results in a very high-quality user experience that fits quite well with the goals of this application. It will allow for well-known, smooth and uncomplicated navigation while granting much flexibility for customization which allows this project's vision to be realized while maintaining a unique look.

LAN-001: Interactive components such as squares representing posts should have gaps between each other.

LAN-002: Components of the screen should be separated into a header, body and footer.

LAN-003: Post previews should include a title and category.

LAN-004: The header should contain a button represented by a left-pointing arrow that takes the user to the previous page.

LAN-005: The footer should allow the user to quickly navigate to the home page, settings and my profile pages with the tap of a button.

LAN-006: If the user is already in the home page or settings, the respective footer buttons should take them to the start of that page.

3.1.2.2 Home page

The home page is where the application will always default to when restarted and will contain many of the important functionalities if not direct access to the next page. Scrolling downwards from the start, the user will be presented with a series of squares, each previewing a post (request for volunteers).

LAN-007: The header of the home page should allow a moderator to filter posts based on the following criteria: Recency or pending approval. For regular users, the default is recency and there is no header.

LAN-008: Scrolling down the body of the home page, the user should be able to view a series of squares that preview a volunteering request (post).

LAN-009: The user should be able to view the post by pressing on its preview.

LAN-010: The home page is where the application should always boot to when (re)started.

LAN-011: A button represented by a plus sign on the bottom right of the home page body should take the user to the post creation page.

3.1.2.3 Post creation

LAN-012: The body of the post creation page should allow the user to write a title & description and select a category.

LAN-013: The footer of the post creation page should contain extra buttons to either submit or discard the post.

3.1.2.4 Settings

LAN-014: The body of the settings should contain line separated rows of toggleable or pressable (leading you to a new page) series of customization options.

3.1.2.5 View profile

Viewing a user's profile will show a larger version of their photo and their personal description. From this page, the user can go back to where they came from or press the home button on the footer to return.

LAN-015: The body of the view profile page should display the profile picture followed by the full name of the user and their description (“About me”), from top to bottom.

LAN-016: The body of the view page should allow the user to customize their description (“About me”) if it is the profile of the logged-in user.

LAN-017: Below the personal information of the user, the body of the view profile page should present a series of post previews that, based on the filter, present the user’s authored posts or bookmarked posts.

LAN-018: A drop-down menu above the series of posts related to the viewed profile should allow the user to filter posts based on authored or bookmarked.

3.1.2.6 View post

Pressing on a post preview takes you to the complete post. The header contains an arrow pointing to the left which returns the user to the page they came from. Other headers, while containing all of the information from the preview, with extra features present. On this page, users can see the request content, which is where most of the information regarding the event will be written.

LAN-019: The body of the view post page should display the title and request content.

LAN-020: The footer of the view post page should include a button to delete and edit post.

LAN-021: A button, represented by a triple dot on the header of the page, should allow the user to report a post.

LAN-022: If the user is a moderator or the author of a post, they should be able to see if the post is Accepted, Rejected or Pending with writing below the title and colored respectively as green, red or yellow.

3.1.3. Consistency

The application will have a clear and concise consistency with clear feedback provided at every step of the navigational process. Any touch feedback resulting in some action will be met with a clear prompt that something has occurred. Any buttons that are deemed dangerous will be marked with a more red color to indicate destructive action is about to be taken with confirmation prompts.

CON-001: Pressing a button must always give the same result.

CON-002: Buttons must be of the same size.

CON-003: Buttons that trigger an action with significant risk will be highlighted in red.

CON-004: Squares representing post previews should be colored consistently according to their categories.

CON-005: Interactions with the applications should be predictable and intuitive.

3.1.4. User Personalization & Customization Requirements

As we use Google as an OAuth provider we will take certain user attributes from there such as profile picture, and name.

The application will support dark and light mode which will consist of blacker and lighter color depending on the option picked for accessibility reasons.

PAC-001: The system must use Google OAuth to import attributes such as profile picture and name by default.

3.2. Interfaces to External Systems or Devices

3.2.1. Software Interfaces

SIN-001: The system should interact with a database service to be able to store and retrieve data.

3.2.2. Hardware Interfaces

The system does not utilize any special hardware.

3.2.3. Communications Interfaces

CIN-001: The application server must have access to the internet.
Rationale: The application server requires networking to be able to reach out to the client applications.

The system will require internet access to communicate with the application server. The application server requires internet access to be able to reach out to the client applications.

CIN-002: The application must have access to the device I/O interface.

CIN-003: The application shall have access to device memory interface.
Rationale: While not essential, caching rarely saved settings will lead to improved efficiency and performance .

4. Business Rules

4.1. As the mobile application system is designed by volunteers for volunteers, the content of the application posts should not contain provocative expressions, requests or advertisements related to the monetary transactions, etc.

4.2. The application may not be used by for-profit organizations to search for free labor.

5. System Constraints

5.1. Implementation constraints

CIM-001: Application must be implemented to be able to support both Android and iOS mandated by BKN-003; The application will be designed for both iOS and Android. All USR tests must pass on both platforms.

CIM-002: Application must be designed with Flutter mandated by the The general style of the GUI section of the User Interfaces section in this document.

CIM-003: The system must support CRUD (Create / Read / Update / Delete), mandated by USR-009 to support delete operations, USR-010 to support update operations. USR-005 required create operations. USR-003 for supporting read operations. All the constraints are also further specified by the requirements tagged with Database.

CIM-004: The system must implement storage capability that is persistently mandated by BKN-002; The system must store data and not lose it if it reboots.

CIM-005: Application must follow material theme design mandated by the The general style of the GUI section of the User Interfaces section in this document.

CIM-006: System must support two access levels user and moderator constrained by the suffix **MOD** requirements and **USR** requirements.

CIM-007: The system must integrate with Google's services to be used as an authentication provider mandated by BKN-001.

5.2. Deployment Constraints

COD-001: Application component of the system must be deployed to Android and iOS mandated by requirement BKN-003.

6. Use-Cases

6.1. General Use-Case: Växjö Volunteers Functionality.

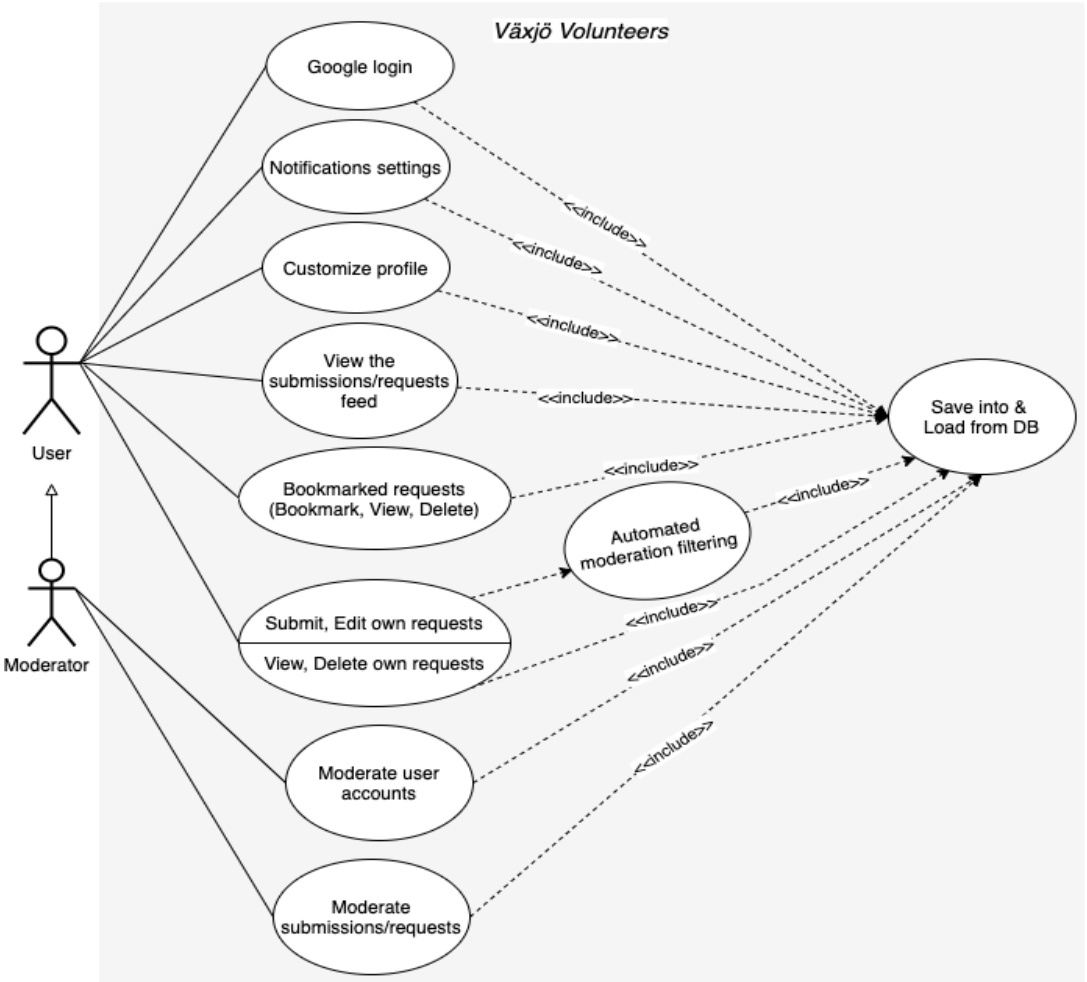


Figure 6.1. General Use-Case: Växjö Volunteers Functionality

6.1.1. Brief Description

This use case gives a general overview of the Växjö Volunteers mobile application functionality. The greyish area in Figure 6.1 represents the scope of the mobile application system.

6.1.2. Actor Brief Descriptions

- **User** is a standard non-privileged application user;
- **Moderator** is a privileged application user, who is responsible for the moderation of the application content, therefore moderator is capable of deleting volunteer requests from the public view, blocking and unblocking

user accounts, overall monitoring of the users' activity within the application.

6.1.3. Preconditions

1. User must have a mobile device with an Android or iOS operating system.
2. User must have the application installed on their mobile device.
3. User must have internet access via Wi-Fi connection.

6.1.4. Basic Flow of Events

1. The use case begins when the user launches the application on their mobile device.
2. The user passes the authentication check.
3. The user gets access to the application functionality that is granted according to the user credentials.
4. The user sees the current feed of the submissions and explores the navigation layout.
5. The user exists application.
6. The use case ends.

6.1.5. Alternative Flows

1. If in step 1 of the basic flow the application system does not respond then no further actions are available and the information message is displayed to the user (according to Connectivity In Case Of Technical Issues (SAR002)).
2. While exploring the navigation layout on step 4 of the basic flow, the user may choose to perform alternative actions according to the Subflows of the current section.
3. If in step 5 of the basic flow the user chooses to log out from the system, then the system does not preserve the previous log in session.

6.1.6. Subflows & Key Scenarios

1. Log in to the application system.
2. Saving in and loading from the database storage.
3. Submit, View, Edit and Delete user’s own requests.
4. View the submissions/requests feed.
5. Bookmark the other users’ posts. View bookmarks and Remove items from the bookmarked list.
6. Automated moderation filtering.
7. Manual moderation of the users’ accounts and submissions/requests by the moderator.

6.1.7. Post-conditions

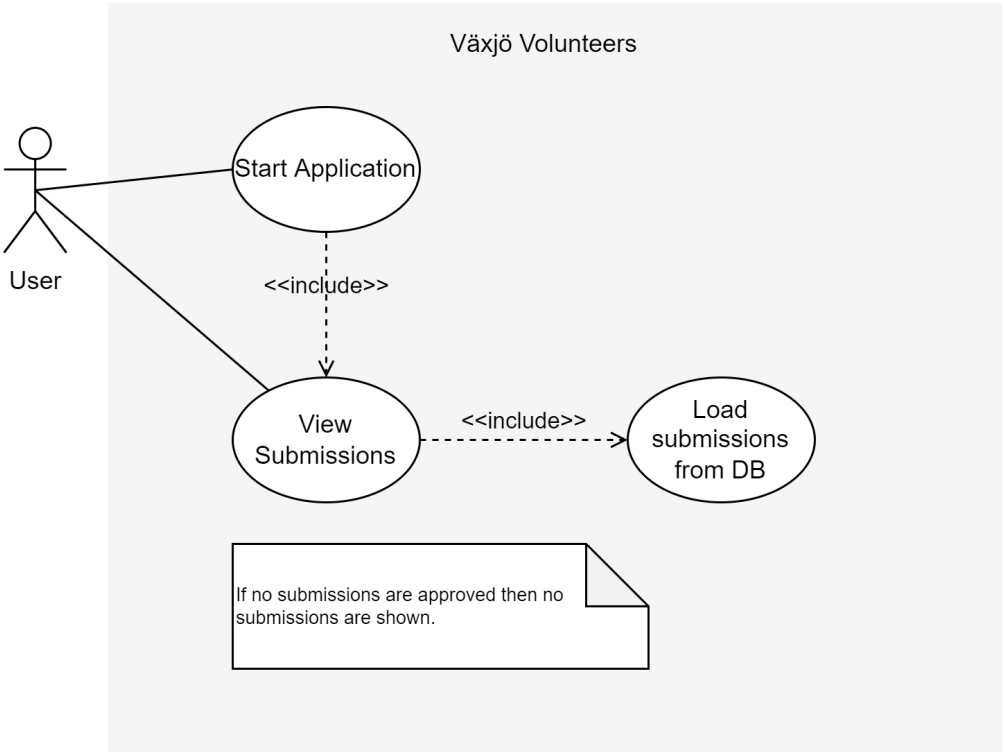
1. The user was able to perform tasks within the application system.

2. The moderator was able to perform tasks within the application system.

6.1.8. Special Requirements

Non-functional requirements that are elicited in Section 2 and Section 3 of the current document.

6.2. Use-Case: View submissions



6.2.1. Brief Description

This use-case represents the functionality of USR-003; The system must allow a user to list all volunteer requests.

6.2.2. Actor Brief Descriptions

User is a standard non-privileged application user.

6.2.3. Preconditions

1. User must have the application installed.
2. User must be logged in.
3. System must be accessible.

6.2.4. Basic Flow of Events

1. The use-case begins when the actor has opened the application.
2. Submissions will show up on the screen.
3. The use-case ends.

6.2.5. Alternative Flows

1. If there are no submissions submitted per requirement USR-005 and use-case Submit a volunteer request, there will be no submissions to view.

2. If there are rejected submissions or pending submissions they will be hidden per requirement MOD-007.

6.2.6. Subflows

There are no subflows.

6.2.7. Key Scenarios

1. User can at a quick glance see all available submissions.

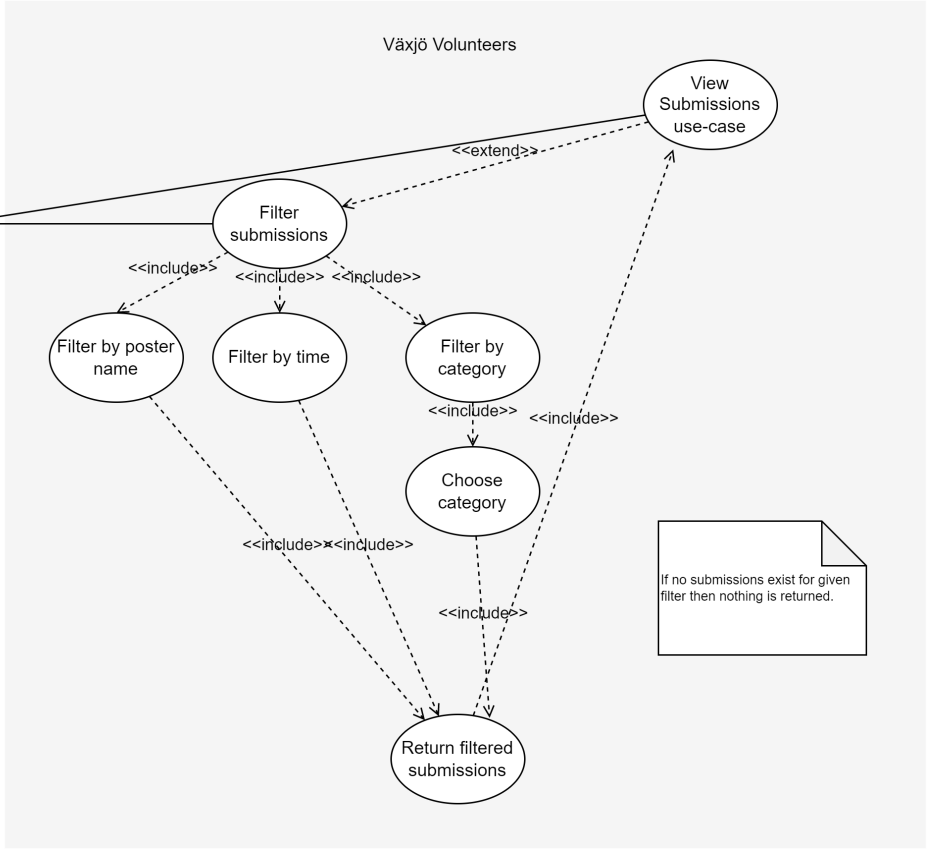
6.2.8. Post-conditions

1. The user is left with a view of submissions if available

6.2.9. Special requirements

There are no special requirements for this use-case.

6.3. Use-Case: Filter submissions by category



6.3.1. Brief Description

This use-case represents the functionality of USR-004; The system must allow a user to sort volunteer requests by poster name, time, or category.

6.3.2. Actor Brief Descriptions

User is a standard non-privileged application user.

6.3.3. Preconditions

1. User must have the application installed.
2. User must be logged in.
3. The system must be accessible.
4. Use-case 6.2: View submissions must be done.

6.3.4. Basic Flow of Events

1. The use-case sees all available submissions after use-case view submissions.
2. User will press the filter category button.
3. The user will choose the type of filter by poster name, time or category per requirement USR-004.

4. User will see matching submissions.
5. The use-case ends.

6.3.5. Alternative Flows

1. If there are no submissions submitted using USR-005 that matches there will be no submissions to view and use-case will end.

6.3.6. Subflows

1. If a user picks a category then the user will choose any of the categories presented from non-functional requirement BKN-007.

6.3.7. Key Scenarios

2. User can filter all submissions that relate to them.

6.3.8. Post-conditions

2. The user is left with a view of filtered submissions.

6.3.9.Special requirements

1. The special requirements from Use-Case 6.2 apply from preconditions.
2. Available categories to filter from per BKN-007.

6.4. Use-Case: Submit a volunteer request.

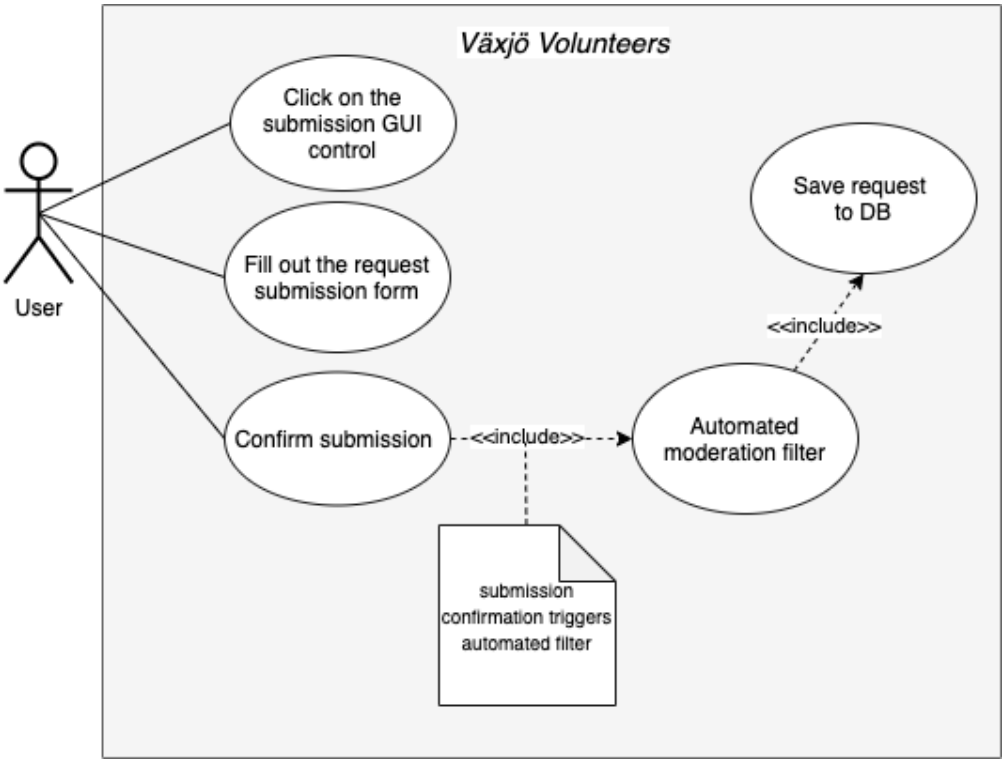


Figure 6.4. Use-Case: Submit a volunteer request.

6.4.1. Brief Description

This use-case represents the functionality of USR-005: The system must allow the user to submit a volunteer request.

6.4.2. Actor Brief Descriptions

- **User** is a standard non-privileged application user.

6.4.3. Preconditions

1. User must have the application installed.
2. User must have internet access.
3. User must be logged in.
4. System must be accessible.

6.4.4. Basic Flow of Events

1. The use-case begins when the user decides to submit a new volunteer request by clicking a respective control element (e.g. the button “submit new request”).
2. The system provides a user with a new request submission form to fill out.

3. The user fills out the content of the submission form.
4. The user confirms a new request submission by clicking the respective control element (e.g. the button “OK”/”Confirm”).
5. The use case ends.

6.4.5. Alternative Flows

1. If on step 4 of the basic flow the user does not want to confirm request submission, then the user can just close the submission form.

6.4.6. Subflows

1. Pick up the category when filling out the submission form USR-007.
2. Trigger the automated moderation filter MOD-001.
   Saving to the database storage is modelled just to show the subsequent flow of events after the automated moderation filtering.

6.4.7. Key Scenarios

1. The request submission form to fill out. More specifically step 2 in the Basic Flow of Events.

6.4.8. Post-conditions

1. Availability of the submitted volunteer request for the subsequent mandatory automated moderation and saving to the database storage.

6.4.9. Special Requirements

1. **BKN-007**. Available post categories are Animals, Nature & Environment, Refugees & Immigrants, People with disabilities, Day-to-day help.

6.5. Use-Case: Automated moderation filter

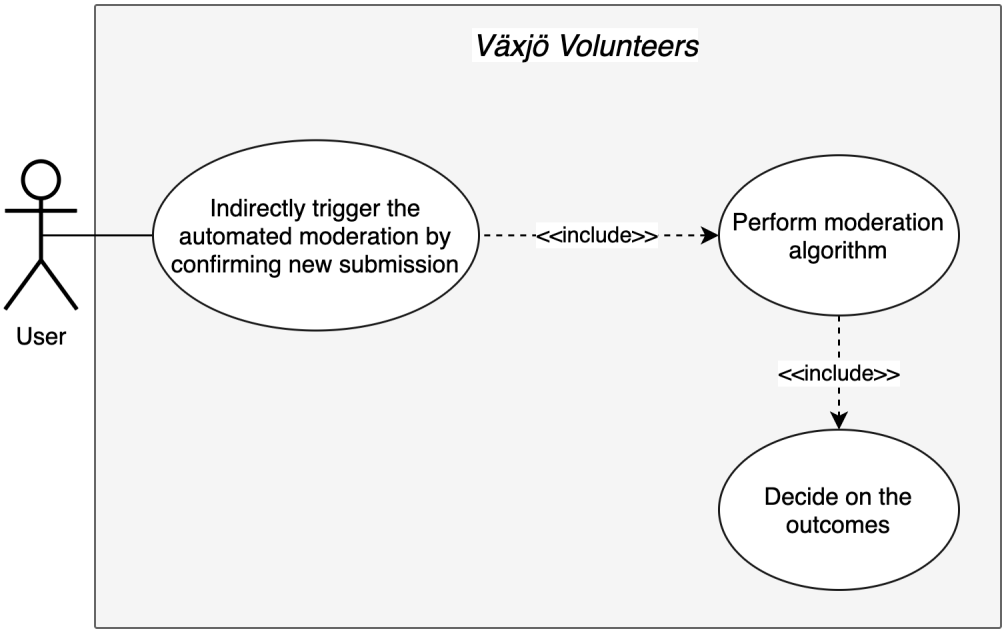


Figure 6.5. Use-Case: Automated moderation filter

6.5.1. Brief Description

This use-case represents the functionality of MOD-001: A new post or changes to an existing post must successfully pass through an automated moderation filter to be approved for public view.

6.5.2. Actor Brief Descriptions

User is a standard non-privileged application user. In this use-case the submission of a new volunteer request or changes to the existing submission trigger the automated moderation filter, thus User indirectly activates the use-case scenario.

6.5.3. Preconditions

1. The user has confirmed the submission of the new volunteer request or changes to the already existing submission.

6.5.4. Basic Flow of Events

1. New submission or changes to the already existing submission trigger an automated moderation filter/algorithm.
2. The system parses the content to be moderated against the matching keywords/inappropriate terms.

3. The system decides on the outcomes of the moderation depending on the matches that it found or not found in step 2 of the current basic flow of events.

6.5.5. Alternative Flows

There are no alternative flows.

6.5.6. Subflows

1. Assigning a status to the submission according to the moderation outcomes according to MOD-007.

6.5.7. Key Scenarios

1. The implementation of the algorithm that provides

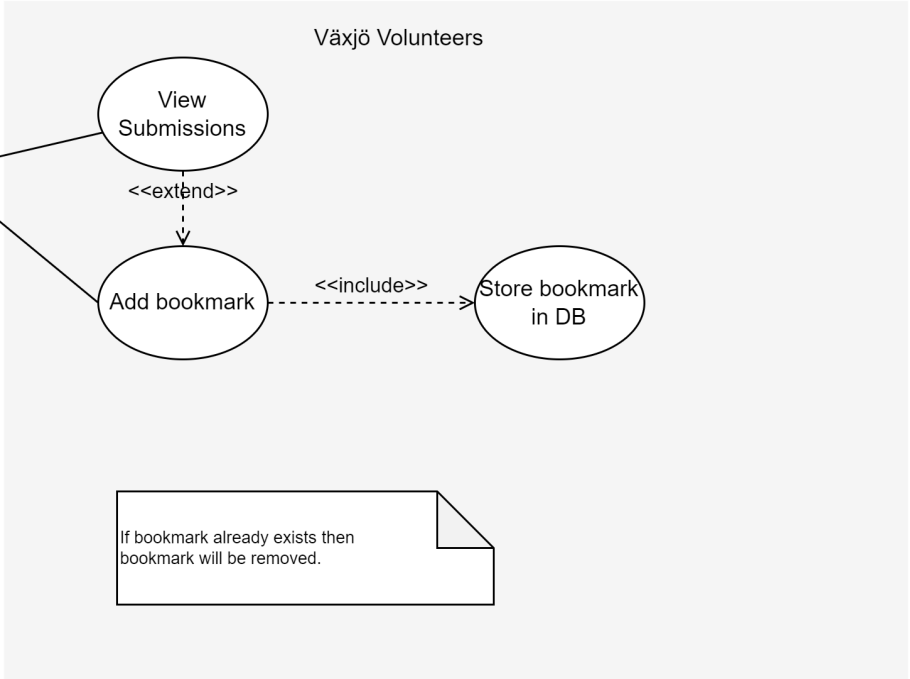
6.5.8. Post-conditions

1. A moderated submission that is assigned a status depending on the results of the automated moderation filter and is ready to be stored in the database.

6.5.9. Special Requirements

1. **MOD-010:** The automated moderation filter should not take more than 3 seconds to process a post.

6.6. Use-Case: Bookmark



6.6.1. Brief Description

This use case represents the functionality of the USR-011 , USR-012 and USR-013 requirements.

6.6.2. Actor Brief Descriptions

- **User** is a standard non-privileged application user.

6.6.3. Preconditions

1. User must have the application installed
2. User must have internet access.
3. User must be logged in.
4. The system must be accessible.

6.6.4. Basic Flow of Events

1. The use case begins when the user is browsing the post feed
2. The user presses the bookmark icon of a specific post.
3. The icon turns red to mark that the post is bookmarked.
4. The post is added to a list of bookmarks.
5. The use case ends.

6.6.5. Alternative Flows

1. If on step 2 the icon is already red, pressing the button will result in the bookmark icon returning to its original state and the post being removed from the list of bookmarks.

6.6.6. Subflows

Subflow 1. Viewing the list of bookmarked posts.

1. The user presses the three dots icon on the top right of the application.
2. The user selects the option view list of bookmarks.
3. The posts in the standard feed are replaced with the posts that have been bookmarked.

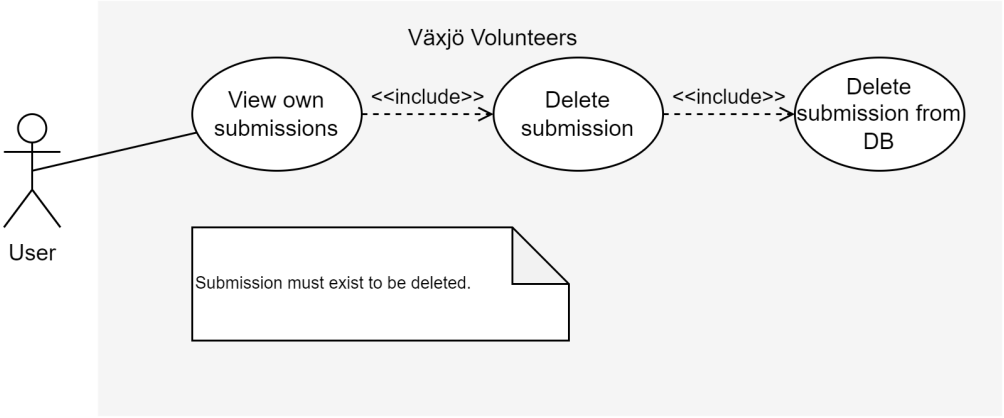
6.6.7. Key Scenarios

1. Users want to keep track of posts of interest to them without applying to volunteer.

6.6.8. Post-conditions

2. A post has to be bookmarked in order for it to appear in the list.
3. A post has to be bookmarked in order for it to be removed from the list of bookmarks.

6.7. Use-Case: Delete Submission



6.7.1. Brief Description

This use case represents the functionality of USR-009; The system must allow a user to delete their own volunteer requests.

6.7.2. Actor Brief Descriptions

- **User** is a standard non-privileged application user.

6.7.3. Preconditions

1. User must have the application installed
2. User must have internet access.
3. User must be logged in.
4. The system must be accessible.

6.7.4. Basic Flow of Events

1. The use case begins with the end of use-case view submissions.
2. The user picks one of their own submissions to delete.
3. The user deletes by pressing the GUI control element associated with delete.
4. The user confirms the action.
5. Submission is deleted.
6. The use case ends.

6.7.5. Alternative Flows

1. If the user does not confirm the action with the GUI control element then the use-case jumps to 15 and ends.

6.7.6. Subflows

There are no subflows.

6.7.7. Key Scenarios

1. Users want to delete their own volunteer requests if they change their mind or request is completed.

6.7.8. Post-conditions

1. Post is deleted from submissions for all users.
2. Post is left unaltered from submissions.

Appendix – Time Report

[The time report shows dates, team-member, activity, time spent in hours]

Notes

- "All members" specifies that the activity was done together over a live group meeting.

Timesheet

Date	Member	Activity	Time (hours)
2022/04/11	All members	Initial brainstorm	1h 30m
2022/04/11	All members	Stakeholder identification	1h
2022/04/11	All members	Requirements elicitation	1h
2022/04/12	Albert	Formatting requirements, adding rationale, writing req. classification & analysis	1h 30m
2022/04/12	Alija	Suggesting a few things on System-Wide requirements, Working on Software Constraints	30m
2022/04/12	Richard	Review of requirements and edit suggestions	30m
2022/04/12	Alija	Writing on software constraints	30m
2022/04/12	Kateryna	Business Rules, writing section 4.1. System Access Rules	40m
2022/04/12	Kateryna	Business Rules, writing section 4.2.	40m
2022/04/13	Atakan	Planning for section 3.1.1	30m
2022/04/13	Alija	Work and revise section 5. Software Constraints	1h
2022/04/13	Atakan	Writing section 3.1.1	45m
2022/04/13	Atakan	Planning section 3.1.2	15m

2022/04/13	All members	Meeting regarding the document	1h
2022/04/13	Albert	Finished first draft of section 2	3h
2022/04/13	Kateryna	Business Rules, planning and writing section 4.3. Moderator Functionality Rules	40m
2022/04/14	Alija	Making suggestions, working on 3.2. Draft on use-case list submissions.	1h
2022/04/14	Kateryna	Writing suggestions/comments about the user and moderation requirements	45m
2022/04/14	Kateryna	Writing revisions of the Business Rules	30m
2022/04/15	Albert	Started on version 2 of the requirements	1h
2022/04/15	Kateryna	Modelling and writing combined use-case for requirements USR-003, USR-006, MOD-001. Work in progress.	1h
2022/04/16	Alija	Writing suggestions for section System-wide requirements, Working on 3.2 and 5.0	1h
2022/04/16	Kateryna	Writing basic and alternative flows of events for the combined use-case of requirements USR-003, USR-006, MOD-001. Planning to write the remaining sections of this use-case.	45m
2022/04/16	Richard	Wrote introduction	30m
2022/04/16	Atakan	Planning and writing section 3.1.2	1h
2022/04/17	Kateryna	Writing subflows, key scenarios and post-conditions for the combined use-case of requirements USR-003, USR-006, MOD-001.	1h

2022/04/18	Richard	Planning next use case and edits for the system interface section.	2h
2022/04/18	Alija	Working on combined use-case for requirement USR-002 and USR-006	2h10m
2022/04/18	Richard	Added Combined use cases USR-012 , USR-013 and USR-014	1h
2022/04/18	Alija	Working on UML for combined use-case for requirement USR-002 and USR-006	1h
2022/04/18	Kateryna	Suggestions about Look and Feel section. Planning to split the combined use-case of req. USR-003, USR-006, MOD-001	1h
2022/04/19	Kateryna	Revision of the system-wide requirements	1h 30min
2022/04/19	Alija	Suggesting on system-wide, and revising Interfaces to external systems.	30m
2022/04/19	Alija, Kateryna, Albert	Working on revised requirements.	1h40m
2022/04/20	Alija	Working on 3.2	30m
2022/04/20	Albert	Cleaned up requirements v2, classified them	2h
2022/04/20	Kateryna	Labeling and rewriting Look & Feel 3.1.1. requirements	40m
2022/04/20	Albert	Risk assessment and systematic validation of new requirements, plus layout-related work	1h20m
2022/04/20	Kateryna	6.1.General Use-Case: Växjö Volunteers Functionality. 6.4. Submit a volunteer request (revising)	4h
2022/04/20	Alija	Working on Use-case 6.3 and 6.4	3h

2022/04/20	Richard	Rewriting favorites use case to bookmark use case and research for section 3	1h
2022/04/21	Albert	Test cases for requirements v2	1h
2022/04/21	Atakan	Refactoring and adding to 3.1.1 and 3.1.2	3h
2022/04/21	|Richard	Add requirements to 3.1.3 and 3.1.4	1h
2022/04/21	Alija	Work on use-case 6.2 and 6.3 UML.	3h
2022/04/21	Richard	Rewriting 3.2.3 in requirement	
2022/04/21	Albert	Linking between requirements in section 2	45m
2022/04/21	Richard	Added introductions to 3.1.2.1 - 3.2.1.6 based on writing by Atakan.	30m
2022/04/21	Alija	Wrapping up system constraints.	1h40m
2022/04/21	Alija	Add UML for use-case bookmark, and add delete submission use-case.	2h
2022/04/21	Albert	General participation in final touches	1h
2022/04/21	Alija	Finish touches	30m
2022/04/21	Kateryna	Use-cases: general functionality, submit a new request, automated moderation filter. Polishing section 3.1. Document formatting, general participation in final touches	8h