Supplementary Material

Participant Responses and Thematic Analysis

TABLE I: Participant Reported Systematic Processes for Categorising User Reviews into NFR Categories (Q9)

Part.	Original Response	**Step 1** Feedback Collection	**Step 2** Initial Screening	**Step 3** Classification of User Reviews	**Step 4** Prioritization	**Step 5** Integration into Dev Workflow
P1	*"There is a special team inside the organization, their responsibility to **gather the feedback from the App Store** or from call center they categories this feedback to **multiple categories based on the features** that they have it in the application after that they deliver this requirement to the product team and the product team will **redefined this requirement** to achievable requirement for each quarter."*	✓	✓	✓		
P2	*"Team roles is well defined with the Business analyst and the product owner **gathering user feedback, analyzing** it then sharing it with the **technical team for further analysis (if needed) or for development**."*	✓		✓		✓
P3	*"We are **open bug in jira board and assign it** for the deve team to fix it with **severity and priority** and retest it after fixing."*		✓		✓	✓
P4	*"**Each review or comment being handled as ticket** which has cycle with multiple stages, first stage to **classify**, second stage is the **assignment to find the propoer solution, oppurtunity for improvement**, or response from CS, then **define the required workflow**."*		✓	✓		✓
P5	*"**The flagged reviews** are then **mapped into NFR categories**: Usability (ease of navigation, UI/UX, accessibility) Performance (speed, crashes, responsiveness, battery usage) Security (privacy, data protection, login issues) Other NFRs as applicable (maintainability, compatibility, scalability, etc.) Validation: A second team member verifies the classification to reduce bias or errors. Documentation and Tracking: **The categorized NFRs are documented and linked to Jira or Trello tasks for potential inclusion in future sprints**. Review Meetings: The team **discusses high-frequency NFR issues in sprint planning to prioritize fixes or enhancements**."*		✓	✓	✓	✓
P6	*"We have a Business Analyst who **reviews user feedback and prioritizes** it. During our meetings, he shares the results with the team, and then **creates the important ones as User Stories in JIRA**."*			✓	✓	
P7	*"After **gathering the reviews** we set with the BA to **classify the NFR**."*	✓		✓		

TABLE II: Proposed Process Steps for NFR Classification and Traceability with Release Notes (Q17)

Part.	Original Response	**Step1** Define NFR Categories	**Step2** Collect User Reviews	**Step3** Prepare Reviews	**Step4** Classify Reviews into NFRs	**Step5** Consolidate & Prioritise NFRs	**Step6** Prepare Release Notes	**Step7** Trace NFRs to Releases	**Step8** Human Validation	**Step9** Track & Visualise	**Step10** Iterate & Improve
P1	*"**By defining keywords that often relates to technical issues**, I.e. "it took more time than usual, Feature X wasn't working properly, etc.." and so on. If it's **a time related** issue it can point to **performance issues** in general. If it's about a feature, then through investigation we can detect which service/s is/ are responsible for this feature and **trace its release history**."*	✓			✓			✓			

Part.	Original Response	**Step1** Define NFR Categories	**Step2** Collect User Reviews	**Step3** Prepare Reviews	**Step4** Classify Reviews into NFRs	**Step5** Consolidate & Prioritise NFRs	**Step6** Prepare Release Notes	**Step7** Trace NFRs to Releases	**Step8** Human Validation	**Step9** Track& Visualise	**Step10** Iterate & Improve
P2	*"**Define NFR categories – Use a fixed taxonomy** (e.g., Performance, Security, Usability, Reliability, Compliance) with **definitions and keywords**. 2. **Extract & classify – Parse user reviews into sentences, detect quality attributes (via keywords/ML), and assign each to an NFR category with a confidence score. 3. Canonicalize & prioritize – Merge duplicates, rate severity by frequency, sentiment, and impact. 4. Match to release notes – Normalize release notes into a searchable structure (features, fixes, dates), then match NFRs by component + keywords/embedding similarity to see if they were addressed".*	✓		✓	✓	✓	✓	✓			
P3	*"**Clean reviews, classify into NFR categories, then match them to release notes using keywords, semantics, and date alignment**."*			✓	✓			✓			
P4	*"**Pre-process reviews → classify into NFR categories with ML/NLP. Pre-process release notes → segment and embed. Use semantic similarity to align NFRs with release notes. Build a traceability matrix for monitoring addressed user concerns. Iterate with human feedback for continuous improvement**."*			✓	✓		✓	✓			✓
P5	*"**Classify user reviews into NFR categories using NLP models**, extract key requirements, and **semantically match them with release notes based on content and timing to track how user concerns are addressed**."*				✓			✓			
P6	*"**Collect and Prepare the Data User Reviews**: Gather reviews from app stores, feedback portals, or surveys. **Clean them** (remove duplicates, junk text, and emojis unless sentiment matters). Release Notes: **Collect versioned notes** directly from your repos, changelogs, or published release logs. Tagging Schema: **Decide which NFR categories** you'll use (e.g., performance, security, usability, reliability, maintainability, portability). Don't overcomplicate—5–8 categories is usually manageable. Identify NFR-related Content Automated Pre-Filter: Use a **keyword/phrase filter to flag reviews** that might contain NFR feedback. Example: "slow," "crash," "takes too long" → performance/reliability "confusing," "hard to find" → usability ML Classifier: **Train or fine-tune a text classifier (e.g., using BERT, RoBERTa, or even GPT-5 in zero/few-shot mode)** to recognize NFR vs. non-NFR feedback. Start with labelled examples if you have them; otherwise, label a small set manually. **Manual Validation**: Have humans spot-check 10–20% to make sure you're not wildly off."*	✓	✓	✓	✓		✓		✓		

Part.	Original Response	**Step1** Define NFR Categories	**Step2** Collect User Reviews	**Step3** Prepare Reviews	**Step4** Classify Reviews into NFRs	**Step5** Consolidate & Prioritise NFRs	**Step6** Prepare Release Notes	**Step7** Trace NFRs to Releases	**Step8** Human Validation	**Step9** Track& Visualise	**Step10** Iterate & Improve
P7	*"**User Review Data Collection**: • Extract user reviews from app stores (such as Google Play and the App Store) or other platforms. • **Clean the data** by removing special characters, correcting spelling errors, and performing other data-cleaning steps. Natural Language Processing (NLP): • Linguistic analysis: transform texts into machine-processable formats using techniques such as tokenization and lemmatization. • **Feature extraction: use language models such as BERT or RoBERTa** to extract deep semantic representations from reviews. **Classification of Reviews into Non-Functional Requirement (NFR) Categories**: • Build a deep learning model or use a pre-trained classification model to classify reviews into NFR categories, such as: Performance, Usability, Reliability, Security, Compatibility, Maintainability, Support. **Entity and Keyword Extraction**: • Use Named Entity Recognition (NER) techniques to extract relevant phrases or keywords. **Matching NFRs with Release Notes**: • Use text similarity techniques such as: • **Cosine similarity using TF-IDF or BERT embeddings.** • **Semantic matching using NLP models.** • **Match keywords or functional attributes with release notes** to determine whether the requirements were addressed in the latest release. **Verification and Evaluation**: • Evaluate model accuracy through human review of a sample of the classification and matching results. • **Continuously improve the model using new data and human feedback** ".*		✓	✓	✓			✓	✓		✓
P8	*"With AI getting better, the ideal way to handle user reviews is to **automatically collect them along with information like the user's location and the locale of the review text. The reviews are cleaned up and then classified into non-functional requirement (NFR) categories**, like performance, usability, reliability, security, or maintainability. At the same time, **release notes are analysed** to find changes related to these NFRs. Then, **reviews are matched with the relevant release notes using AI** to see which user concerns have been addressed. Finally, **dashboards can show trends, highlight unresolved issues, and even reveal patterns based on different regions**, helping teams focus on the most important improvements."*		✓	✓	✓		✓	✓		✓	

Part.	Original Response	**Step1** Define NFR Categories	**Step2** Collect User Reviews	**Step3** Prepare Reviews	**Step4** Classify Reviews into NFRs	**Step5** Consolidate & Prioritise NFRs	**Step6** Prepare Release Notes	**Step7** Trace NFRs to Releases	**Step8** Human Validation	**Step9** Track& Visualise	**Step10** Iterate & Improve
P9	*"The ideal process for classifying user reviews into NFR categories starts with using tools to* ***automatically categorize reviews into appropriate categories*** *like performance, usability, or security. Then, a* ***specialist verifies the classifications to ensure accuracy***. *After that, these* ***requirements are compared with the release notes to see if and how they have been addressed in updates***. *Finally, the process should be* ***continuously improved based on feedback and new trends from reviews and app updates***."*				✓			✓	✓		✓
P10	*"****Leveraging ai tools to self classify the requirements*** *to make it easier for the team to work on and to* ***match with each release***."*				✓			✓			
P11	*"****Using an automated tool to classify user reviews to already predefined NFRs Categories, with (Others) as non-existent Category, then reviewing them after each release manually and verifying them with the release notes***."*	✓			✓			✓	✓		
P12	*"An ideal tool would* ***sort user reviews into NFR categories, link them*** *to release notes, and use them as input for each release to check if the release meets those NFRs."*				✓			✓			
P13	*"Very high level process:* ***1. Define categories 2. Pre-process reviews 3. Classify 4. Extract key issues 5. Parse release notes 6. Match 7. Validate***."*	✓		✓	✓		✓	✓	✓		
P14	*"An ideal process involves* ***automated NLP classification of user reviews into NFRs, manual validation for accuracy, mapping to release notes, documenting in a tracking system, and conducting periodic team reviews for continuous improvement***."*				✓			✓	✓	✓	✓
P15	***"Classify reviews into NFRs, tag release notes, match overlaps, track progress, highlight addressed issues, and report remaining gaps***."*				✓		✓	✓		✓	
P16	*"****Compare user-requested NFRs against addressed NFRs to find unmet requirements. Create dashboards showing most frequent NFR categories, coverage by releases, and outstanding requests***."*					✓		✓		✓	
P17	*"****Data Preparation, Classify Reviews into NFR Categories, Extract NFR Statements, Match NFRs with Release Notes, Validate & Iterate and Optional Enhancements***."*			✓	✓			✓	✓		✓
P18	*"The* ***Business Analyst collects and cleans user reviews, classifies non-functional feedback*** *(performance, security, usability), and* ***prioritizes them. QA and Dev validate classifications*** *and plan fixes. After each release, the* ***team matches NFRs with release notes to see if issues are resolved. Continuous monitoring ensures feedback is tracked and the process improved***."*		✓	✓	✓	✓		✓	✓	✓	✓

TABLE III: Recommended Tool Functionalities for NFR Classification and Traceability to Release Notes(Q18)

Part.	Original Response	**F1** Collect Reviews	**F2** Pre-process Reviews	**F3** NFR Classification	**F4** NFR Prioritisation	**F5** NFR Traceability	**F6** Human Validation	**F7** Visualisation & Dashboard	**F8** Continuous Improvement
P1	*"Im not an expert in this area, but if I were to outline an ideal process, it might look like this: Since user reviews come from third-party platforms, the first step would be to extract as much data as possible. Next, an AI service could filter out unhelpful reviews that aren't related to app functionality (such: comments about company policy or pricing). Another AI tool could then classify the remaining reviews into NFR categories and rank them based on relevance or impact, and group related reviews together. I believe would make it much easier and faster for whoever handles app store reviews (whether it's the development team or a dedicated customer experience team)".*	✓	✓	✓	✓				
P2	*"An ideal tool for classifying user reviews into NFR categories should have a few key features. First, it should use natural language processing (NLP) to automatically analyze and classify reviews into categories like performance, usability, and security. It should also have a validation step where the tool can flag uncertain classifications for human review. The tool should be able to compare the extracted NFRs with release notes, identifying if these requirements are addressed in app updates. Additionally, it should allow for continuous learning, improving its classification accuracy based on user feedback and evolving trends. The tool should be easy to use, with a simple interface and automated workflows to streamline the process."*			✓		✓	✓	✓	✓
P3	"**Data ingestion & cleaning → tools like Apache NiFi, Airbyte, or custom scripts. NFR classification → fine-tuned LLM models (OpenAI, Hugging Face) with your NFR taxonomy**."	✓	✓	✓					
P4	*"**Identifying similar complaints, reviews or suggestions, and then bundling these to segments**. Each segment will contain different type of informations about this review, the person, their details. And finally when all reviews collected, it the **tool can provide summaries based on collected user reviews**."*			✓	✓			✓	
P5	*"Actually, I haven't used any tools for this before, but an ideal solution could be built from scratch. Doing so would provide full control and customization over the entire process, including **how frequently user reviews are collected, how they are classified into different NFR categories**, and how release notes are processed and matched with user feedback. Building it in-house would allow tailoring the system to specific product needs, handling multiple locales, adjusting classification rules, and **integrating directly with existing workflows or dashboards**, ensuring that the insights are both accurate and actionable for the team".*	✓		✓				✓	
P6	*"An ideal tool for classifying user reviews into NFR categories and matching them with release notes would integrate **NLP-based review classification, requirement extraction, and semantic similarity matching with version-aware alignment, providing insights into how user concerns map to software updates**."*			✓		✓		✓	
P7	*"I would say a **tool that all user review sources (app store reviews, tickets to support team, comments in social media, etc..) pour into, keeps register of these reviews, classify them, and keep them up to date when issues are resolved**."*	✓		✓		✓			
P8	*"An ideal process involves **automated NLP classification of user reviews into NFRs, manual validation for accuracy, mapping to release notes**, documenting in a tracking system, and conducting **periodic team reviews for continuous improvement**."*			✓		✓	✓		

Part.	Original Response	F1 Collect Reviews	F2 Pre-process Reviews	F3 NFR Classification	F4 NFR Prioritisation	F5 NFR Traceability	F6 Human Validation	F7 Visualisation & Dashboard	F8 Continuous Improvement
P9	*"An **ideal tool ingests reviews and release notes, extracts NFRs, classifies them into categories** (e.g., performance, reliability), **auto-tags release notes, matches overlaps by version/date/keywords**, and shows dashboards of addressed vs open gaps."*	✓		✓		✓		✓	✓
P10	*"An AI-based platform that **provides multi-source inputs** and incorporates an **advanced NLP engine, including a classification model trained on NFR data, an intelligent matching system, and API integration**. Such a platform can be implemented either as a custom solution built using Python, leveraging libraries such as spaCy and Transformers with **Streamlit for the user interface**, or by adopting existing tools such as MonkeyLearn with additional customization, or AWS Comprehend combined with SageMaker for professional-grade text classification and analysis."*	✓		✓		✓		✓	
P11	*"A tool that **cleans reviews, classifies them into NFR categories, and matches them to release notes** by meaning, keywords, and dates."*	✓		✓		✓			
P12	*"We may use the newly **emerged AI tools, like chat GPT and Copilot**, but it needs to be trained first, and **review after the results**."*			✓			✓		
P13	*"**CLAP** paper, crowd listener for released planning)."*			✓					
P14	*"**RoseMatcher**."*					✓			
P15	*"**Unified Ingestion, Smart NFR Classification Actionable NFR Extraction,Release Note Matching,Feedback & Verification Loop and Visualization & Reporting**."*	✓		✓		✓	✓	✓	