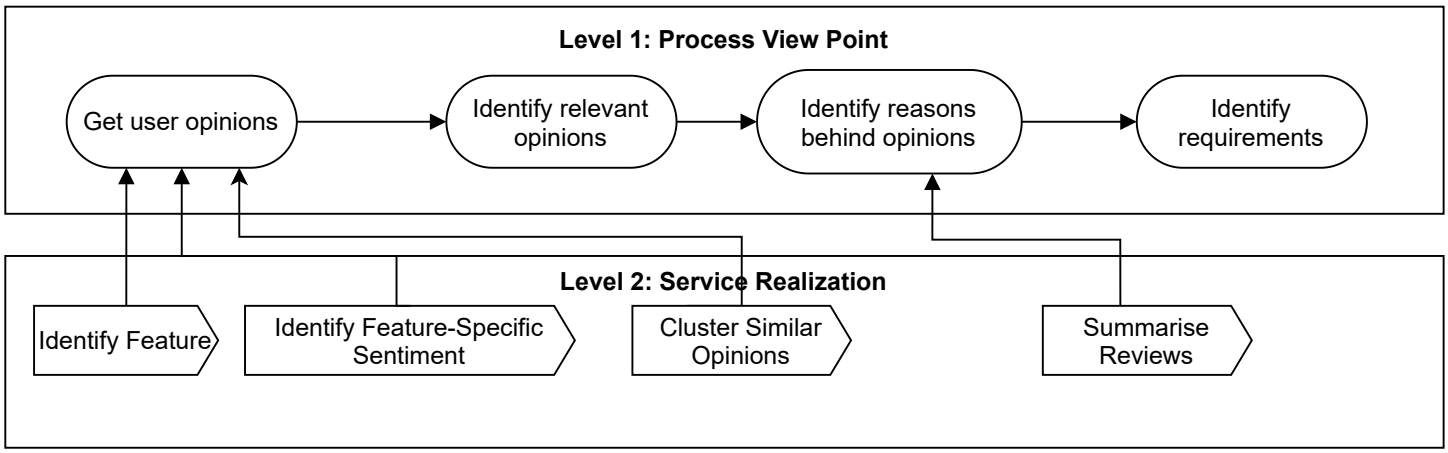
UC1 - Requirements Elicitation (1)



DESCRIPTION OF STEPS IN THE PROCESS VIEW POINT:

Get user opinions (input: review; output: opinions): identify user opinions from app reviews.

Identify relevant opinions (input: opinions, statistics; output: opinions'): identify opinions indicating problems with core features, reported by large number of users or increasing rapidly

Identify reasons behind opinions (input: opinions, reviews; output: user needs): Identify reason behind negative opinions; Understand what should be done improved.

Identify requirements (input: opinions; output: user needs): elicit requirements based on users' needs

USE CASE VIEW POINT:

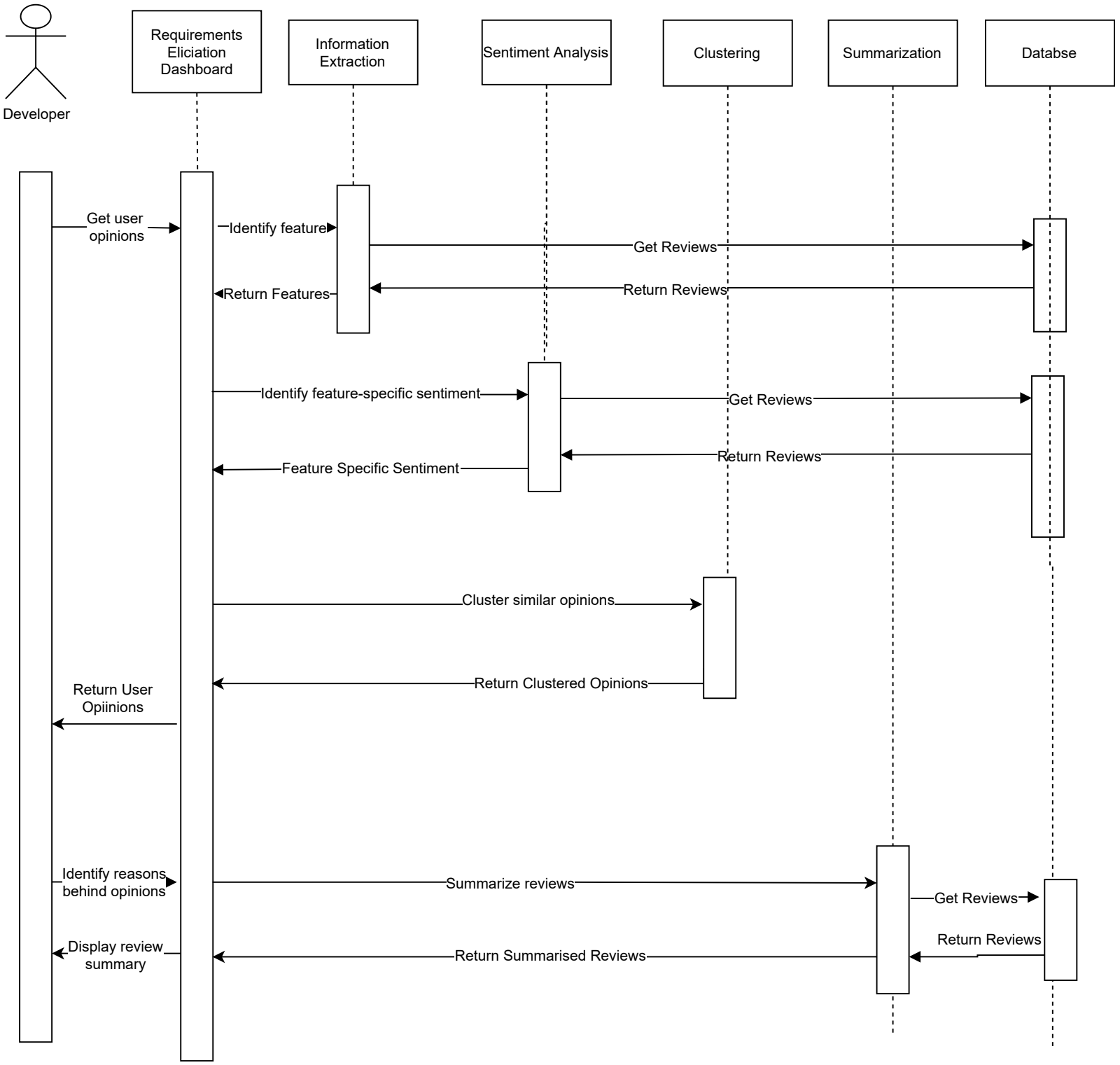
Use Case Name: Requirement Elicitation (1)

Description: App Developer (AD) Elicit Requirements By Analysing Users Opinions

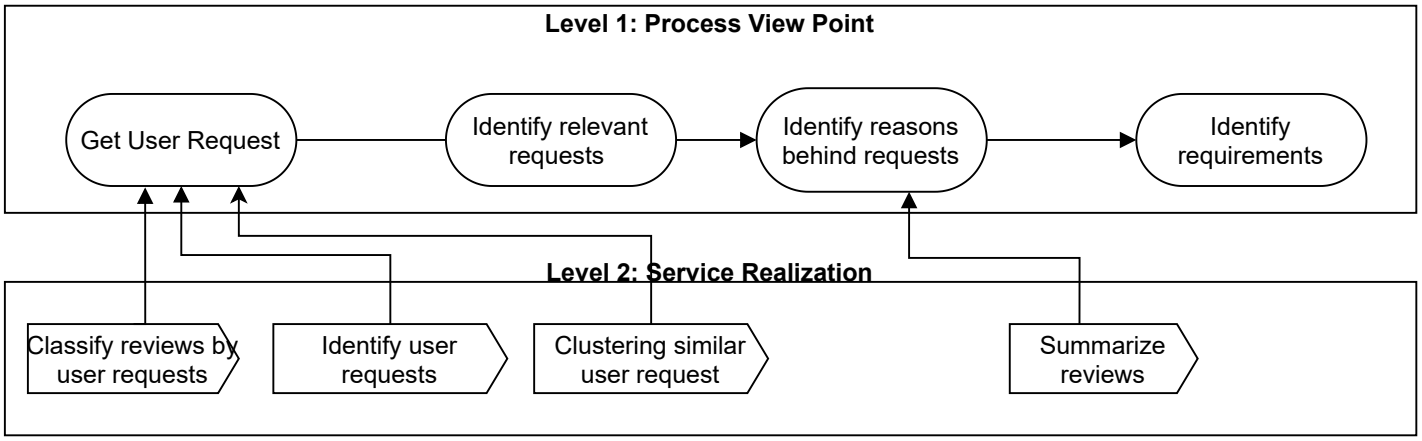
Normal Flow:

1. AD uses a system to get user opinions from app reviews.
2. The system identifies user opinions; cluster them thematically and display to user.
3. AD identify relevant opinions.
4. AD system uses a system to identify reasons behind opinions.
5. The system displays summary of reviews for selected opinion.
6. AD uses a system to identify requirements.

SIMPLIFIED SEQUENCE DIAGRAM ILLUSTRATING SERVICE COMMUNICATION



UC1 - Requirements Elicitation (2)



DESCRIPTION OF STEPS IN THE PROCESS VIEW POINT:

Get user requests (input: reviews; output: user requests): get user requests from app reviews i.e., bug report, feature request

Identify relevant user requests (input; user requests; output: requests'): identify requests indicating core problems, reported by large number of users or increasing rapidly

Identify reasons behind requests (input: user requests; output: user needs): Identify reasons behind requested features and reported problems; Understand what and why should be improved.

Identify requirements (input: user requests; output: requirements): elicit requirements based on users' need

USE CASE VIEW POINT:

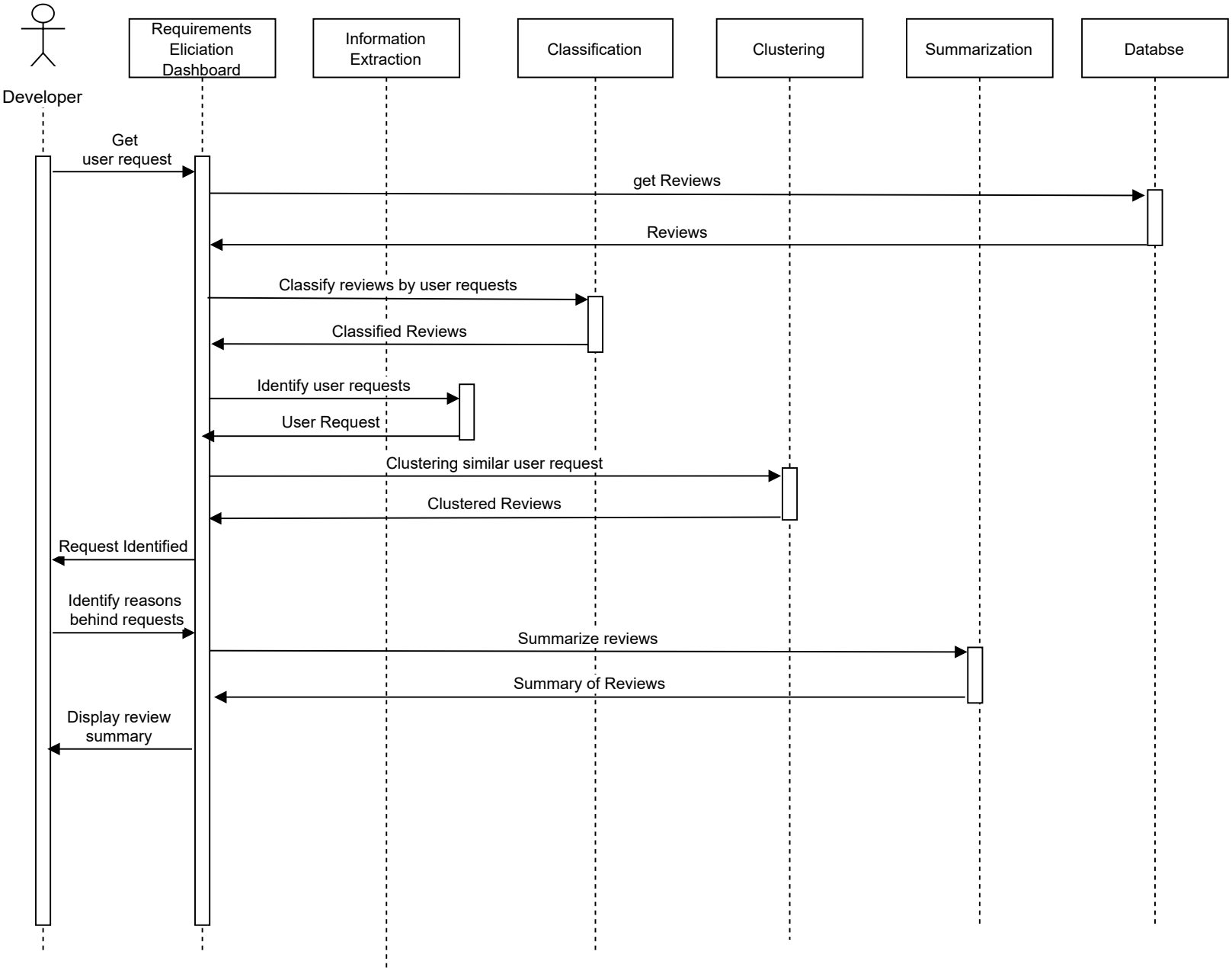
Use Case Name: Requirement Elicitation (2)

Description: App Developer (AD) Elicit Requirements by Analysing User Requests

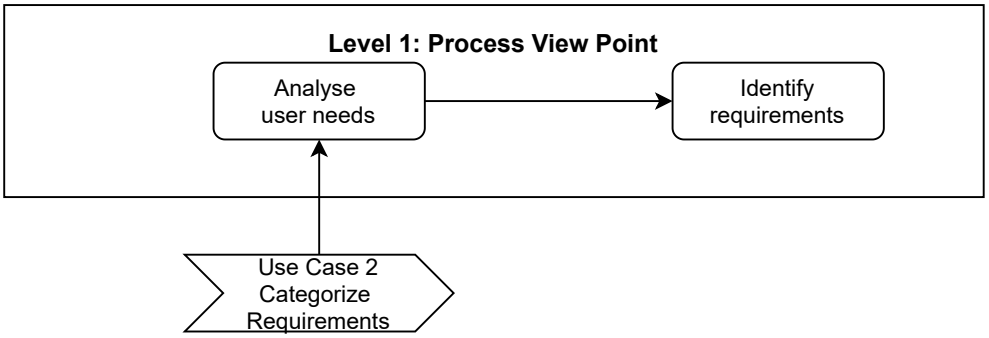
Normal Flow:

1. AD uses a system to get user requests from app reviews.
2. The system classifies reviews by user requests type; extracts user request from reviews; and clusters user requests.
3. AD uses a system to identify relevant user requests.
4. The system summarises user requests; Display user requests and their statistics.
5. AD uses a system to identify reasons behind requests.
6. The system displays summary of reviews for selected requests;
7. AD uses a system to identify requirements.

SIMPLIFIED SEQUENCE DIAGRAM ILLUSTRATING SERVICE COMMUNICATION



UC1 - Requirements Elicitation (3) (requires UC2)



BUSINESS PROCESS VIEW POINT:

Analyze user needs (input: classified reviews; output: user needs): Analyze categorized reviews conveying specific types of user needs (FR, NFR); Understand what and why should be improved.

Identify requirements (input: user requests; output: requirements): elicit requirements based on users' need (manually or automatically in Future Work)

USE CASE VIEW POINT:

Use Case Name: Requirement Elicitation (3)

Pre-Condition: UC 2 (Requirements Classification) was performed.

Description: App Developer (AD) Elicit Requirements From Review Classified with Types of Requirements

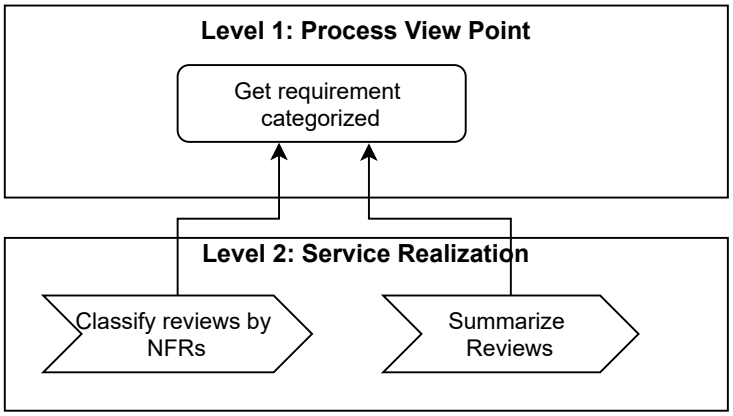
Normal Flow:

1. AD uses a system to **analyze use needs**; Examine reviews to understand user needs and their rationale.

2. The system **displays a summary of reviews** conveying concrete types of requirements.

3. AD **identifies requirements** from reviews

UC2 - Requirements Categorizations



BUSINESS PROCESS VIEW POINT:

Get requirement categories (input: reviews; output: classified reviews): get reviews classified based on requirements categories they convey i.e., FR, NFR and detailed NFR.

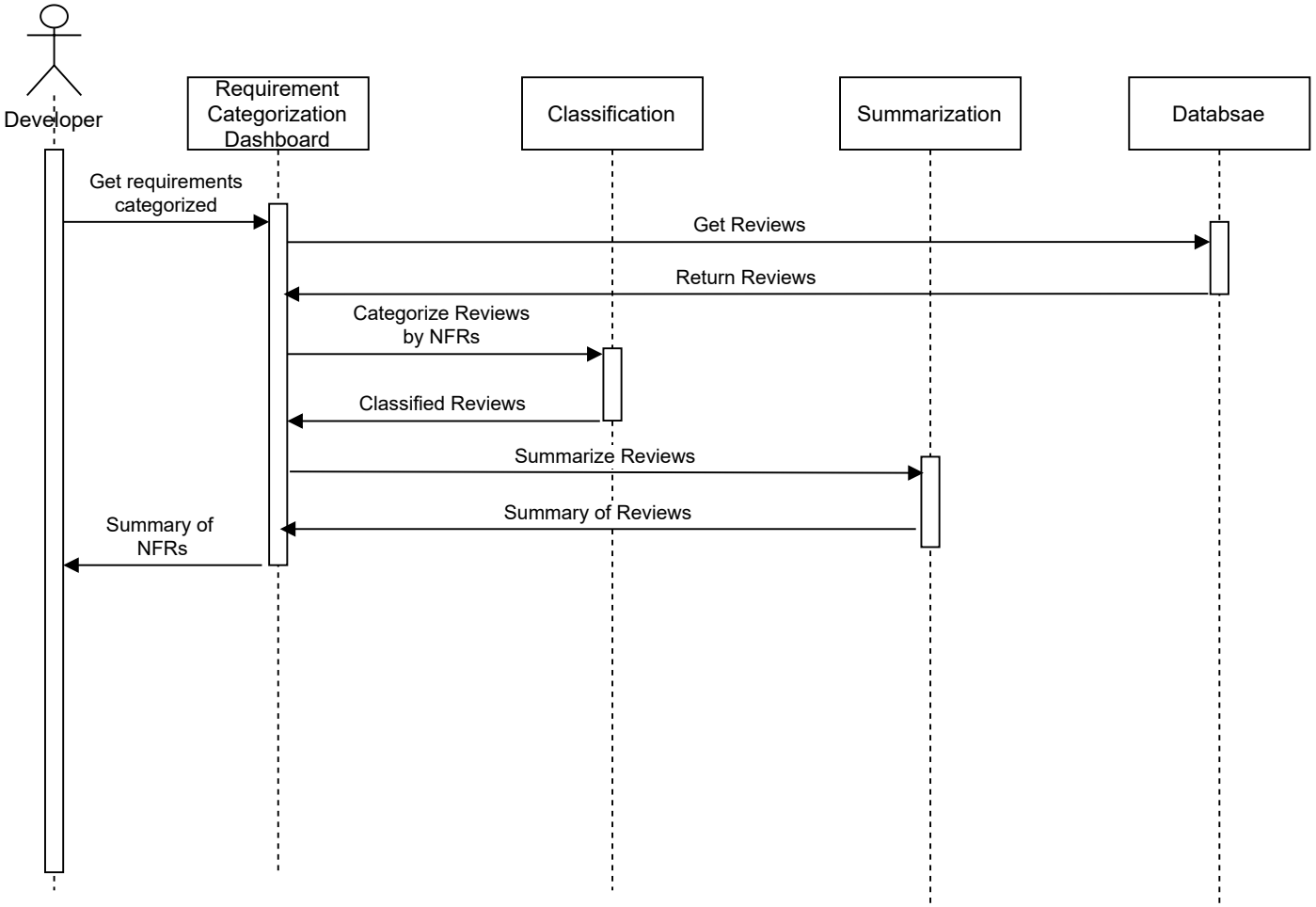
USE CASE VIEW POINT:

Use Case Name: Requirement Categorization

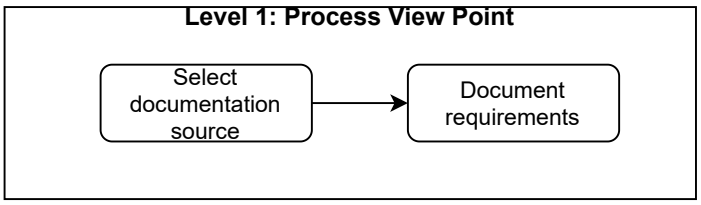
Description: App Developer (AD) Categorize Requirements in Reviews

Normal Flow:

1. AD uses a system to **categorize user reviews by requirements they convey**
2. The system **displays summary of types of requirements** review convey



UC3 - Requirements Specification



BUSINESS PROCESS VIEW POINT:

Select documentation (input: reviews/their summary; output: review/reviews summary): Select review or a summary of reviews conveying a new requirement

Document requirement (input: user requests; output: requirements): ad hoc requirement specification document is generated; stored in the system and export as an external file. The report contain information like requirements type (FR/NFN), concerned function / quality, no. reported users, speed of their growth, first reported date

USE CASE VIEW POINT:

Use Case Name: Requirement Specification

Description: App Developer (AD) use a system to specify Requirements From Reviews or Their Summary

Pre-Condition:

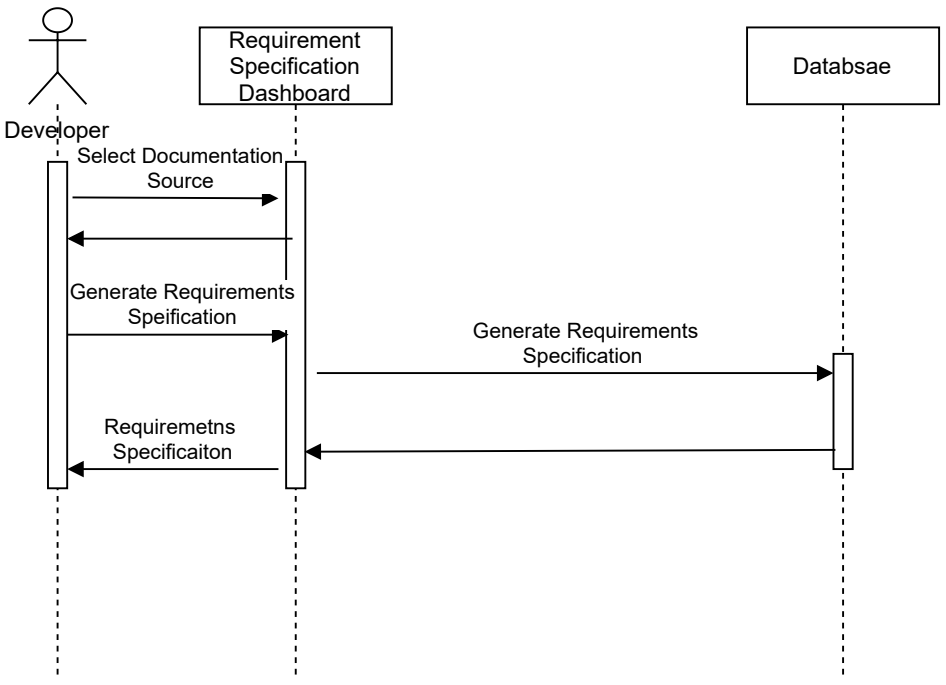
1. Requirement Elicitation Use Case (out of 1-3) was conducted.

Normal Flow:

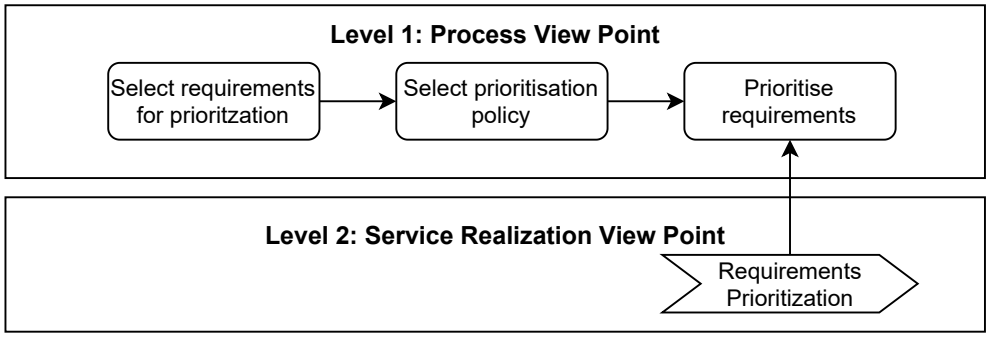
1. AD uses a **system to select documentation source**; Either summaries of reviews or specific reviews conveying user requirement

2. AD uses a system to **generate requirements documentation** (ad hoc).

3. The system generate **requirements specification form; Store the form in database and export as an external file.**



UC3 - Requirements Prioritization (1)



BUSINESS PROCESS VIEW POINT:

Select requirements for prioritisation (input: requirements; output: requirements'): Select requirements for prioritisation

Select prioritisation policy (input: list of prioritisation policies; output: prioritisation policy'): Select prioritisation policy

Prioritise requirements (input: requirements; output: prioritised requirements'): prioritise requirements

USE CASE VIEW POINT:

Use Case Name: Requirements Prioritisation (1)

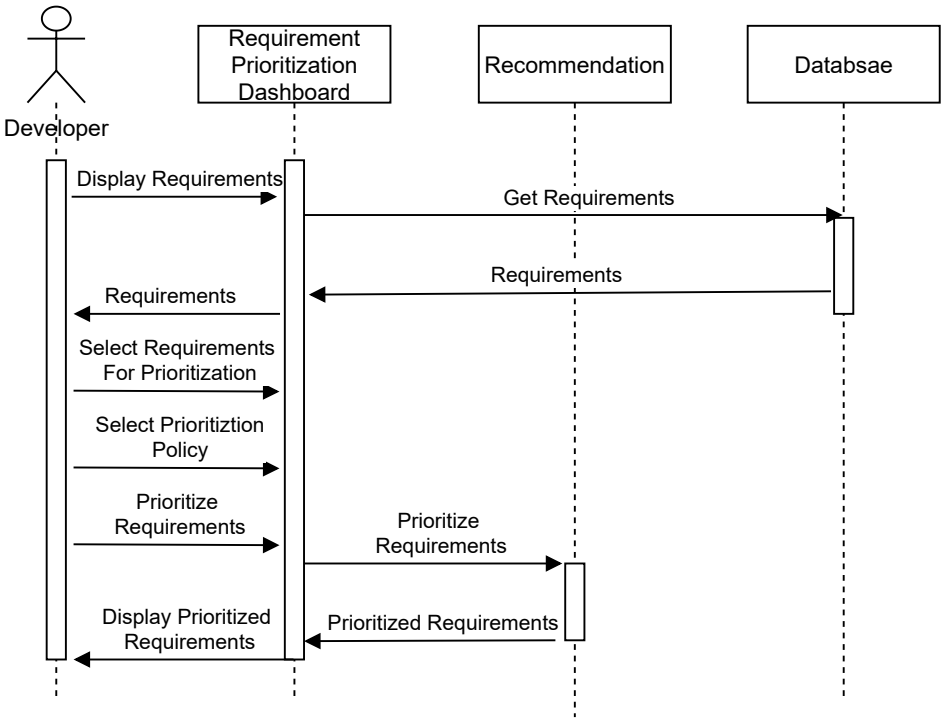
Description: App Developer (AD) use a system to prioritise Requirements conveyed in Reviews

Pre-Condition:

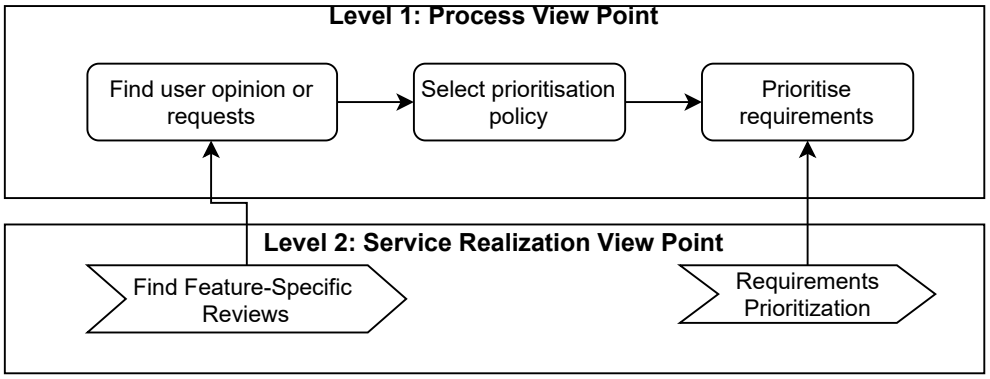
1. Requirements Elicitation Use Case was conducted.
2. Requirements Specification was conducted

Normal Flow:

1. AD uses a **system to select requirements** to be prioritised
2. AD uses a **system select prioritisation policy** (procedure and criteria)
3. AD use a **system to prfioritize requirements**
4. The system prioritises requirements and display results to AD



UC3 - Requirements Prioritization (2)



BUSINESS PROCESS VIEW POINT:

Find user opinions or requests (input: opinions/user requests; output: opinions/requests and their statistics): Find user opinions or user requests and their statistics

Select prioritisation policy (input: list of prioritisation policies; output: prioritisation policy'): Select prioritisation policy

Prioritise requirements (input: requirements; output: prioritised requirements'): prioritise requirements

USE CASE VIEW POINT:

Use Case Name: Requirements Prioritisation (2)

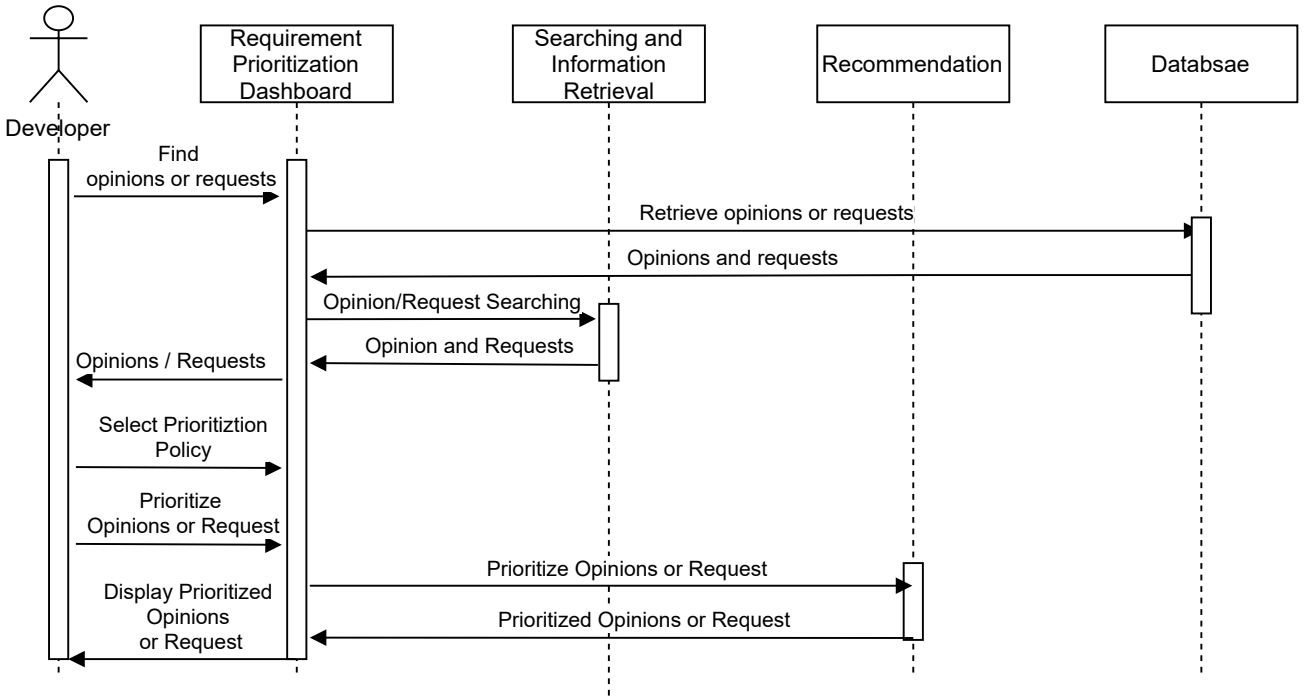
Description: App Developer (AD) use a system to prioritise Requirements in Reviews

Pre-Condition:

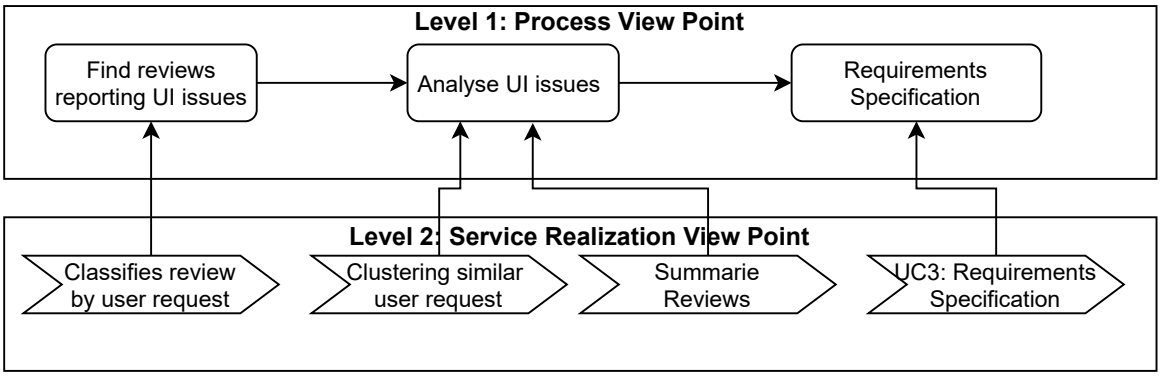
1. User opinions and user requests has been already mined from app reviews and summarised.
2. AD knowns requirements expressed as negative opinions or user requests; They wants to prfioritize them.

Normal Flow:

1. AD uses a system to **find specific user opinions or user requests** and their statistics
2. The system displays statistics about user requests / opinions
3. AD use a **system to select prioritisation policy** (procedure criteria)
3. AD use a **system to prfioritize requirements**
4. The system prioritises user opinions / user requests and display results to AD



UC5 - User Interface Design



BUSINESS PROCESS VIEW POINT:

Find reviews reporting UI issues (input: reviews; output: reviews reporting UI issues): AD finds reviews discussing UI issues reported by app users. These reviews can provide recommendation for improving UI (e.g., how to improve interface layout, boost readability and easy app navigation)

Analyse UI issues (input: reviews; output: user needs): AD examines the reviews to understand what is the reported issue about e.g., problem or recommendation. AD interprets what possible change should be made to address the issue.

UC3: Requirements Specification (input: user needs; output: requirements): AD elaborates requirements specification addressing UI issues;

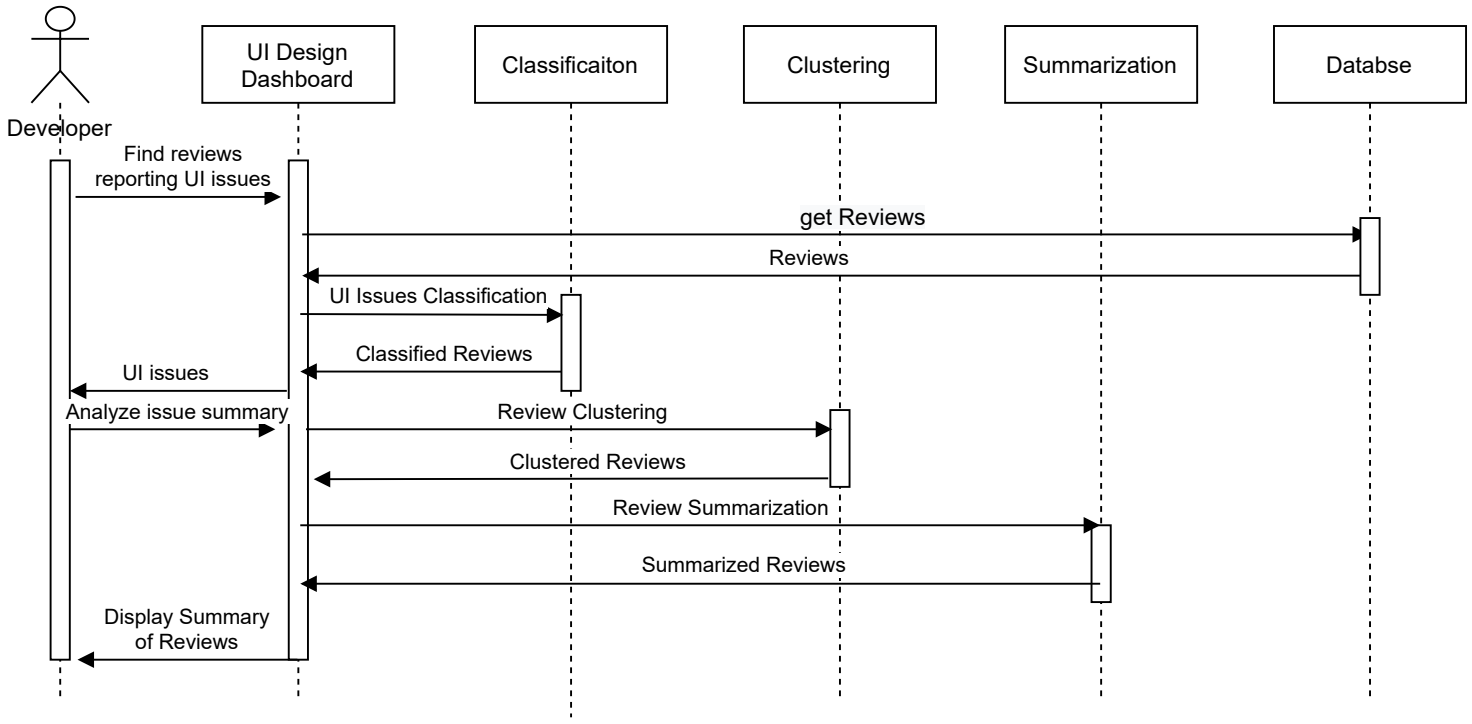
USE CASE VIEW POINT:

Use Case Name: User Interface Design

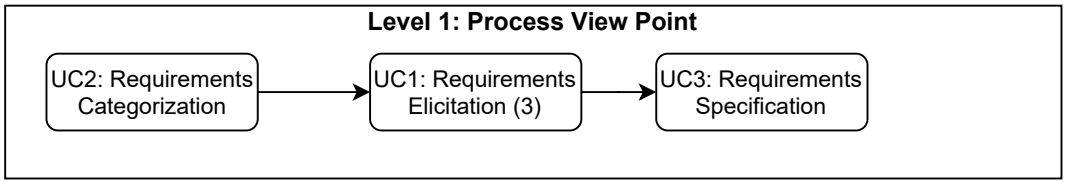
Description: App Developer (AD) use a system to elicit Requirements for User Interface Design

Normal Flow:

1. AD uses a **system find user-reported issues concerning UI.**
2. AD uses a **analyse issue.**
3. The system **display summary of UI issues** reported by app users
4. AD interpret issues and **document requirements** addressing them.



UC6 - Design Rationale Capture



BUSINESS PROCESS VIEW POINT:

UC2: Requirements Categorisation (input: reviews; output: review classified into NFR categories): AD classifies reviews into NFR categories these reviews discuss.

UC1: Elicit Requirements 1 (3) (input: reviews; output: requirements): AD examines the reviews to understand what is the reported issue or what are new requirements. AD interprets what possible change should be made to address the issue.

UC3: Requirements Specificaiton (input: user needs; output: requirements): AD elaborates requirements specification that could serves later as design rationale.

USE CASE VIEW POINT:

Use Case Name: Design Rationale Capture

Description: App Developer (AD) use a system to elicit design rationale (NFR or user justification)

Normal Flow:

1. AD uses system to **classify reviews** by NFR categories review discuss

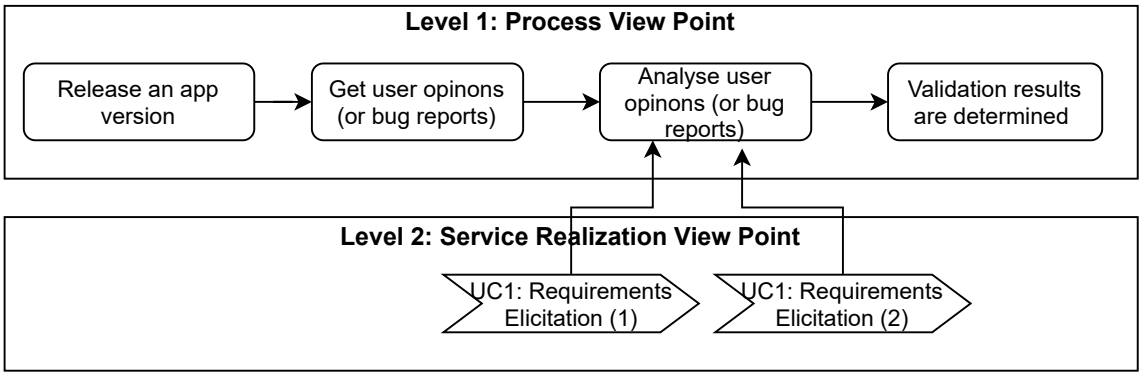
2. AD uses a system to **elicit requirements (design rationale).**

4. AD use a system to **specify requirements (design rationale).**

Alternative Flow:

A2. AD uses system to **identify user justifications.**

UC7 - Validation by Users



BUSINESS PROCESS VIEW POINT:

Release a new app version: AD releases a new app versions (to selected users) to examine users' reactions about developed features and identify (potential) bugs

Get user opinons (or bug reports): AD collects user opinons (feature-specific sentiment) and bug reports to identify potential problems related to newly released app version.

Analyse user opinons (or bugs): AD analyses user opinons and bug repots to understand user acceptance about an updated app product

Validation results are determined: AD determines the results of validation app by users; What problems needs to be address, what changes are successfully accepted.

USE CASE VIEW POINT:

Use Case Name: Validation by users.

Description: App Developer (AD) use a system to validation app with users

Normal Flow:

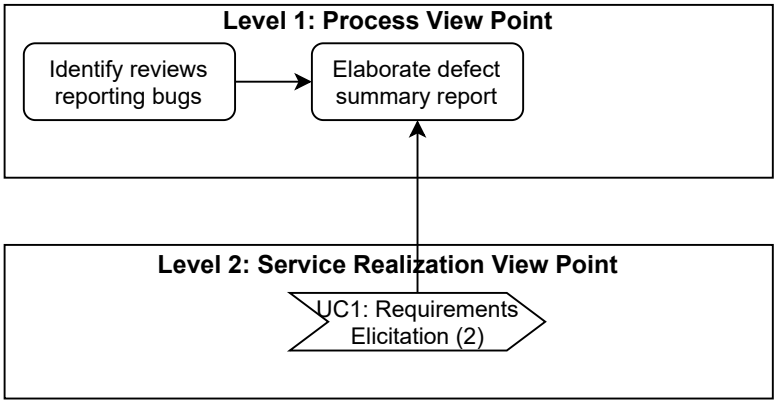
1. AD uses system to **analyse user opinons .**
2. AD uses a system to **determine validation results.**

Alternative Flow:

A1. AD uses system to **analyze bug reports.**

UC8 - Test Documentation (1)

Test Documentation: AD uses system to generate bug reports based on problems reported in reviews. The ad hoc documentation sort of defect report would inform AD about problems user encounter and guide them in debugging and fixing it. Such defect would include information like (release version, date, author, summary, no. users reporting the problem)



BUSINESS PROCESS VIEW POINT:

Identify reviews reporting bugs: AD collect reviews and identifies those reporting problems users encountered with the app.

Elaborate defect summary: AD elaborates test documentation (e.g., defect report) based on reviews reporting problems.

USE CASE VIEW POINT:

Use Case Name: Test documentation

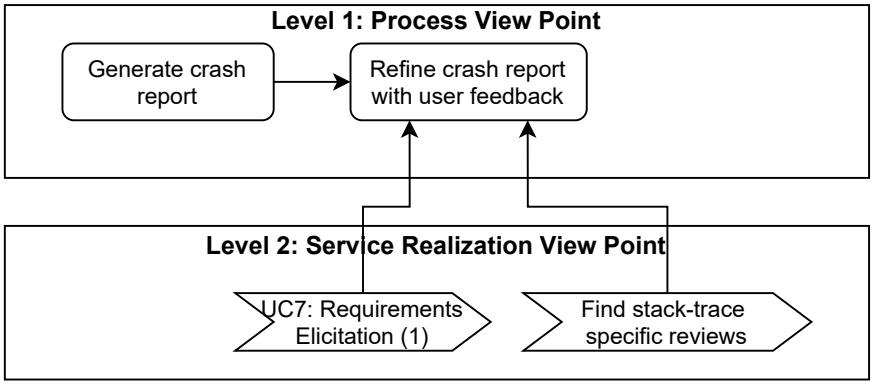
Description: App Developer (AD) use a system to elaborate test documentation.

Normal Flow:

1. AD uses system to **identify bug reports in app reviews**.
2. AD uses a system to **generarete defect summary report.**

UC8 - Test Documentation (2)

Test Documentation: Automating testing tools generates crash reports of stack traces along with the sequence of events that led to the crash. Understanding the problem based on such information may be often overwhelming. In such case, AD may uses a system to link user comments to related stack trace to information tester where to look up for the emerged fault.



BUSINESS PROCESS VIEW POINT:

Generate crash reports: AD generates crash reports using automatic testing tool. The report provide information about reported crashes and related stack traces.

Refine crash report with user feedback: AD find enrich stack traces with related user feedback. Bug reports in the feedback complement stack traces so that AD can more easy understand the crash.

USE CASE VIEW POINT:

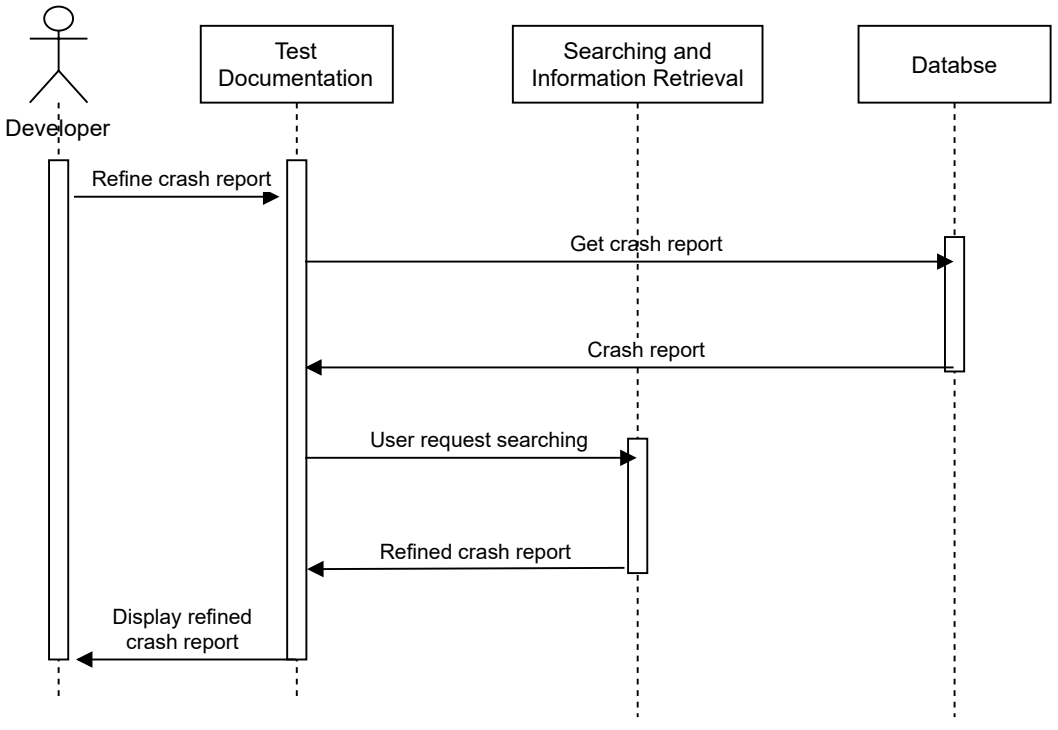
Use Case Name: Test documentation

Description: App Developer (AD) use a system to enrich crash report with user feedback reporting the problem.

Normal Flow:

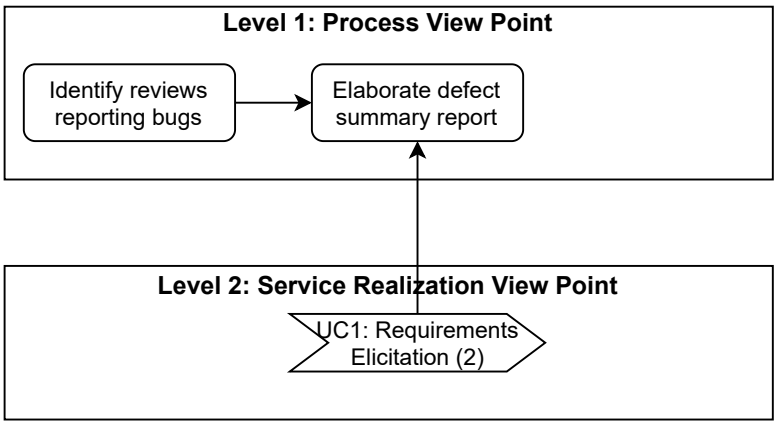
1. AD uses system to **generate crash report**.

2. AD uses a system to **find bug report for specific stack trace.**



UC9 - Test Design (1)

Test design: AD use the system to generate defect reports to determine app behavior and features to be tested. He analyses defects to identify usage scenarios in which an unusual situation emerged or there was lack of workarounds. The information help developers to design test cases capturing exceptions leading to a problem or exercise new alternative scenarios other than initially considered.



BUSINESS PROCESS VIEW POINT:

Identify reviews reporting bugs: AD collect reviews and identifies those reporting problems users encountered with the app.

Elaborate defect summary: AD elaborates test documentation (e.g., defect report) based on reviews reporting problems.

USE CASE VIEW POINT:

Use Case Name: Test Design

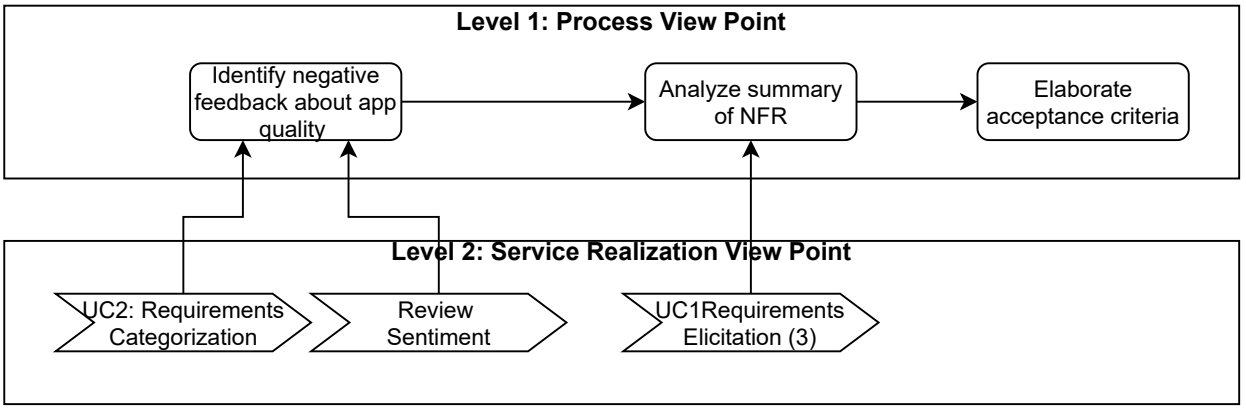
Description: App Developer (AD) use a system for test design.

Normal Flow:

1. AD uses system to **identify bug reports in app reviews**.
2. AD uses a system to **generarete defect summary report.**

UC9 - Test Design (2)

Test design: AD uses a system to generate summaries of negative feedback w.r.t app quality characteristics. The information help AD to specify acceptance criteria an app should hold. For example, AD identify user complaints about performance efficiency indicating a time criteria for certain functions that are expected to finish faster.



BUSINESS PROCESS VIEW POINT:

Identify negative feedback about app quality: AD collect reviews and identifies user feedback providing negative opinions about app quality.

Analyze summary of app qualities: AD analyzes a summary providing information about user opinons (app qualities commented negatively). He examines negative comments about a quality to understand the problem.

Elaborate acceptance criteria: Inspired by the reported probled, AD elaborates an acceptance criteria he will use for designing test case (e.g., https://www.professionalqa.com/acceptance-criteria-vs.-acceptance-test)

USE CASE VIEW POINT:

Use Case Name: Test Design

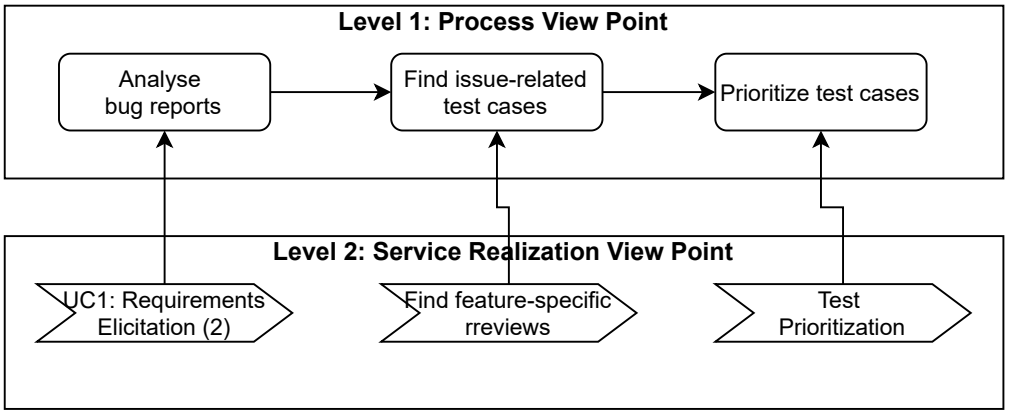
Description: App Developer (AD) use a system for test design.

Normal Flow:

1. AD uses system to **identify negative comments about app qualities.**.
2. AD uses a system to **analyse of user opinions about app qualities.**
3. AD **elaborates acceptance criteria** based on the problem reported w.r.t app qualities.

UC10 - Test Prioritization

Test Prioritisation: User-reported issues have different impact on app rating (a key measure of app success). AD may use the system to prioritise issue-related test cases (scenarios?) based on the level of impact of these issues. Testing core scenarios and ensuring they function correctly could mitigate the impact; Users often revised their rating when their problem has been solved. It could also prevent new negative comments about these issues.



BUSINESS PROCESS VIEW POINT:

Analyze bug reports: AD identify what problems has been reported by users, what usage scenario they concern, and what impact these problems have on app rating.

Find issue-related test cases: AD identify test cases (scenarios?) concerning user-reported issues.

Prioritise test cases: AD prioritise test cases (scenarios?) related to users-reported issues based on the impact of these issue on app rating.

USE CASE VIEW POINT:

Use Case Name: Test Prioritzation

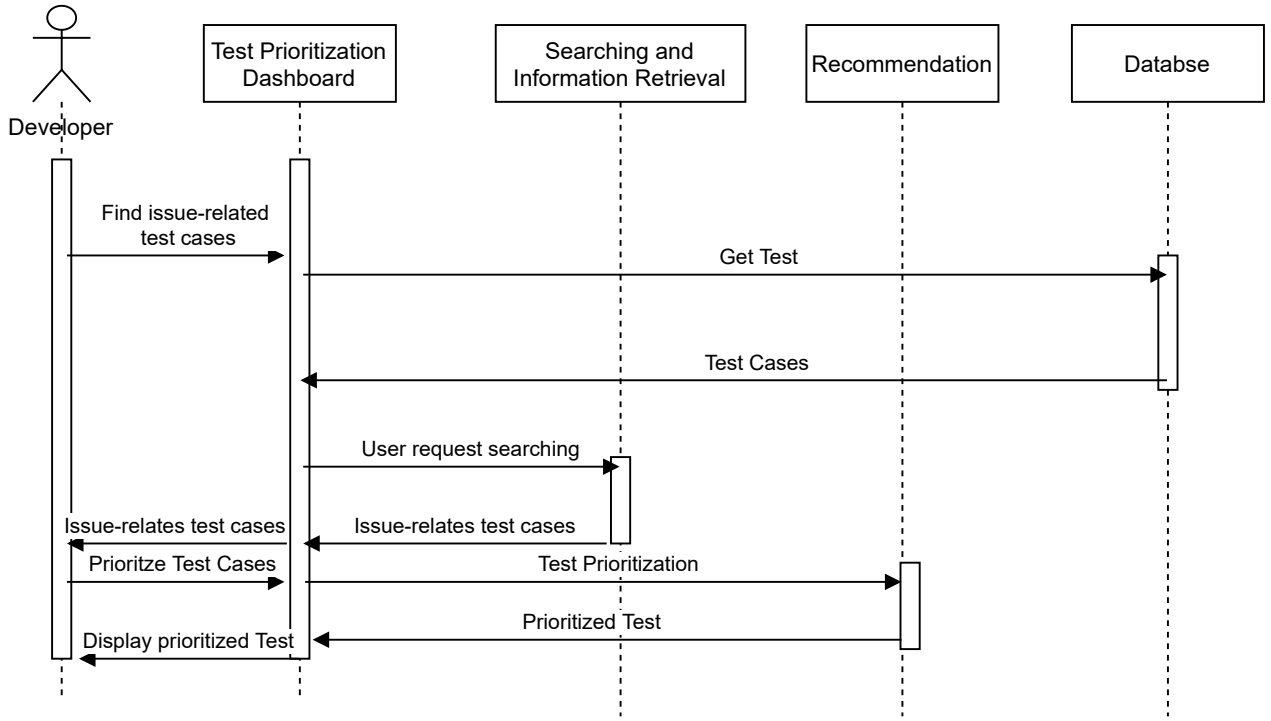
Description: App Developer (AD) use a system test prioritization.

Normal Flow:

1. AD uses system to **analyze bug reports** in app reviews.

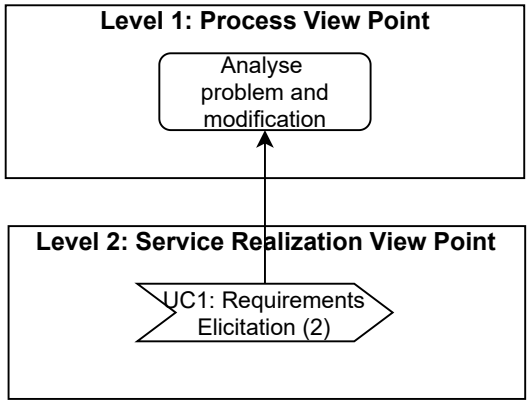
2. AD uses system to **find issue-related test cases (scenarios)**.

3.AD uses system to **prioritise issue-related test cases.**



UC11 - Problem and Modification Analysis

Problem and Modification Analysis: App reviews is a rich source of change requests i.e., problem reports and modification requests. To satisfy users' needs and keep their product competitive in the market, AD can use the system to analyse change requests. It will help them to correct faults as well as perfect app features and their quality.



BUSINESS PROCESS VIEW POINT:

Analyze problem and modification: AD identifies what problems and change requests has been reported by users, what usage scenario they concern, and what impact these problems have on app rating.

USE CASE VIEW POINT:

Use Case Name: Problem and Modification Analysis

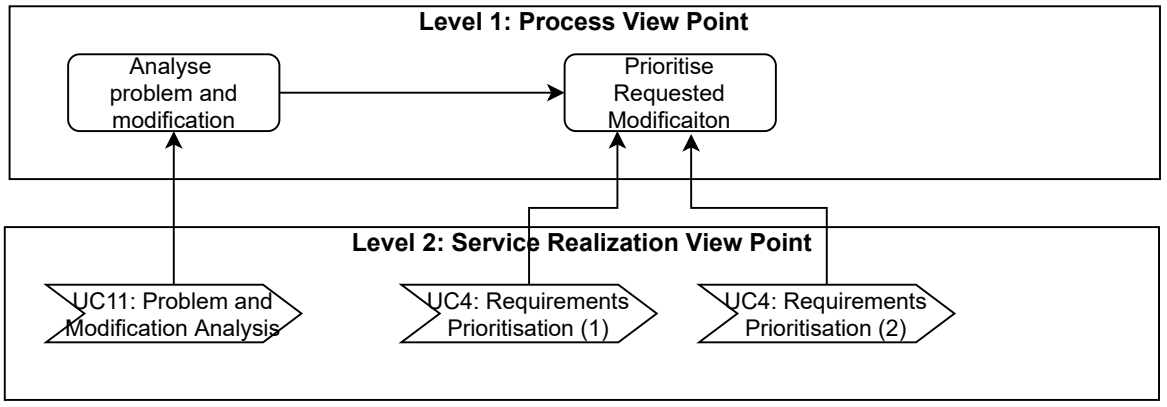
Description: App Developer (AD) use a system to identify problem and modificaitons requests in app reviews.

Normal Flow:

1. AD uses system to **analyze problem and modification** reported in app reviews.

UC12 - Requested Modification Prioritisation

Requested Modification Prioritisation: AD may receive hundreds or even thousands of reviews requesting modifications and reporting problems. Selecting which requests should be addressed in the next release can be a non-trivial task. AD may use a system to prioritise these requests based on their relative users' importance (no. people reporting these requests, how these numbers changed over time, or their impact on app's success (how these request impact overall app rating)



BUSINESS PROCESS VIEW POINT:

Analyze problem and modification: AD identifies what problems and change requests has been reported by users, what usage scenario they concern, and what impact these problems have on app rating.

Prioritise Requested Modification: AD uses a system to prioritise user requests (problem report or modification requests) and decide which should be implemented in the next release.

USE CASE VIEW POINT:

Use Case Name: Requested Modification Prioritisation

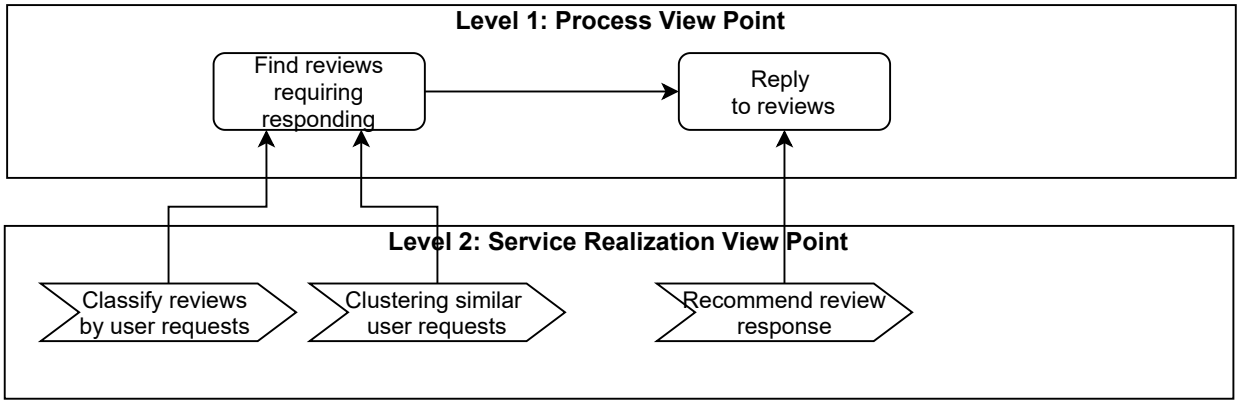
Description: App Developer (AD) use a system to prioritise requested modification for the next release.

Normal Flow:

1. AD uses system to **analyse problem and modification** reported in app reviews.
2. AD uses system to **prioritise requested modifications**.

UC13 - Help Desk

Help desk: AD responds to reviews to answer users' questions, inform about fixing problems or to thank users for their kind remarks. Developers' responses often motivate users to revise their feedback or ratings to be more positive. Finding reviews requiring responding as well as replying to a large number of reviews can be, however, time-consuming. AD can use a system to highlight reviews require responding, and generating automatic replies to these reviews.



BUSINESS PROCESS VIEW POINT:

Find reviews for responding: AD looks for reviews that needs to be answered, those in which users ask for some information, or developer's assistance.

Reply to reviews: AD replies to reviews by providing answer to users' questions or explaining solutions to users' problems. The system recommend responses to reviews based on reviews that had been previously responded (with similar topic)

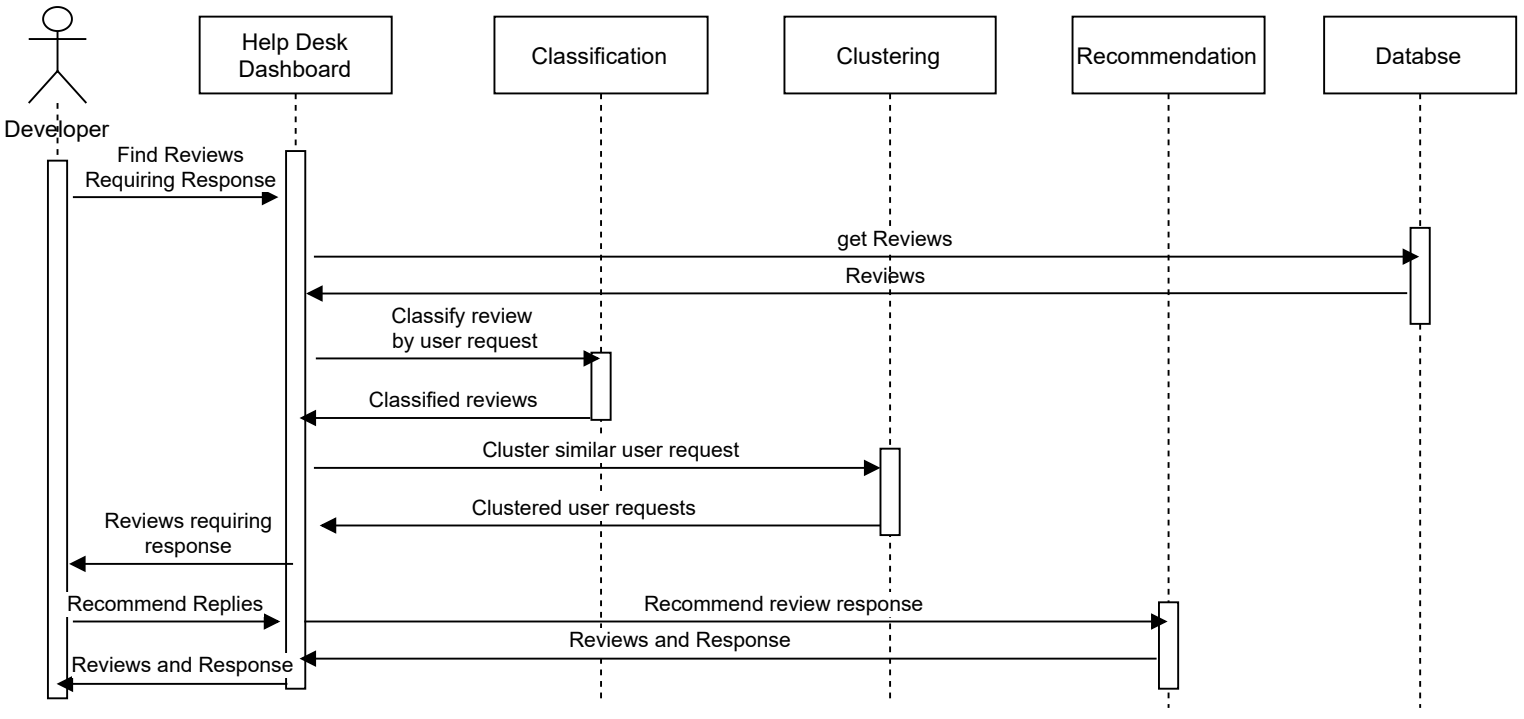
USE CASE VIEW POINT:

Use Case Name: Help Desk

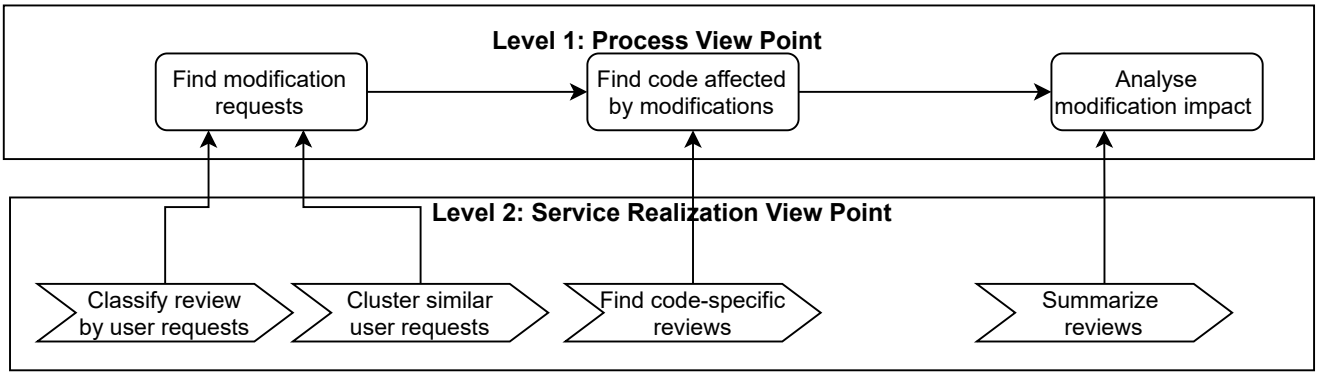
Description: App Developer (AD) use a system to respond to reviews.

Normal Flow:

1. AD uses system to **find reviews requiring responding**.
2. AD uses system to **reply for reviews**.



UC14 - Impact Analysis



BUSINESS PROCESS VIEW POINT:

Find modification requests: AD search for reviews requesting modifications in the app. The system identifies reviews requesting changes and group them based changes these reviews refer to.

Find code affected by modifications: AD find code snippets related to the requested modifications, i.e., source code with/without implemented changes,

Analyse modification impact: AD analyzes what modifications requests have (not) been implemented, quantify no. these requests, localize source code to be changed or implementing these requests.

USE CASE VIEW POINT:

Use Case Name: Impact Analysis

Description: App Developer (AD) use a system analyze the impact of requested modifications.

Normal Flow:

1. AD uses system to **find modification request** in app reviews.
2. AD uses system to **find code affected by modification requests.**
3. AD uses system to **analyse modification impact.**