1 Preface

This document presents the requirements of the solution and is to be read by the stakeholders, the development team, and the assessor (Dr. James Archbold). It details and justifies the nature of the solution and its features, project development principles, group management, project and system evolution, and project constraints.

2 Introduction

The client requires a live-feedback system where event participants can submit real-time feedback to the event host. Hosts of such events are provided with sentiment-based analytics and live feedback. See the project's full brief and specification here [1].

The client's existing systems collect feedback after an event; this leads to both a sub-standard participant experience and inadequate feedback. The live feedback paradigm allows hosts to rectify event-related issues in real-time, improving event quality and participant experience. With such a system, feedback now has a direct influence over event experience. Consequently, participants are inclined to provide more feedback. User-created templates will provide a framework for event feedback, increasing the applicability and quality of feedback.

3 Glossary

Template:	A framework for feedback, structured with form components.
Generic event:	Common real-world events for which the solution is likely to be used.
Sentiment:	An estimate of the overall opinion(s) of a particular instance of feedback.
Mood:	An estimate of the overall opinion(s) of a group of feedback instances.
MoSCoW:	An order of requirement implementation priorities: Must, Could, Should, Won't.
C/D:	Customer / Developer Facing Requirement
GDPR:	General Data Protection Regulation
NLP:	Natural Language Processing

4 Requirements Validation

Each requirement was subject to a five-part test, with collated requirements subject to a four-part test. These assessments ensure requirement quality and are derived from common industry standards outlined in 'Writing Quality Requirements' [2].

Testing against requirements considers correctness, feasibility, necessity, ambiguity, and verifiability. Every requirement has passed this test. Testing requirements as a whole evaluates prioritisation, consistency, modifiability, and traceability. The Moscow system is used to prioritise requirements. The requirements have been checked and found to be internally consistent. All requirements are subject to evolution and revisions; changes will be noted in the final report. Every requirement has had its source analysed when checking for necessity, and requirements will be referenced in subsequent documents to ensure traceability. Given these requirements have passed internal validation, they provide a sound basis for solution design and development.

5 User Requirements Design

1. Users (Hosts and Participants)
   - 1.C - Users must be able to access the solution from their mobile devices and personal computers.
   - 1.D - There must be a web-app, and native Android and iOS mobile apps, on which users can access the solution.
2. Hosts
   - 1.C - For host authentication, host accounts must be accessed with a private word list (generated at host creation). This is traceable to the customer's interest in withstanding hostile acts and influences [1].
   - 1.D - The solution must generate a unique, host account specific word list upon host creation.
3. Participants
   - 1.C - Participants must be able to submit feedback against the event they are attending (including via host-defined templates). This is traceable to specification point 3 [1].
   - 1.D - The solution must provide the participant with a template to submit feedback.
   - 2.C - Participants must be able to join events with an event-specific access code. This ensures only invited participants can join events. This is traceable to specification point 2 [1].
   - 2.D - The solution must implement an event-specific access token system.
   - 3.C - Participants should have the option to submit a name when first joining an event, prior to feedback-submission.

3.D - The solution should offer a name input option upon joining an event, if a name is given it should be stored.

4.C - For each feedback instance, all participants should be able to submit feedback anonymously to the host. This is necessary as it is traceable to point 4 of the specification [1].

4.D - Feedback submitted anonymously should not store the name of the submitting participant. This is verifiable by user acceptance testing (Test if host users trace anonymous feedback).

6 System Requirements Specification

Non-Functional Requirements

4. The solution should be reliable and secure.

   1.C - The solution should consistently perform as expected. This is traceable to the customers interest in reliability [1].

   1.D - The solution should be able to withstand and recover from minor component failure.

   2.C - The solution should be able to withstand hostile acts and influences. This is traceable to the customers interest in security [1].

   2.D - All inputs must be sanitised, to prevent code injection (attacks). Database interactions must be parameterised (with values bound against fixed types) to avoid SQL injection. Verifiable using test data in unit testing.

   3.C - The solution should be proven to consistently perform as expected.

   3.D - The reliability of the solution should be validated through user acceptance testing.

5. The solution should be efficient, performant, and scalable.

   1.C - The solution should remain responsive.

   1.D - User experience should not be impacted by system latency. The solution should scale to satisfy a dynamic user-population. The performance of the solution should be validated using time complexity analysis within system testing.

   2.C - The solution should be hardware independent and portable across hardware architectures.

   2.D - The software should not be designed for any one particular hardware architecture.

6. The solution should be usable and accessible.

   1.C - The solution should be intuitive, require little to no training, and be suitable for users who are inexperienced with technology. This is traceable to project brief non-functional point 1 [1].

   1.D - The solution should explain any complex functions, and have a clear and intuitive interface. This is verifiable by user acceptance testing.

   2.C - The solution should be available for use on every working day of the year. This is necessary to fulfill the customers desire for an accessible service.

   2.D - The solution should be able to run continuously for five days. Verifiable through system testing.

7. The solution should be maintainable.

   1.C - Code should be well organised, and be fully and concisely commented.

   1.D - Code within the solution's code-base should be fully and concisely commented, and should conform to code etiquette standards and language style guides. Developers should be using static code analysis tools (linters).

   2.C - Solution components should be well and concisely documented. Documentation should be client accessible.

   2.D - Component development should be well documented in each scrum cycle.

   3.C - Code-base and project development should be tracked to improve maintainability, as future developers can refer to such logs and gain valuable insight.

   3.D - Solution development should make use of version control software to track changes.

8. The solution must comply with all relevant legal regulations.

   1.C - The solution must comply with applicable data protection and privacy laws.

   1.D - The solution must satisfy GDPR [3], when operating in the UK and EU.

   2.C - The solution must comply with applicable copyright laws.

   2.D - Any external frameworks and libraries used by the solution must have a valid license. The solution must reference any externally sourced code.

   3.C - The solution must comply with applicable accessibility regulations.

   3.D - The solution must comply with the 2018 Accessibility Regulations and the 2010 Equality Act, when operating in the UK [4].

9. Research

   1.C - Effective and thorough research should be employed when designing the solution, and elsewhere when relevant. This is traceable to the customers desire for accessibility and user satisfaction [1].

   1.D - Research should use the following techniques: rapid prototyping, existing product research. Research could use the following techniques: potential user consultation, potential user questionnaires.

Primary research areas include the graphical user interface, generic event types, default templates, template design, and feature explanation.

Functional Requirements

10. Events
    - 1.C - Hosts must be able to create events, and specify the event's details: title, description, template in-use. This is traceable to specification point 1 [1].
    - 1.D - The system must only allow hosts to create events, and provide fine-tuned control over events.
    - 2.C - Hosts should be able to select and edit default events (based on generic events) when creating an event.
    - 2.D - The system should provide an event-creation interface, and a predefined set of editable, generic events.

11. Templates
    - 1.C - Hosts should be able to create usable templates by choosing form elements. This is traceable to specification point 8 [1].
    - 1.D - The system should implement template-generation functionality while ensuring template usability. This is verifiable by user acceptance testing.
    - 2.C - Hosts should be able to select and edit default templates when creating an event. This is traceable to specification point 8 [1].
    - 2.D - The template-generation functionality should have a predefined set of editable, generic templates.
    - 3.C - Hosts should be able to create templates that allow participants to include the general feeling and context of their feedback. This is traceable to specification point 8 [1].
    - 3.D - The system should implement form elements as described in 11.3.C into template-generation functionality.

12. Sentiment and Mood Calculation
    - 1.C - For each instance of feedback, the solution must derive an individual sentiment. This is traceable to specification points 5 and 7 [1].
    - 1.D - Sentiment is calculated to be the weighted mean of scores taken from both NLP and responses to queries with pre-defined inputs. Weighting depends on the template. Verifiable using test data within unit testing.
    - 2.C - The solution must derive a common sentiment (mood) from any period of time across an events duration. This is traceable to specification point 7 [1].
    - 2.D - For any period of time across an events duration, the solution must be able to calculate the mean of the mean sentiments of every feedback instance with a common author. Verifiable using test data within unit testing.

13. Feedback Data
    - 1.C - Hosts must be able to view any instance of (non-muted) feedback, and its sentiment, in real-time. This is traceable to specification point 6 [1].
    - 1.D - Hosts must be able to view all instances of (non-muted) feedback (and their individual sentiments) stored against their ongoing events. Updates to event feedback will occur during feedback analysis.
    - 2.C - Hosts must be able to view the mood of all feedback stored against their ongoing events. This includes real-time mood, average mood, average mood over time, and mood over time. Mood metrics are likely to evolve during the project.
    - 2.D - The system must send feedback and mood-related data to the host in real-time.
    - 3.C - Hosts should be able to view mood metrics (see 13.2.C) relating to past events.
    - 3.D - The mood metrics of an event should be host-accessible and persist after event completion.
    - 4.C - Hosts could be able to view repeated and popular feedback against their past and current events.
    - 4.D - Repeated requests against a host could be stored after event completion, and be accessible to said host.

14. Misuse Protection
    - 1.C - Hosts could be able to mute any participant within their running event; to prevent abuse by participants.
    - 1.D - Hosts could have the option to mute the sender of any given feedback instance.
    - 2.C - Participants could be prevented from automating feedback submissions (e.g. via CSRF [5] attacks). This is to prevent malicious behaviour.
    - 2.D - The system could allocate unique tokens for feedback instances (these would also identify participants). The system could only accept one feedback instance per token.
    - 3.C - At event creation, hosts could be able to blacklist words and phrases from feedback to prevent abuse.
    - 3.D - Feedback containing blacklisted words could be discarded and not considered within sentiment analysis. The solution could implement a host-configurable feedback-text exclusion system. This exclusion system won't be dynamic: editable by hosts after event creation.
    - 4.C - Hosts could be able to implement spam filters at event creation; to prevent malicious user attacks.
    - 4.D - The system could implement host-specified rate limiting against said host's event. The system won't implement group spam filtering.

5.C - Frequently mentioned feedback against an event won't be filtered out from the host's view. This avoids redundancy: once a host is aware of feedback, said feedback is no longer relevant.
5.D - The solution won't implement a system that excludes repeated feedback.

7 System Evolution

Assumptions
The solution assumes the client has the necessary infrastructure to deploy the system and its services to their employees. This deployment need not be restricted to the client's internal network, as event access is controlled via private access codes. It is assumed the client's infrastructure is capable of storing all relevant past, current, and future system data. Client infrastructure must also be reliable: the system must be accessible on every working day of the year. An assumption is that system users have the hardware that satisfies 'Microsoft defined industry standard web application requirements' [6] (not listed here for readability). They should also have one of the following web browsers: Google Chrome `v.88`, Mozilla Firefox `v.84`, Microsoft Edge `v.88`, Opera `v.72`, or Safari `v.14`. Users should have both an internet connection that meets 'FCC international standards' [7] (minimum `1Mbps` up/down), and some experience with computer software and web applications. The system assumes hosts can distribute invite codes to participants and that participant feedback accurately represents their actual event experience.

Changing User Needs
The system front-end is subject to review for every 'major software update' for each of the aforementioned browsers (signified by a change in a version or release number [8]) to check browser compatibility. These reviews must also ensure the user-interface conforms with modern design practices. The solution software will be clearly documented as will the design process and project history. These will allow future developers to effectively evolve the solution as customer needs evolve.

Hardware Evolution
Web applications are hardware independent, meaning desktop hardware evolution is not a major concern. On mobile devices the app must be checked with every 'major software update' to the OS, to ensure it is compatible with the current OS. The proposed system's back-end technologies will be portable across hardware architectures, ensuring the solution can be deployed to the clients hardware (assuming it meets the criteria outlined in assumptions).

8 Group Management

Team Roles

Name	Primary Role	Secondary Role	Development Lead
Thomas Cowley	Project Manager	Developer	Testing
Alexander Price	Business Analyst	Developer	Sentiment Analysis
George Denny	Developer		Front-end, UI and UX
Moustafa Eladawy	Developer		Back-end Development
Yang Tang	Developer		Mobile Development

Role allocation within the group was mutually agreed and chosen to best utilise the available human capital. Naturally, primary roles are prioritised over those that are secondary. Development began in Week 4; development meetings will be held three times each week, at 10 am to noon, on Mondays, Wednesdays, and Fridays. Before development, meetings were held bi-weekly on Tuesdays and Thursdays, both at 10 am to noon. Details relating to methodology and logistics can be found in our Planning and Design Document.

References

[1] Deutsche Bank AG and The University of Warwick DCS. *CS261 (Software Engineering): Group Software Development Project*. (Accessed 1st Feburary, 2021). URL: `https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs261/project`.

[2] Karl E. Wiegers. *Writing Quality Requirements*. (Accessed 1st Feburary, 2021). URL: `https://www.processimpact.com/articles/qualreqs.pdf`.

[3] The European Parliament and the Council of the European Union. 'GDPR: General Data Protection Regulation'. In: *Official Journal of the European Union* (). (Accessed 2nd Feburary, 2021). URL: `https://gdpr-info.eu/`.

[4] GOV.UK. *Making your service accessible*. (Accessed 2nd Feburary, 2021). URL: `https://www.gov.uk/service-manual/helping-people-to-use-your-service/making-your-service-accessible-an-introduction`.

[5] OWASP. *OWASP: Cross-Site Request Forgery*. (Accessed 2nd Feburary, 2021). URL: `https://owasp.org/www-community/attacks/csrf`.

[6] Microsoft. *Web Application Requirements*. (Accessed 3rd Feburary, 2021). URL: `https://docs.microsoft.com/en-us/power-platform/admin/web-application-requirements`.

[7] US Government FCC. *Broadband speed guide*. (Accessed 2nd Feburary, 2021). URL: `https://www.fcc.gov/consumers/guides/broadband-speed-guide`.

[8] Law Insider. *Definition of Major Upgrade*. (Accessed 1st Feburary, 2021). URL: `https://www.lawinsider.com/dictionary/major-upgrade`.