Planning and Design Document

CS261 Group 7
Vita Bush, Dominika Daraz, Nkosi Khumalo, Megan Livermore, Joe Moore, Caleb Venable

Introduction

This document details the plan and design for a live event feedback system for use in meetings, group projects and other events at Deutsche Bank, based on the project specification[1]. The system outlined here will provide an easy-to-use interface through which hosts can create meetings, invite attendees, and receive to-the-minute updates on the general mood and engagement of attendees. The system has practical applications in many scenarios, since receiving live feedback enables a meeting organiser to respond in real time. The use of semantic analysis algorithms to detect the mood of textual feedback given allows the host to gain a snapshot of how the attendees are feeling at any given time, and identify trends.

Technical description

In this section we specify and justify the technical decisions made in order to develop our solution in the Requirements Analysis document. The notation [R.x.x] is used to refer to specific requirements.

The Event Feedback System will be a web-based application (web app) so the system is light and versatile, and to allow compatibility with all devices and automatic user familiarity. The interface will be created using React[2], a Facebook-developed JavaScript library for building user interfaces. Our solution relies on an interactive interface that must be able to respond to prompts from a server and display constantly changing data. React was created with this functionality in mind, and with the use of React Hooks[3] our interface can quickly respond to input. This is the main reason we have chosen to use this technology rather than a standard HTML-CSS web app, which would require a more complex implementation to achieve such interactivity. React is widely used in industry, and has a large technical support community online and a plurality of 3rd party libraries available.

The back-end server will be built using Flask[4], a micro web framework written in Python, chosen due to its robustness, as well as the wide range of available libraries. Its robustness stems from Flask's fast debugger; the lightweight and modular design also makes it easy to adapt it for our use. Furthermore, it requires minimal overhead, is fast to develop with, and has strong integration with Pytest[5], a Python integration for unit testing code. We will be using Flask to create a Representational State Transfer (REST) API[6] that will allow the front-end to access back-end functionality through HTTP requests to specific endpoints that will respond with the appropriate data (see [R.1.1]).

The Python Flask backend will be implemented in conjunction with an SQLite[7] database, chosen due to its natural compatibility with Python. The application will constantly handle data from users, and use Create, Read, Update and Delete (CRUD) commands to modify the database (see Figure 1) , as well as analyse and sort data to provide summary feedback results.

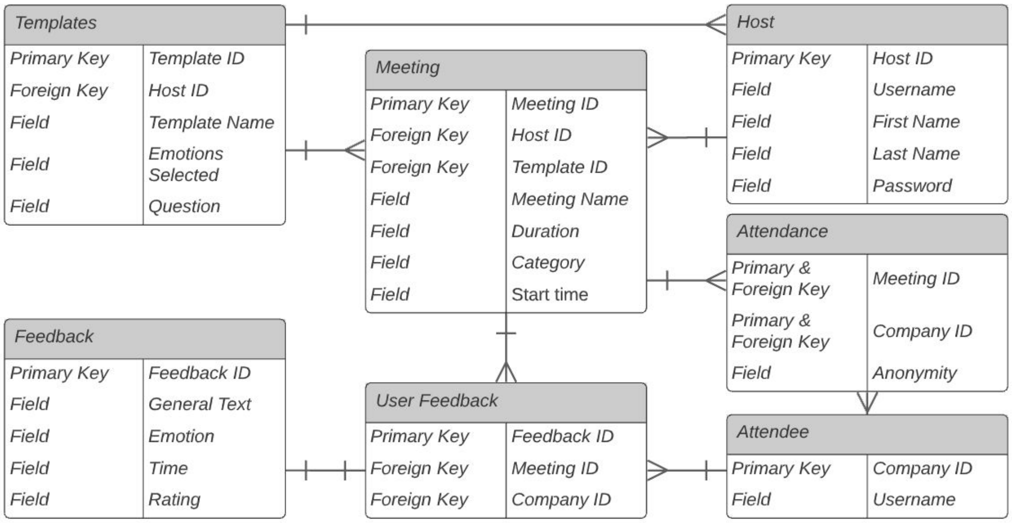

Figure 1: The planned relations and fields in the SQLite Database, showing all the components necessary to accurately represent the system information. This database is in 3rd Normal Form to avoid redundancies in storage.

We will be using an Object-Oriented approach, with objects for each type of user inheriting the general user class, each meeting and the template it corresponds to, and one object per piece of feedback given. Figure 2 depicts our planned system structure.

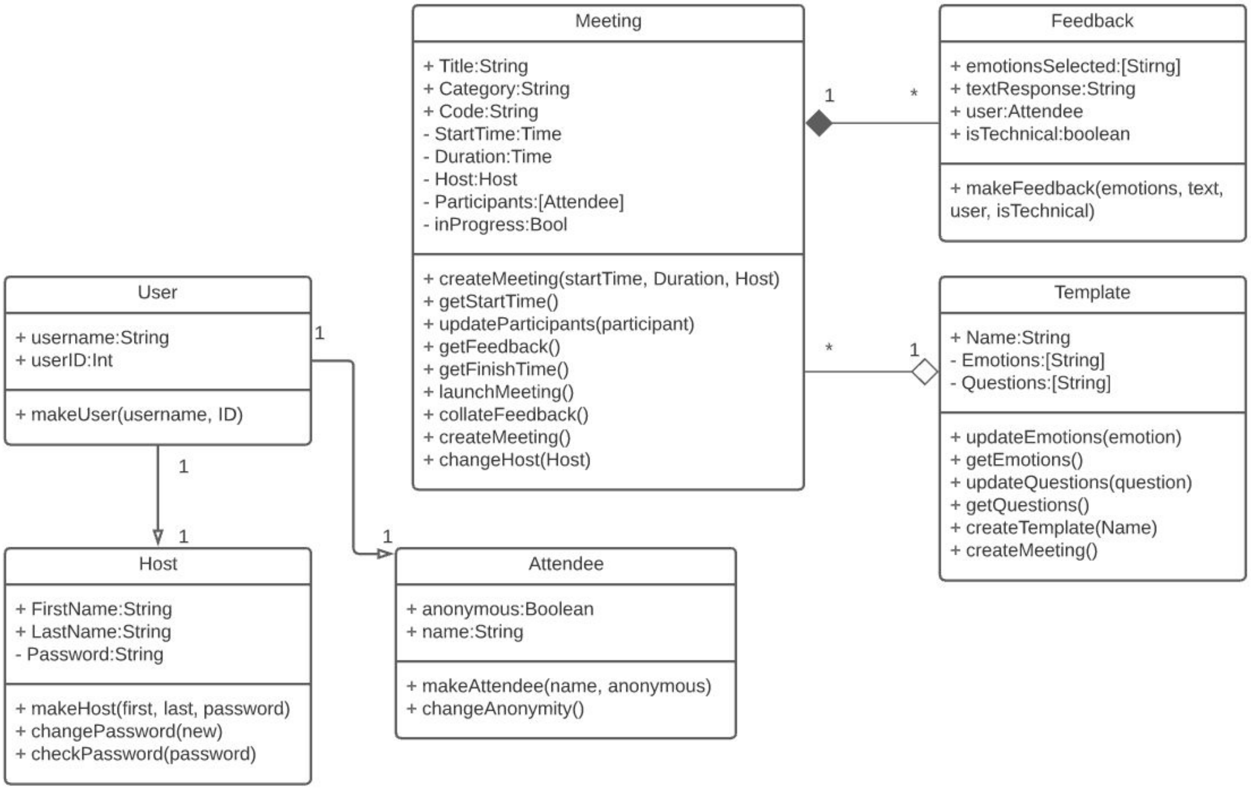

Figure 2: The proposed class diagram to be used for instantiating objects in the system, with composition, aggregation and inheritance relations between objects shown.

Since security need not be thoroughly considered, the system implemented will provide a simple user identification agent without performing any complex or substantial checks or handling of data. As such, passwords will be stored with a simple hash, but a more rigorous form of security could be implemented in the future. Hosts will be assigned a unique ID to connect them with a meeting they host, but attendees will be assigned an anonymous ID which will only be connected to their name if they choose not to remain anonymous (see [R.7]).

Hosts will be able to create an event, with meetings optionally grouped by category and displayed in chronological order (see [R.2],[R.3]). Once a meeting is ready to start, the host can generate and share a QR code to create a unique link to their meeting, and an alphanumeric meeting code. Both will direct attendees to the appropriate endpoint to join the meeting, e.g "https://appname/meeting/[meetingid]". We will use a QR code creation library[8] to facilitate this (see [R.5] and [R.6]).

A host can create and store a template (see [R.4]) to be used for any meeting, or choose pre-designed templates. Each template is a number of host-selected emotions, a free text box, and up to 3 questions asked at the end of the session.

Once an attendee provides feedback through the web interface, this data is sent to the server via a HTTP request to the appropriate endpoint, e.g. "https://appname/feedback/meetingid", with the feedback data in the body of the request. This information is stored in the database and also sent to the host through a real-time connection facilitated by HTTP Sockets via the socket.io library[9]. Once the host has access to this data, appropriate visualisations are created. React's versatile components system allows creation of custom visualisations, in conjunction with the Recharts library[10] (see [R.8], [R.9] and [R.10]).

To analyse and summarise data (see [R.10]), an SQL query will fetch data stored in the database. A string splitter will be applied to separate feedback into sections. Here, Sentiment Analysis (SA) of text messages will occur. SA is the process whereby an algorithm can draw meaning from a sentence or paragraph. Lexical semantics plays an important role in SA, allowing machines to understand relationships between words, including synonyms, antonyms, homophones and homonyms. The process of Word Sense Disambiguation[11] is applied to determine the meaning of a word in its given context. This can be achieved through the use of dictionary and knowledge-based methods, or supervised methods which require less resources but are trained on sense-annotated corpora which are a set of accepted outputs based on given inputs used for training algorithms. Finally, Relationship Extraction[12] can be used to detect the semantic relationships present in a text. It is more likely we will find success using a keyword extraction approach rather than one focused on entities, since the key things we are looking for are pieces of feedback rather than names.

Error handling will need to take place to allow the program to run in circumstantial edge cases. The use of try-catch blocks can be implemented to prevent the program from crashing. More importantly the use of different error types, to display different error messages based on the error, will be critical for displaying useful error messages. We will make use of different error types in conjunction with Python's try-except block for example, `except SyntaxError:` to catch syntax errors or the use of brackets, `except (TypeError, ZeroDivisionError):` to catch multiple types. This will allows a user to understand what they did to cause an error, and avoid repeating the mistake (see [R.11]).

User Interface design

Figures 3 through 7 depict use cases of the Event Feedback System, and sequences of typical user events, for potential attendee and host users.

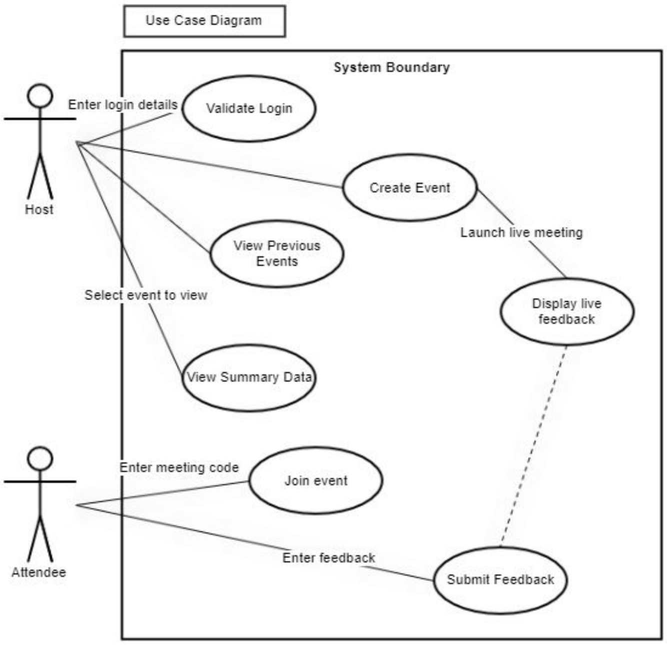

Figure 3: Use case diagram for host and attendee, showing how users interact with the system.

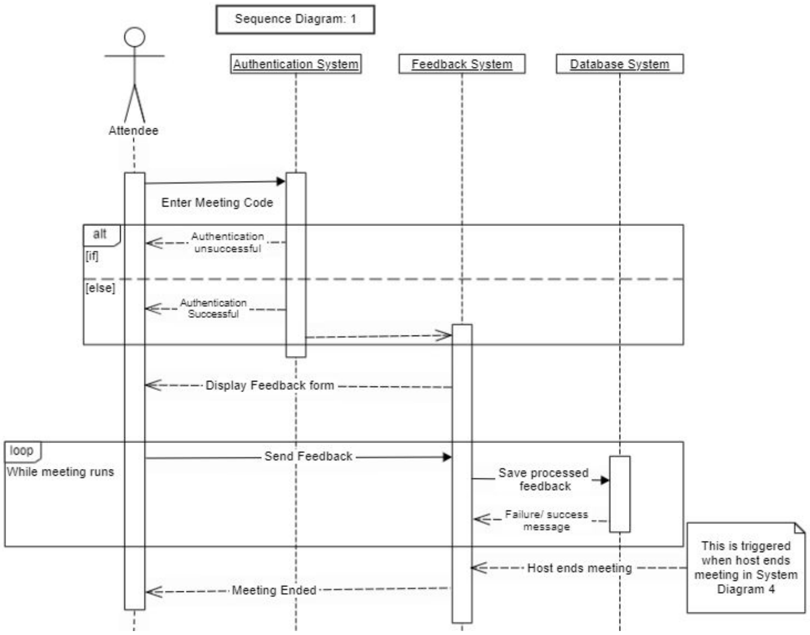

Figure 4: Sequence diagram depicting system collecting attendee feedback data

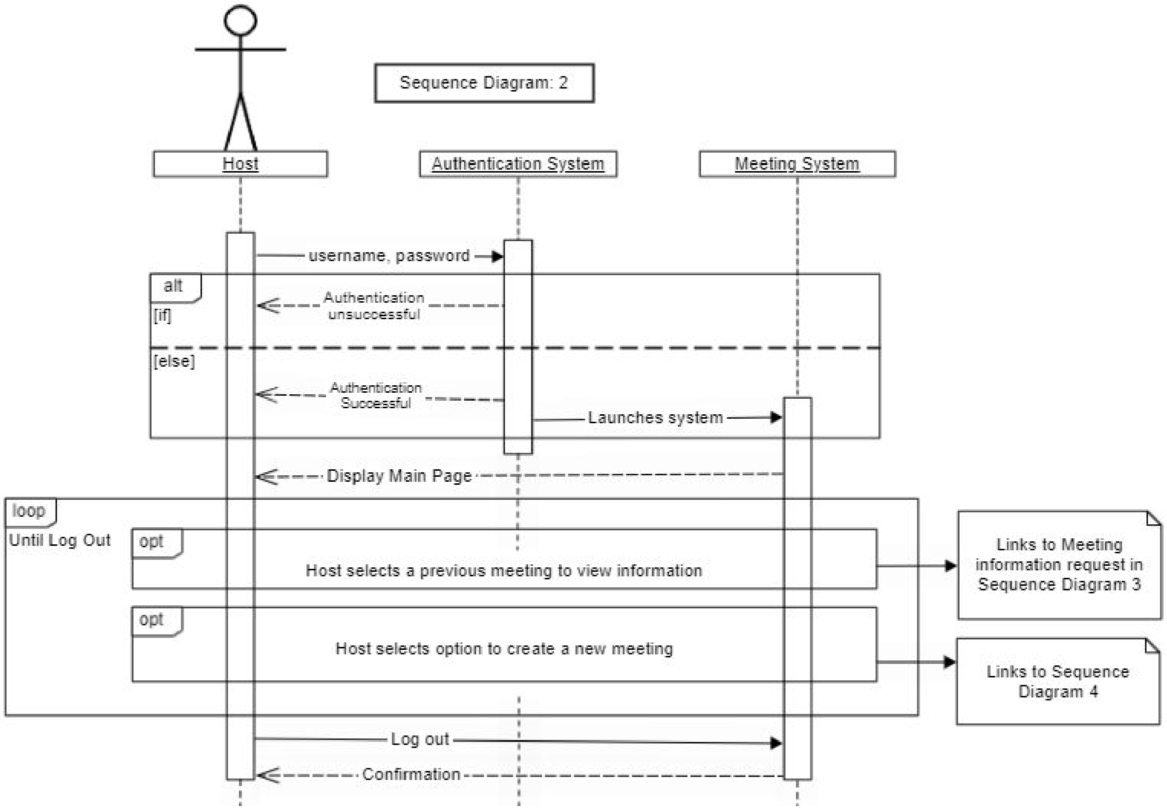
Sequence Diagram: 2
Host
Authentication System
Meeting System
username, password
alt
[if]
Authentication unsuccessful
[else]
Authentication Successful
Launches system
Display Main Page
loop
Until Log Out
opt
Host selects a previous meeting to view information
Links to Meeting information request in Sequence Diagram 3
opt
Host selects option to create a new meeting
Links to Sequence Diagram 4
Log out
Confirmation

Figure 5: Sequence diagram depicting a host logging in, viewing information about events, and given the option to create a new meeting and log out, interacting with the authentication and meeting systems at various stages.

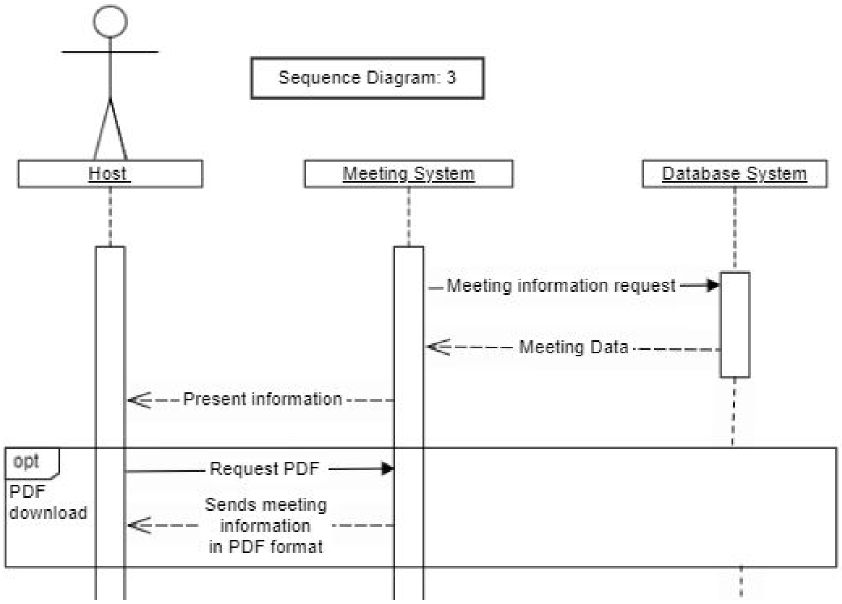

Figure 6: Extension of Figure 5; host interacting with stored meeting. A request is sent via the meeting system to the database, and the information is relayed to the user. The meeting system also formats information as a PDF when requested.

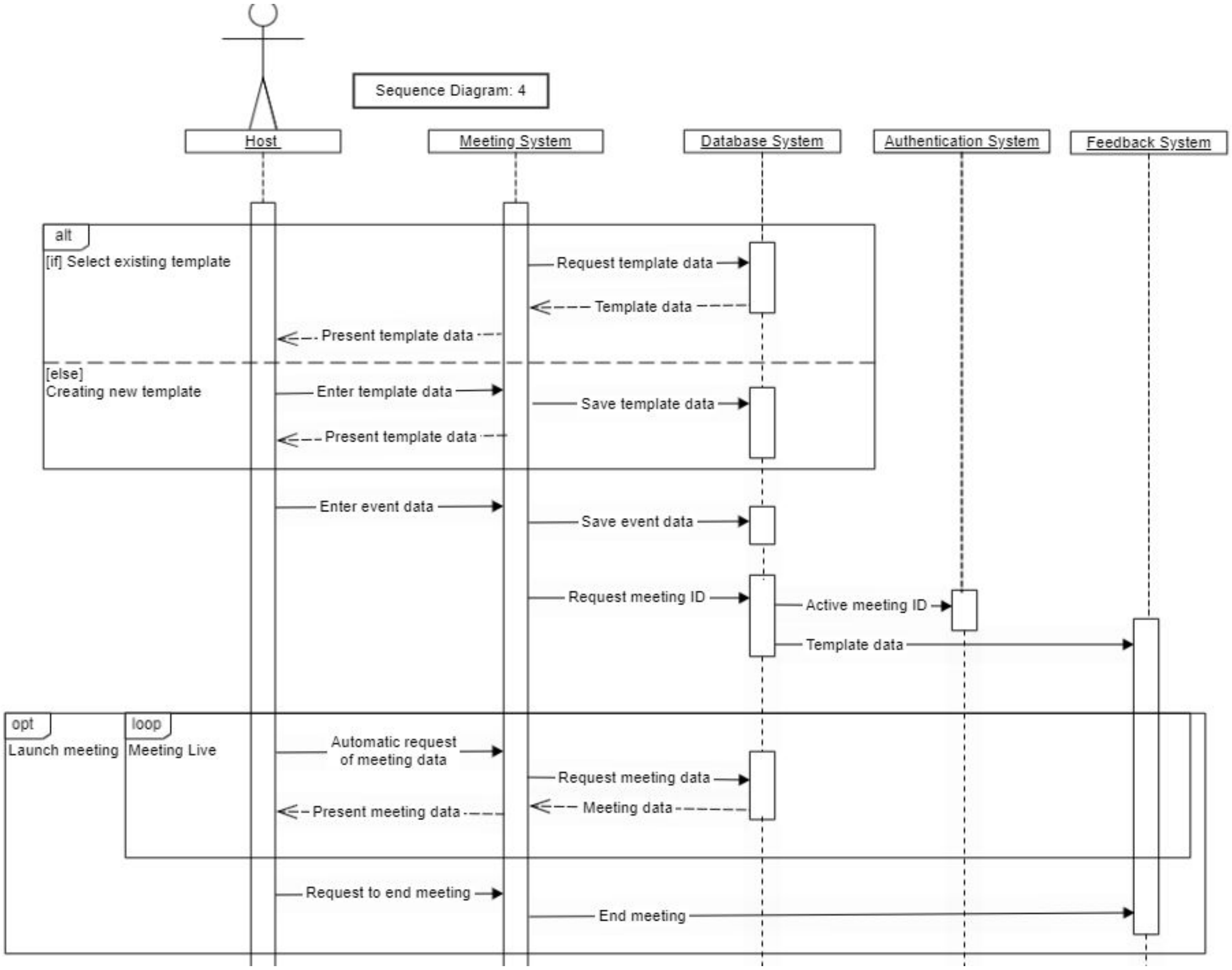

Figure 7: Extension of Figure 5; host creating and launching a meeting. Each subsystem is used during the execution of the following activities: the host can create and edit a template, create a meeting, launch a meeting and end a meeting.

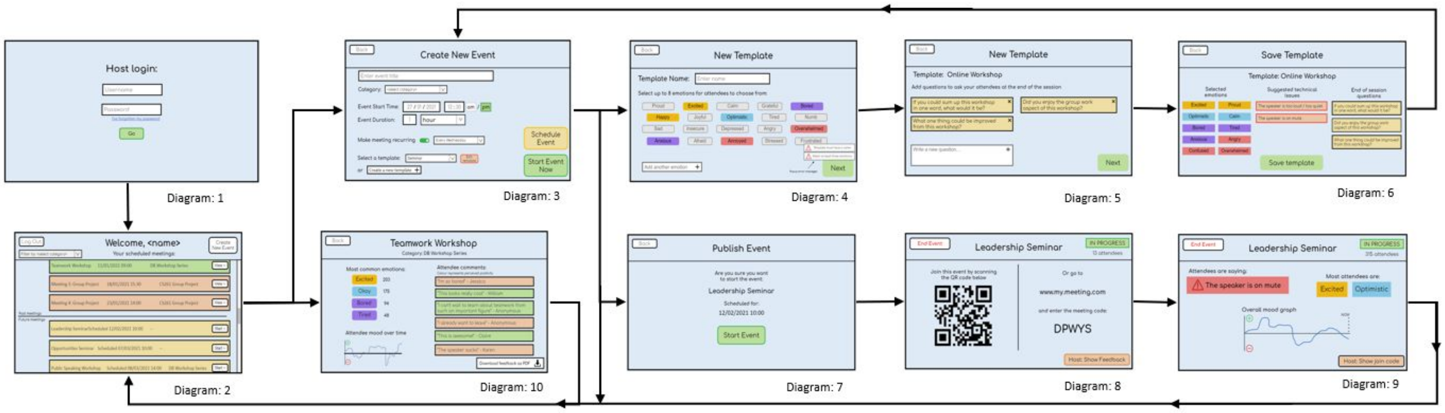

Figure 8: Design of a host user story from logging in (Diagram 1) to creating a new template (Diagram 3) through to seeing visual feedback (Diagram 9).

In Diagram 1, the host can use their email and password to log in (see [R1.2]). When the ‘Go’ button is pressed, the details entered will be checked to see if they are valid, at which point the host can access the system.

In Diagram 2, the host is able to view both events scheduled in the future, where pressing ‘Start’ takes you to Diagram 3 where the meeting can be launched, as well as any previous events, where selecting ‘View’ will display the information collected, as seen in Diagram 10. This feedback goes into detail about specific comments and also allows the host to download a PDF of the feedback (see [R.2.2]). The host is able to sort events by date, and filter by status (ongoing, past, future) or category (see [R.2.3]).

In Diagram 3 the host is able to input information to create a meeting (see [R.3]). This includes a meeting title in a text box, a meeting category from a dropdown menu, a meeting date and time through a calendar pop-up (defaulting to the current date and time), and the meeting duration. The option to have a recurring meeting and select how often the meeting recurs from a drop down list is also available (see [R.3.3]).

Each meeting also has a template to use to collect feedback, which can be selected from a drop-down list of existing templates, or by creating a new template/editing an existing one (see [R.4]).

Once they have entered meeting details, a host can either start the event immediately, directing them to Diagram 7, or schedule the event for the specified time, directing them back to the welcome page (Diagram 2).

Diagrams 4-6 show how the host can customise the feedback they receive from attendees. These stages cover both the process of creating a new template, and editing an existing one. Diagram 4 shows the form to make a template, which allows the host to enter a name for the template, and select up to 8 emotion keywords using the buttons or by adding their own emotions (see [R.4.2]).

The green ‘Next’ button allows the host to progress to Diagram 5. Here the host is able to write questions to ask attendees when the meeting has finished (see [R.4.3]).

In the final stage of template creation, Diagram 6, the host is presented with a summary of the template they have created, with the option to either press the ‘Back’ button to make changes, or to press ‘Save template’. Once the host has saved their template, they are returned to Diagram 3 to finish creating their meeting.

When a host starts a meeting, from the ‘Start Event’ button in Diagram 7, they are directed to Diagram 8 which displays the codes which allow attendees to join a meeting. These are a QR code (see [R.5.2]) and the unique meeting code (see [R.5.3]). The host can then share this information with attendees, and use the toggle button in the bottom right corner to switch to a live feedback summary page. The join codes can be displayed at any point.

Diagram 9 displays live feedback to the host during the meeting. This consists of an overall mood graph showing the mood of all the attendees as the meeting progresses, which emotions are being reacted to most at the present time, and which technical problems are being reported (see [R.10]).

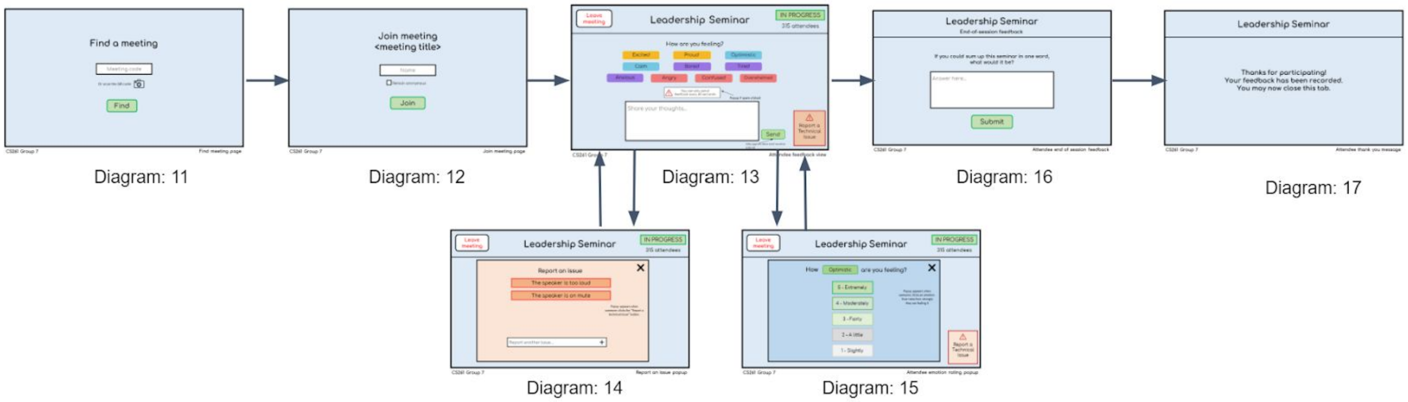

Figure 9: Design of attendee user story, showing entry to a meeting, pages to give feedback, and end-of-sesion questions.

Diagrams 11 and 12 show how the attendee will join the meeting. On Diagram 11, the attendee has the option to join the meeting by either pressing the button to open their camera and scan a QR code (see [R.6.1]), or typing a unique code into a text box (see [R.6.2]). Once the attendee has entered the code, the ‘Find’ button will check if the code is valid. If the code is invalid an error message will be displayed (see [R.11.2]).

A valid code will take the attendee to Diagram 12, where they can type their name into a text box, or check a box to indicate they wish to remain anonymous (see [R.7]). The attendee can then use the ‘Join’ button to enter the meeting, which will take them to the screen in Diagram 13.

Diagram 13 shows how the attendee will input their feedback during the event, which can happen in the following ways: Buttons labelled with emotions may be selected by the attendee, where pressing an emotion takes you to Diagram 15, in which they can choose the intensity of the emotion that they feel; Typing free form feedback into a text box and pressing the ‘Send’ button; Reporting a technical problem using the red button, directing to Diagram 14, where the

attendee can select the problem they are having. The attendee will be returned to Diagram 13 after submitting feedback, allowing them to submit again should they wish to do so.
If they attempt to submit further feedback less than a minute after the last feedback, excluding reporting of technical issues, an error message will be displayed asking the user to wait (see [R.8] and [R.11.2]).

Diagram 16 allows the attendee to answer post-meeting questions defined by the host (see [R.4.3]). After the event has ended, a thank you message is displayed as shown in Diagram 17 (see [R.11]).

System accessibility

Creating a web application ensures that any user on any device with a web browser should be able to access the program. This is more convenient than downloading an app, especially if the event has started and attendees need to join quickly. Buttons are large so they can be used even on devices with smaller screens (see [R.11.1]).

The user interface has been designed with ease of use in mind. Each screen is simple and contains only the minimal content required for good functionality, so it doesn't crowd the page. All buttons and options are clearly labelled, and positioned generally in the centre or bottom right corner, as is standard. This makes the system intuitive to use with minimal training (see [R.11]). The use of colour helps the user to understand the meaning of the buttons; for example, the technical issue buttons are red which is associated with urgency. Buttons with the same purpose are the same colour throughout, for consistency.

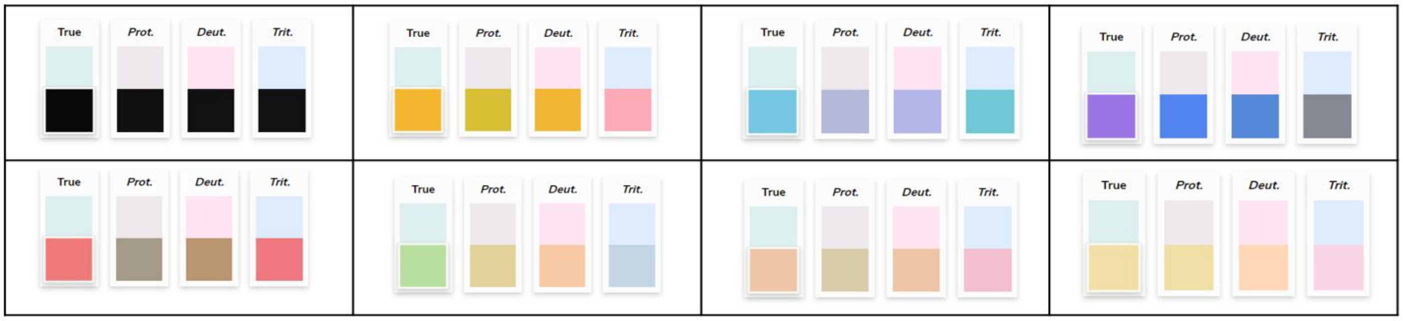

Figure 10: System colour palette showing chosen colour scheme and representation to users with visual impairments.

This colour palette has been selected to be accessible to those people with colour-blindness (see [R.15]). Each of the colours selected is shown in Figure 10 in its true form, and also how people with protanopia, deuteranopia and tritanopia see them[12]. They are shown in comparison with the background colour to demonstrate that all elements are distinguishable. Bolder colours have been selected for emotions as colours are often associated with feelings. Paler colours have been selected for other components to create a user interface that is easier on the eyes.

Process documentation

We have chosen to base our approach on the waterfall methodology[13]. The project naturally lends itself to this approach as the specification, despite being vague, is unlikely to change, and the client is unavailable for feedback on the deliverables. The project requires that we make our own decisions and assumptions which allows the team to accurately predict what will be developed. The Client requires the product at a set deadline and will not provide feedback on any version before that, and as such updating the Client regularly on the system's development is not required. As we are a relatively inexperienced team, exhaustive documentation will help us ensure the project is running smoothly and we are able to help one another as needed. The project is not long-running and can't overrun so the environment, technologies and business objectives are unlikely to change.

In order to ensure deadlines are met and we have enough time to design, develop and test the system, the Gantt chart below shows our projected plan for work on the system. Bars which are partly filled in show progress of those sections up to the date of this report submission. These times are approximate and subject to change, but in order to ensure deadlines are met there will be frequent progress reviews and meetings, to keep the project working to its original timescale and produce a functional prototype in the given time.

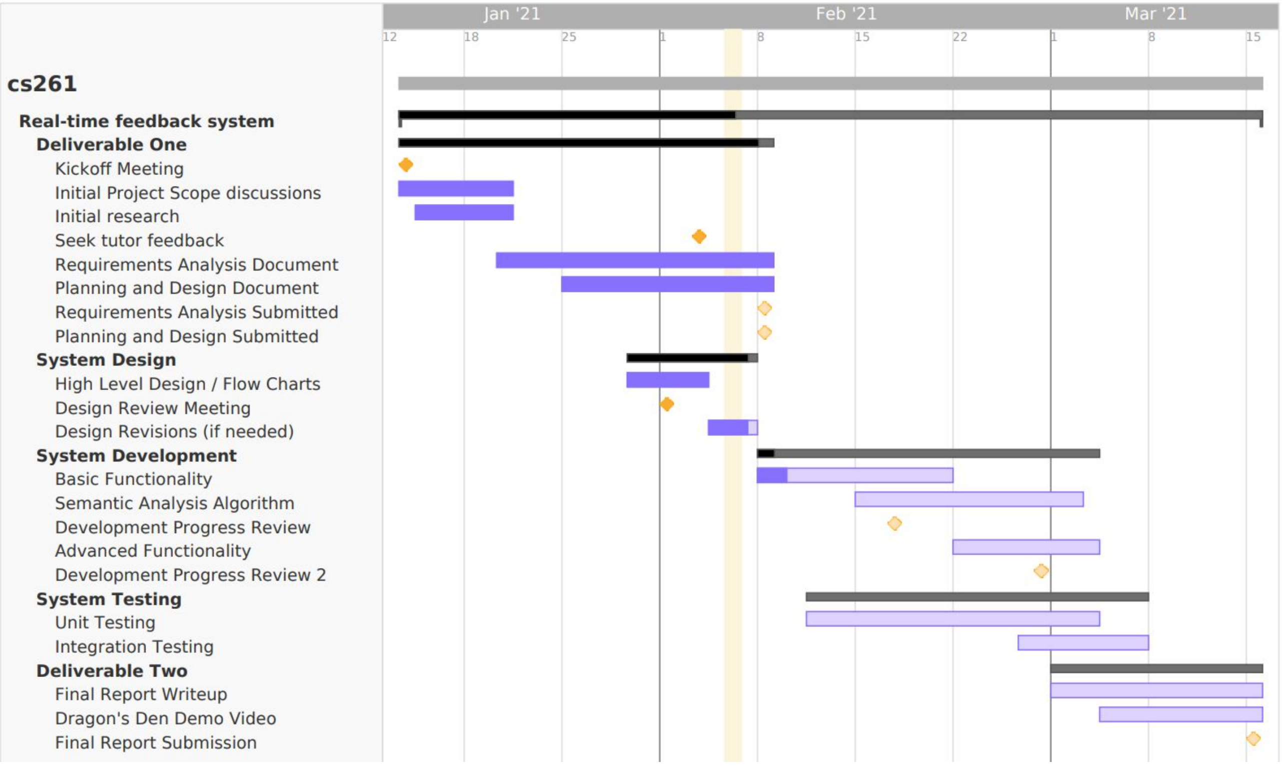

Figure 11: Gantt chart to show the sections of the system development and their respective timescales.

The team roles we have assigned and their descriptions are outlined in detail in the Requirements Analysis document.

We will use a variety of communication channels to ensure effective dissemination of information. For team meetings we predominantly meet through Microsoft Teams scheduled meetings, and this has proven useful for file sharing and general organisation. To aid team collaboration, we use a shared Google Drive folder to allow every team member to view documents simultaneously and work together in real time, as well as a private GitHub repository to establish a shared codebase and allow version control in case of development problems.

Any project is subject to potential errors occurring. To minimise the problems any risks could pose, Table 1 shows our analysis of potentially impactful scenarios and describes the contingency plans in place to avoid any major problems. For brevity, table headings have been abbreviated: L=Likelihood, I=Impact, RI=Reduced Impact. Numerical ratings are on a scale from 0 (impossible/insignificant) to 5 (very likely/catastrophic).

Risk	L	I	Contingency plan	RI
Team member illness	2	2	Work can be reassigned to another team member. Since our methodology encourages well-documented work, picking up someone else's work should be straightforward.	1
Management change	0	4	There is no likelihood of our project managers changing during this project, since the project is effectively managed by the module organiser who will not be changing.	4
Planning / size underestimation	2	3	Devote more time to the project, ensure the initial plan contains extra time to allow for things taking longer than expected. If necessary, prioritise basic features over extended functionality.	2
Change in requirements	2	4	Due to our use of the waterfall methodology, a significant change in the project requirements would make a rewrite of documentation and design necessary, setting the project back significantly.	4
Data loss	1	3	Both GitHub and Google Drive have version control systems implemented to allow	1

			rollback if data is lost or deleted.	
Technology change	1	4	If significant changes in the technologies we have chosen occur, code may have to be rewritten and progress would be set back. There is little we can do to mitigate this type of problem.	4
Competition	0	2	There is no competition for this project since we are essentially working internally for Deutsche Bank, and nobody else is likely to be designing similar software for free.	2

Table 1: Risk analysis and mitigation plans

Testing plan

Unit testing will be done by individuals on each section of the code that is developed. Utilisation of pair programming will reduce the number of errors made and speed up debugging. Peer review with a member who was not present in the original coding will aid clarity and correct functionality. To unit test the Sentiment Analysis algorithm, we will create a mock dataset of input that is likely to be raised by attendees, matched with intended sentiment, to determine if the algorithm functions correctly.

Integration testing is necessary to ensure the front-end and back-end of the system, which have been developed separately, can communicate with each other. Integration testing will happen as soon as practical: when there will be enough components to make the system function. Regular dialogue between the back-end and front-end teams will aid the development of a cohesive system. We aim to carry out soaking tests to ensure the system is reliable and can serve its purpose to the Client. Upon successful completion of these, we will carry out a stress test, to minimize the risk of an overload when it is used by the Client, and verify robustness.

There is no possibility of invalid input in selecting emotions as users can only select from a preset list of emotions. The free text response will be stripped of any characters not part of the standard Latin alphabet.

There is no need to test against different operating systems as it is a web-based application. Basic tests to ensure the system is compatible with various popular browsers, such as Chrome, Opera, Edge and Mozilla will be performed, to ensure the page loads correctly. No installation is required, and since the system is web-based, it can be run on a variety of devices, with the only requirement being internet access and a compatible browser.

System testing/functional testing can be done visually by running the system and observing if all elements are present and system is functional on a local machine. We will conduct the following system tests to determine if requirements have been met:

(T1) System displays a log-in page for a host

(T2) System does not allow login with empty or incorrect password

(T3) System allows login with correct username-password combination

(T4) System displays welcome page with option to Log Out and Create New Event

(T5) `Log Out` button returns user to log-in screen

(T6) Welcome page displays message if no events available to view

(T7) `Create New Event` button allows user to create new event

(T8) All necessary text boxes and dropdown menus are present to create an event

(T9) A new template can be created by the host

(T10) Emotions can be selected and deselected for the template

(T11) A custom emotion can be added to the template

(T12) More than 8 emotions cannot be added to one template

(T13) Next button cannot be pressed if fewer than 3 emotions are selected

(T14) Questions can be written for end-of-session feedback

(T15) Next button can be pressed when no questions have been written

(T16) Template is displayed to confirm creation

(T17) Event can be scheduled for a future date, system returns to the welcome screen

(T18) Selecting a past event displays summary feedback

(T19) Past event summary data can be downloaded in PDF format

(T20) Selecting a scheduled event gives the option to start it

(T21) Starting an event displays a QR code and meeting code

(T22) Live event view shows connection state

(T23) Live event view shows number of attendees

(T24) Live event view shows option to end event

(T25) Live event view shows option to toggle between displaying meeting code and displaying summary of feedback
(T26) Live event summary feedback is displayed as a line graph alongside most commonly selected emotions
(T27) Technical issues flagged up appear on the live event summary screen
(T28) An attendee can use the QR code to join a meeting in progress
(T29) An attendee can enter the meeting code to join a meeting in progress
(T30) An attendee has the option to enter their name
(T31) An attendee can check a box to remain anonymous
(T32) Pressing join displays a page where an attendee can submit feedback
(T33) Submitting feedback more than once per minute displays an error message and the feedback is not sent
(T34) Attendee can select an emotion
(T35) Attendee can rate how strongly they feel that emotion from 1 to 5
(T36) Attendee can press `back`-button to stop selecting an emotion
(T37) Attendee can write in a feedback text box
(T38) Attendees can press `submit` to submit textual feedback
(T39) Attendee can press a button to report a technical issue
(T40) Attendee can select a default technical issue
(T41) Attendee can submit a technical issue by text
(T42) Attendee can press `back` to stop reporting a technical issue
(T43) When session is ended by host, attendee can no longer submit emotions and textual feedback
(T44) After a session ends, attendee is shown end-of-session questions from template, with text box to answer
(T45) End-of-session questions cannot be answered more than 10 minutes after a session ends
(T46) Attendee can submit end-of-session questions with a `submit` button
(T47) Attendee is shown thank you message after submitting all questions
(T48) Attendee cannot join a meeting that has ended
(T49) A colourblind user can visually distinguish the parts of the interface
(T50) A colourblind user can easily interact with the elements of the display
(T51) The system removes emoji and non-regular characters from text

This document summarises the planned system functionality and our approach for implementation. We will build and test this system over the course of the next 5 weeks, ensure it meets the specification, and present our solution to Deutsche Bank.

Bibliography

1. Archbold, J. (2021). Group Software Development Project. Online at https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs261/project, accessed 05 Feb 2021.
2. Unknown (2021). React Documentation, Facebook Inc. Online at https://reactjs.org/, accessed 19th Jan 2021.
3. Unknown (2021). React Hooks Documentation, Facebook Inc. Online at https://reactjs.org/docs/hooks-intro.html, accessed 19th Jan 2021.
4. Unknown (2010). Flask Documentation, Pallets. Online at https://flask.palletsprojects.com/en/1.1.x/, accessed 23rd Jan 2021.
5. Krekel, H. et al. (2020). PyTest Documentation. Online at https://docs.pytest.org/en/stable/, accessed 28 Jan 2021.
6. Fielding, R.T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). Chapter 5: Representational State Transfer (REST). University of California, Irvine.
7. Unknown (2021). SQLite Documentation. Online at https://www.sqlite.org/about.html, accessed 04 Feb 2021.
8. Sangmin, S. (n.d.). Cross-browser QRCode generator for javascript. Online at https://davidshimjs.github.io/qrcodejs/, accessed 30th Jan 2021.
9. Unknown (n.d.). Socket.io Library Documentation. Online at https://socket.io/docs/v3, accessed 1st Feb 2021.
10. Unknown (2020). Recharts Documentation. Online at http://recharts.org/en-US/, accessed 3rd Feb 2021.
11. Brown, P.F. et al. (1991). Word Sense Disambiguation using Statistical Methods. In *29th Annual meeting of the Association for Computational Linguistics* (pp. 264-270).
12. Ramakrishnan, C., Kochut, K. J. and Sheth, A.P. (2006). A Framework for Schema-Driven Relationship Discovery from Unstructured Text. *Proc. International Semantic Web Conference*. (pp. 583–596.)
13. Nichols, D. (n.d.). Coloring for colorblindness. Online at https://davidmathlogic.com/colorblind, accessed 19th Jan 2021.
14. Sommeville, I. (2016). Software Engineering, Pearson. 9th ed. (Ch. 2.1.1, pp. 30-32.)