

# Final Report

CS261 Group 7

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

## **Introduction**

Feedback is integral to the development of any high-quality product or service. According to Forbes, 65% of employees want more feedback on their work<sup>1</sup>, but getting feedback quickly and easily is a challenge few successfully overcome. We have created a proof of concept of a system that addresses the two main issues holding people back from providing feedback: insufficient time, and uncertainty about how to do it. This report summarises the design, implementation and testing of a live event feedback system for use in meetings, group projects, workshops and other events. It gives leaders an insight into the sentiment of their audience in real-time, and helps them lead to the best of their ability by providing analytics to enable them to make informed changes.

The specification for this project is provided by Deutsche Bank<sup>2</sup>.

## **Requirements review and product features**

The system we have produced is widely functional and has several useful features to aid those hosting various types of meetings. We have met the requirements outlined in the specification in the following ways:

### **1. Allow a user to set up an event for a particular session, series of workshops, project etc.**

The system allows a host to login and create a new event, with details including a name, a start time, a duration and a category, to tailor each event to the host's needs. The option to create recurring events (like project meetings or a series of workshop sessions) exists but is not currently functional, and is an area in which we would like to expand our product if given more time.

### **2. This event should be viewable by attendees or team members.**

Each event can be joined from a web browser using a meeting code, which can be shared with attendees or team members as necessary.

### **3. Attendees or team members can provide live feedback, possibly using templates.**

When an attendee or team member joins a meeting, they have two options for giving feedback: by selecting an emotion they are feeling, or through textual feedback, or both. At the end of a meeting, attendees are prompted to answer up to 3 questions written by the host, to give specific feedback on the session. We limited this to 3 because longer feedback forms are less likely to be filled out<sup>3</sup> and get lower-quality data as a result.

### **4. Attendees or team members can choose to be anonymous when submitting feedback.**

When joining a meeting, attendees can check a box to remain anonymous, and their user object is stored in the database with an anonymous flag set. Any feedback they give is tagged as 'Anonymous' to the host, so their identity is not revealed, but their unique company ID is stored in order to count attendance accurately. A bespoke semantic analysis model detects if textual feedback contains offensive content. If flagged as offensive, feedback is not saved in the database and does not undergo sentiment analysis, nor is it displayed to the host. In addition, to prevent spam, attendees are not allowed to submit feedback more than once every 60 seconds.

5. Attendees or team members can also provide a 'mood' and context for their feedback.

Every piece of feedback is either an emotion, conveying the attendee's current feeling or mood, or a piece of text in which they can explain how they feel about the session (i.e. provide context).

6. Provide hosts with analytics on their feedback such as general sentiment, sentiment over time, repeated requests etc.

Feedback from attendees is sent to the back-end, to be stored in the database and analysed for sentiment. The sentiment analysis model reports the results in the form of a number between -1 and 1 (where 1 indicates the most positive sentiment) to the host. After a meeting has ended, the host can download a pdf that contains a summary of the data and comments given.

7. Hosts can see feedback live as it is submitted.

The event host is shown a summary page of feedback which is updated every time a new piece of feedback is added to the database: the most common emotions are displayed to the host by name and number of mentions; results of the semantic analysis are presented as a line graph that shows average sentiment over time; textual feedback is presented to the host in the form of a scrollable list; any technical issues are flagged immediately in red.

8. Hosts can edit templates to be specific to their event.

A host can select a default template or create their own, specifying which emotions they want to be selectable to their event's attendees and allowing them to write up to three end-of-session questions. The templates are stored and can be reused for any number of events.

More detailed analysis of the requirements of the specification can be found in our Requirements Analysis document, and our evaluation of the system can be found below.

**System Development Process**

Once the system had been designed, we began to create the individual parts independently, referring to our design documents to ensure the parts could work together. Some members of our team developed the HTML and CSS for the web pages, while others developed the database, back-end and sentiment analysis algorithms. In the last two weeks we tested the front- and back-end system together, and resolved issues that came with that (see section on Team Management and Project Planning). We agree that connecting the two parts of the system sooner would have made the final steps more straightforward, but with some hard work in the final few weeks we removed the problems which this caused. When working on a similar project in the future, we would aim to start integration of the two ends as soon as possible.

Initially, the team found it difficult to learn how to use the version control software Git<sup>4</sup>. Some HTML pages were lost due to overwriting and unfamiliarity with the commands and system. Over time these mistakes happened less often, as the team became more familiar with Git and making branches. We each worked on our own sections in separate branches, and merged all our features to the main branch when we connected the front- and back-end systems. This was very effective for version control, meaning we could roll back to previous states when mistakes were made, and we could quickly share code when debugging and fixing errors together. To prevent accidents involving the main branch (like the previously mentioned overwriting), we set it up so that changes to main required review and approval from another team member.

Learning to create webpages with HTML and CSS was straightforward for basic elements, but more complex styles required research into CSS styling. We used pre-built text boxes from a website<sup>5</sup> to integrate more aesthetic designs, and used other styles such as columns and buttons found in online documentation<sup>6</sup>. Most styles were implemented in classes to facilitate reuse and consistency throughout the system, but some items needed inline styles because they involved complex design elements. Figures 1 and 2 demonstrate our use of consistent button and input box styles.

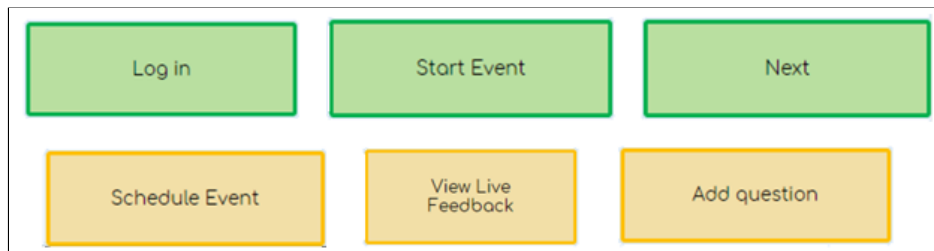


Figure 1: Button styles were consistent throughout the interface

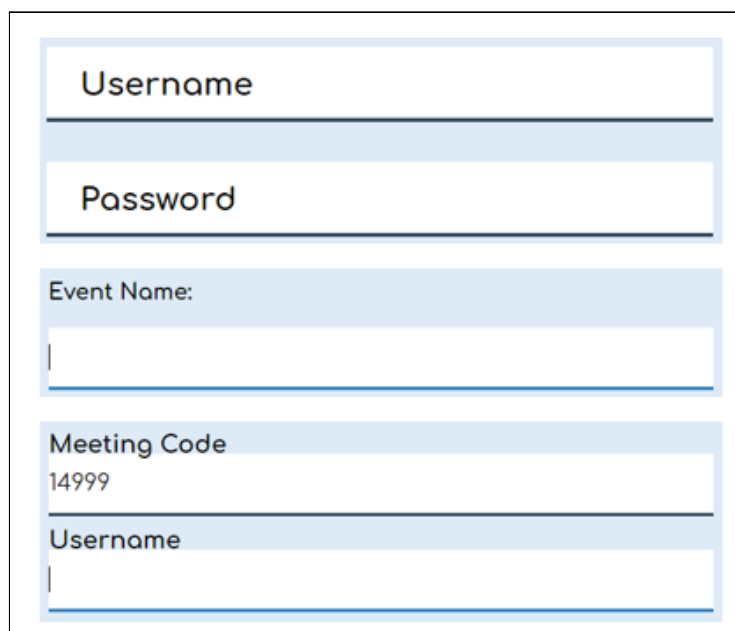


Figure 2: Text input boxes used the same style on each page, with labels moving above the box when entering text

Pages were designed with basic HTML first. However, many pages required elements to respond to user input or information from the database, which made it difficult to check if pages were styled correctly. The overlap between HTML and React<sup>7</sup> code meant team members were not sure what needed to be done by whom, and some work was unnecessarily done with JavaScript and later overwritten in React. We chose to use React over JavaScript because of React Hooks, which re-render a page when information is detected to have changed. The ability to communicate with the back-end via sockets was crucial in displaying live feedback, but involved a steep learning curve for our developers, who were not familiar with React. In addition, React supports a vast array of other functionality in the form of third-party libraries, such as react-dom-router<sup>8</sup> which we used for managing page navigation, and Axios<sup>9</sup> for sending HTTP requests.

Once HTML pages were translated to React, some object placement issues arose such as buttons not being positioned in the right places. As they didn't impair functionality or obscure other elements, we decided the limited time we had would be better spent improving system functionality than aesthetics.

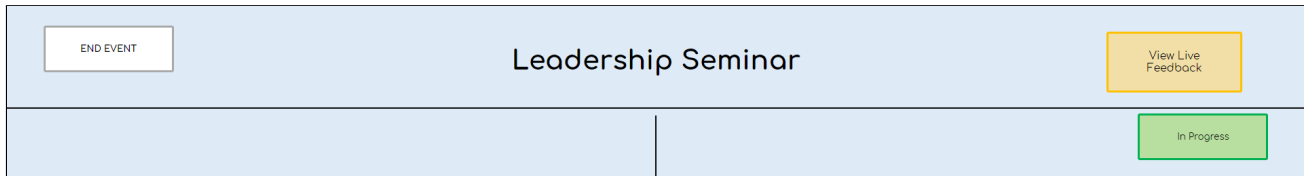


Figure 3: Some object placement issues on the event display page

Our initial database design was very thorough and we identified almost all the key tables and fields that were necessary to create the desired product. The only oversight in our database design was in the table for storing feedback, in which we needed to add an attribute for the sentiment value, to avoid re-analysing it every time the graph was re-rendered. Another database problem arose due to discrepancies between the data types sent by one end and expected by the other. Numerical meeting IDs are stored in the database as strings, but were sent as integers. This was corrected by converting all variables to strings before storing them. Similarly, React expected a different format of date-time objects to what the database stored. Changing date-time formats to timestamp variable types was easy as the database is in third normal form.

Initially testing the look of responsive pages was difficult and involved sending versions of files to a phone to view on a mobile browser. However, after some research, it was discovered that the Chrome and Mozilla Firefox browsers include responsive modes<sup>10</sup> which allow you to preview a device viewport for various models and screen widths. This made visual testing and adjustments much easier.

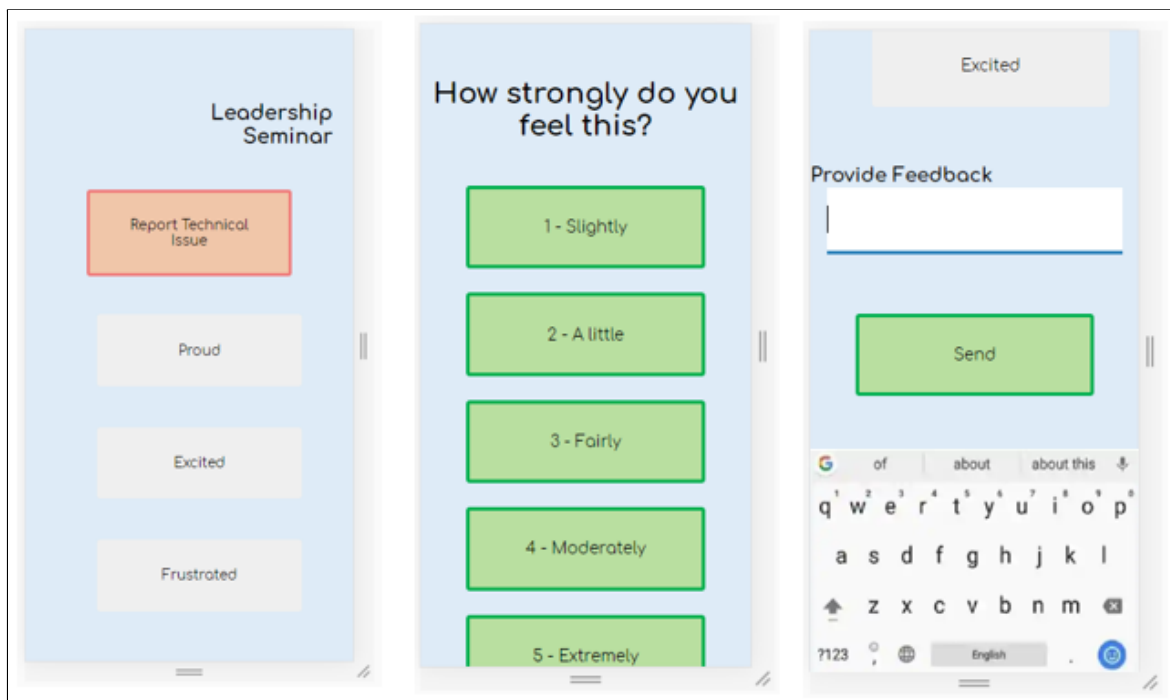


Figure 4: Mobile viewport testing from an attendee's point of view

To aid testing of the back-end, we made use of Insomnia CORE API<sup>11</sup>, which allows sending JSON requests to mimic the front-end. This meant back-end testing could be conducted independently of the progress made on the front-end, which made it an invaluable tool during the initial phase of development.

Final development work in the last two weeks of the project took place on a series of group calls, to which team members dropped in and out when available. This is how we made sure the front- and back-end of the system worked together, and made the final touches. This collaborative approach meant problems could be raised immediately and the appropriate team member could be assigned to fixing them. Issues were discussed with the whole team and we could brainstorm how to approach fixing them, which made the process much faster.

### **Semantic analysis**

We used semantic analysis to achieve two goals in our project: to detect any offensive messages to prevent abuse of anonymity, and to perform sentiment analysis to get a numerical representation of how positive a piece of textual feedback was. This was mostly challenging because we did not discuss the details of how this part should be implemented. The additional research time needed to pick an appropriate tool caused the development of this section to overrun.

We used the open-source PyTorch Natural Language Processing framework Flair<sup>12</sup>. The reasons for choosing Flair were:

- Its simplicity (crucial for the time-constraints we had)
- Ease of integration with the rest of the back-end code (both written in Python).
- Flair's pre-trained sentiment analysis model, which:
  - takes context (negations and intensifiers) into account
  - can estimate the sentiment when it encounters out-of-vocabulary words (such as typos, which can easily happen with feedback quickly typed on a mobile device).

We considered using TextBlob<sup>13</sup> or the Natural Language Toolkit (NLTK)<sup>14</sup> as these are Flair's main competitors. The former was rejected due to lack of sensitivity to intensifiers or negations, and the latter due to inability to infer sentiment from out-of-vocabulary words. We also briefly considered detecting specific emotions from text. Research showed such models are very ineffective (below 50% accuracy<sup>15</sup>) so we decided against it, not wanting to risk misinforming the host.

We made the decision to analyse the feedback sentence by sentence, which gave more weight to longer feedback but allowed negative feedback to be seen, even when in the company of several positively-charged sentences. This approach made it more likely that the result of analysis is correct, as sentiment can get diffused in longer pieces of text.

In order to prevent abuse of anonymity, we trained a model to identify offensive pieces of feedback<sup>16</sup>. The model was trained on a dataset of posts from Twitter, which was stripped of URLs and usernames. The embedding used in the training was uncased, so the model does not differentiate between 'good' and 'GOOD'.

The language used in tweets is quite different to what we would expect to be given as event feedback. As an attempt to mitigate this issue we modified the training, development, and test data to include sentences we

would expect to see. The model generally performed better on our custom input than the whole of the test file. We'd expect to get even better results if we could obtain a dataset better suited to this specific application.

Each attempt at training a model took 6-7 hours, which meant that the number of times we could tweak training parameters and experiment with different embeddings was strictly limited.

Attempts used GloVe<sup>17</sup> word embeddings<sup>18</sup>, and Flair's contextual string embeddings (embeddings based on strings of characters rather than words), but that resulted in low accuracy. Using a combination of these was too computationally-intensive for our machines and was thus terminated. We then discovered Transformer embeddings and tried the uncased BERT<sup>19</sup> embedding, but this also proved too much for our devices. Given more time we would check if using an undistilled version improves accuracy.

We subsequently tried to use DistillBERT<sup>20</sup> (a lighter version of BERT), which achieved an accuracy of 82%. We consider the achieved accuracy a good score, as the sentences with which the model struggled most were extremely negative or ones unlikely to come up in event feedback (such as rap lyrics). We also assume the goal of attendees will be to provide feedback, not trick the model.

### Final design

Our final design was very similar to our initial plan, with a few minor changes. The general colour scheme, page layout and system structure was the same as we planned, but the following pages were removed:

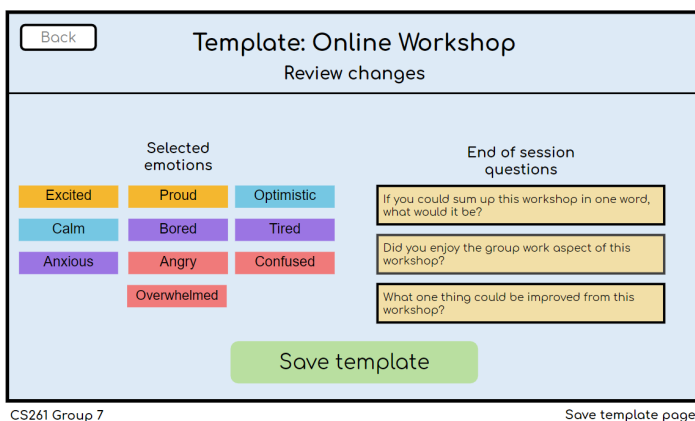


Figure 5: Template summary page, decided to be unnecessary as all template creation was performed on one page.



Figure 6: Attendee thank-you message page, which was not worth creating as redirecting to the login page made more sense.

Other minor modifications included:

- The Create Event page does not include a button to immediately start a meeting once you have entered its details. This was not considered a major drawback since saving a meeting directs you to the events view page where you can click on the meeting to start it.
- Once attendees join a meeting room, they have no button allowing them to leave the room until the event ends. Since this web app runs in a browser, closing the browser tab has the same functionality, so we decided it would not be necessary to add a "Leave meeting" button.
- There is no popup for confirmation upon a host clicking "End event" which we consider a major oversight, since buttons can easily be pressed by accident. An intermediate confirmatory page featuring the text "Are you sure you want to end this event?" with two buttons was part of our initial designs, but was not implemented.

### **Team management and project planning**

Creating a large working system is a complicated process, and managing a team of 6 inexperienced people to effectively work together was a challenge. Team morale and motivation to collaborate was generally high, and we got on well as a team which made working together much easier. Major decisions were made by majority vote, with the Project Manager breaking any ties, to ensure progress was consistently made. For each of our team meetings we kept minutes detailing information discussed, decisions made, and any links shared with the team, to make sure that a team member who missed a meeting could catch up, and to ensure any key decisions were referenceable.

The roles we assigned team members largely remained the same throughout the project. The only exception was the role of Business Analyst which was replaced with Semantic Analyst. The reasoning behind this was twofold. Firstly, the utility of a Business Analyst dropped significantly after submission of the first deliverable as market research was completed and direct communication with the client was impossible. Secondly, developing a semantic analysis tool was a larger task than we initially assumed. As such, Dominika was re-assigned to research and design a sentiment analysis tool, and subsequently to train a model to detect offensive messages.

We believe the waterfall methodology<sup>21</sup> which we chose to follow worked well given the constraints of this project. Although development of each part of the system took longer than initially estimated, we still managed to put together a functional system in the time provided, which we believe meets the requirements outlined in the specification and further detailed in our Requirements Analysis document.

We initially designed our time plan using a Gantt chart to determine when tasks would need to be done in order to complete the project within the specified timeframe. This was not particularly effective due to its lack of sufficient detail. Figure 7 shows our revised Gantt chart based on the actual times taken for development and testing<sup>1\*</sup>. Developing basic functionality took longer than expected, because no functionality was present until we joined the front- and back-end parts of the system together. This also delayed testing since we could not test that the system as a whole worked until it was put together. We also did more research and spent more time on sentiment analysis tools and on training a model fit for our purposes than initially accounted for. Though these tasks took longer than expected, we were able to re-assign team members to do the extra work necessary and complete everything in the allotted time.

---

<sup>1\*</sup> For the originally planned timeline consult the Planning & Design document

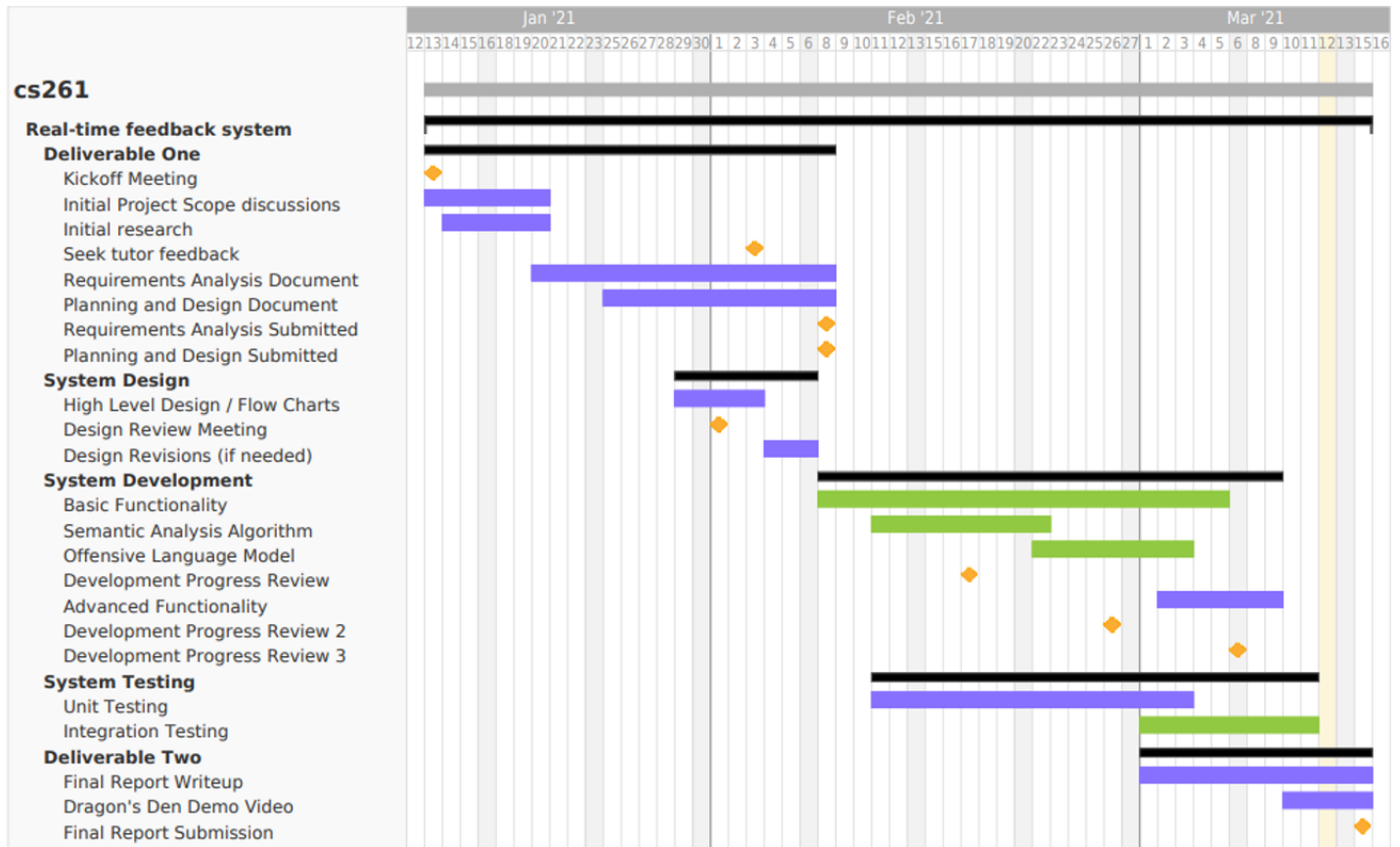


Figure 7: Revised Gantt chart with green bars showing changes

Our development plan for the waterfall methodology which we chose to follow was not detailed enough to be able to be stuck to rigidly. It was difficult to plan how long each element would take to design. The initial creation of HTML, CSS, and responsive elements for mobile devices was more difficult than we planned and required lots of research.

A lag in the development of the Natural Language Processing (NLP) tool was caused by the initial research being not in-depth enough. Significantly more consideration was put into it during the development phase, when implementation should have been happening. The decision to try to use an NLP model to detect and block offensive comments was made after the requirements analysis was finalised, which further slowed down development.

Having regular meetings helped keep us on the same page, and communication was frequent between all team members. However, more detailed progress was not shared as frequently, and different members of the team were unaware of exactly what stage each person was at with their sections. In particular, we had trouble with communication channels between the front-end pages and the back-end system, because we didn't properly decide on standards for transferring information within JSON requests. Lots of time was spent in the final few weeks of development debugging issues with sending and receiving information, which could have been resolved by setting up stricter standards earlier in the development process.



Distributing knowledge about the different aspects of the system was generally poorly done, since individual team members worked with their front- or back-end counterpart and very little communication between these teams happened. This meant our bus factor<sup>22</sup> (the minimum number of team members who could be hit by a bus to stop the project from running) was concerningly low throughout the project. This caused a problem when we came to test communication between the front-end system and the database, because the developer in charge of the React pages and sockets was not available for the meeting, and when we encountered errors we were not able to fix them until we had the whole team together. More substantial documentation and detailed code commenting could have alleviated these issues, and pair programming would have been a more effective solution to this problem, allowing more team members to understand how each part worked.

### **User interface design evaluation**

Throughout the development process, we have been refining the design of our system to better meet the needs of the client. In this section, we will evaluate the design of our system, the principles which underpin our decisions, and the overall outcome and its relevance to the client's needs. One industry-standard set of principles used to evaluate the user interface design is Nielson's Usability Heuristics<sup>23</sup>, which is a set of ten guidelines for designing intuitive, logical and user-friendly interfaces.

#### **Visibility of system status**

After submission, the text input is deleted and all emotions deselected. This signals information has been sent to the server. A host sees changes on screen in real-time after attendees submit feedback.

An icon in the top right of the live event view to indicate if the system is working is not currently implemented. General error messages when fields are filled in with disallowed feedback could be introduced, along with a throbber (loading screen icon) when information is being fetched from the database to keep users informed.

#### **Match between system and the real world**

The system uses the terms "event", "start event" and "login" which users are familiar with and correspond to real-life, tangible events. They should also be familiar with the names for emotions. Text fields are labelled to indicate what input is expected.

Emotions or questions declared by the event host may contain typos or an obscure emotion. We assume this could be dealt with by the host. In the future, gradient colours could be implemented to signify the intensity of an emotion.

#### **User control and freedom**

Back buttons are present on every page, so users can go back to a previous page or step in the process at any point. It is not possible to un-submit feedback after hitting the 'submit' button. An attendee may submit feedback again shortly after if they realise they have more to say.

We did not have time to implement undo buttons for host template creation. Likewise, when leaving the event creation page to create a new template, any previously entered details are lost. To minimise the inconvenience we moved select/create template buttons to the top of the page but temporary data storage should be implemented before the system is deployed.

#### **Consistency and standards**

The terms "event", "start event", and "login" are used throughout the system for internal consistency. Next and back buttons are located in the same area of the screen and are the same colour. Commonly used terms

(like login, back, next, submit) have the usual meaning, maintaining external consistency. Next and Submit buttons are green which have connotations of positivity the user is familiar with. Buttons that the user can click are coloured, and those which are simply for display are grey, so the user can easily differentiate between functional and informational components.

#### Error prevention and correction

Error messages inform a user of a mistake they made in plain language (e.g. logging in with incorrect credentials). The text stored in the database is stripped of any illegal characters. However, this does not discourage the user from repeating the mistake and some sense they tried to convey by use of emojis can be lost.

There is no verification to stop a user joining a meeting due to a mistyped meeting code. This could take them to a different meeting than the one they intended to join. We briefly considered managing this by monitoring the user's location but decided it is an unnecessary prevention mechanism and could potentially be problematic for events held online. Meeting codes are different enough that they should not be joined by mistake and upon joining an event the event title is displayed. If an attendee notices they have joined the wrong meeting, they can use the back button to return to the login page and input the correct one.

Some constraints dictate the type and amount of input allowed, like imposing an upper limit on the number of emotions selected by the host. This is mainly to fit the page on a mobile screen and not overwhelm an attendee. The system is simple and self-explanatory enough that it should be clear to any user.

#### Recognition rather than recall

All buttons displaying functions the user can take are clearly visible on the screen at any time and are labelled with their function. They are colourblind-friendly<sup>24</sup>. There is no long tutorial, and the system does not require any long-term memory beyond remembering your username and password, because it engages sensory stores.

#### Flexibility and efficiency of use

The product doesn't feature shortcuts as it is a relatively linear and straightforward program. Keyboard shortcuts or other methods of speeding up workflow could be introduced in the future. Using a pre-saved template (or one of the ones available by default), as well as indicating the event category could speed up the process of setting up an event, or finding an existing one.

#### Aesthetic and minimalist design

Each page only contains the elements which directly relate to the system's functionality at that point. No images or additional styling have been added to pages to ensure the system remains aesthetically pleasing. We have chosen a font very similar to that which Deutsche Bank (DB) uses on their website because it is simple, clear, and easy to read, as well as being familiar to users (mostly DB employees). Buttons with functions are large and easy to read. To aid recognition of emotions, we have employed the Gestalt Laws<sup>25</sup> of Proximity and Similarity by placing emotions with similar connotations close together, and giving them the same colour so that users will perceive them as a group. We have also grouped past and future meetings by colour on the events view page.

#### Help and documentation

We have a readme document that explains how to install and use the system. The system is self-explanatory. An attendee having trouble using the system should be able to obtain help from the event host or other

attendees. In the future, if the site is accessed by an unsupported browser, there should be a message informing them of which browsers are supported.

### Affordances

In addition to the general principles of good user interface design, our design makes use of affordances<sup>26</sup> - properties of objects which show users the actions they can take. The affordances we have included, and the purposes they serve, are described below.

After an emotion is selected, a popup that asks the user how intense they feel the emotion covers most of the page. This makes the user concentrate on only one aspect of the environment. The same affordance is used when an attendee or team member submits a technical issue. We don't make use of Gaussian blur to hide unnecessary information, but this could be implemented in the future to help focus the user's attention on a part of the screen e.g. when submitting a technical issue.

On the summary page, the host's attention is involuntarily drawn to technical issues as they are large and red, but this is the only affordance relating to attention that we have made use of. We opted not to use sound effects or flashing imagery to grab the host's attention, as we deemed this to be an unnecessary distraction.

We have used icons to represent actions in the system, to allow a quicker understanding of the results of clicking on symbols. We planned to use a camera icon next to the option to scan a QR code, in resemblance of something a user might use to take a picture of a QR code, but the time constraints prevented this. An exclamation mark next to the 'Report a Technical Issue' button is a symbol commonly used for something that requires immediate attention.

### System functionality evaluation

The final product has a variety of unique and interesting features, which are highlighted here. However, there is significant scope for improvement.

#### Features and successes

- The system provides a quick and easy route for attendees to give feedback during a live meeting, eliminating doubts about how to give feedback.
- The login system for hosts is a secure process involving checking a username and password against a database of accounts and hashing passwords, to protect sensitive data and comply with GDPR law.
- Feedback considered to be offensive is removed from the system and not displayed to the host, to prevent abuse or misuse of the system's anonymity option.
- Creating an event is a quick and streamlined process. The system comes with three default templates, with the option to create your own.
- Hosts can view a summary of the sentiment of all comments from the meeting, including a live graph of the positivity of each comment and the most commonly selected emotions.
- Technical issues reported by users are shown to the host immediately, so problems can be resolved with ease when attendees notice them.
- A complete feedback summary for an event can be downloaded to be shared or viewed offline.
- The system is designed with colourblind users in mind and is therefore accessible to many more potential users.

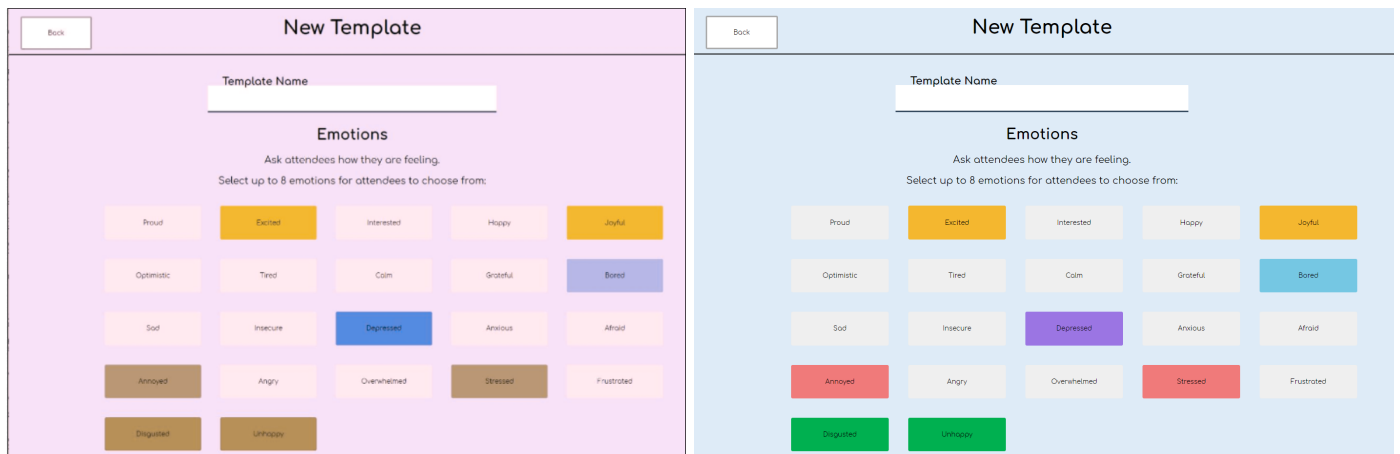


Figure 8: (Left) the system UI as seen by a user with Deuteranopia (green-blindness) and (right) a user who can see the entire colour spectrum

### Weaknesses and problems

- Creating an event or custom feedback template takes a long time at first, though this is likely to improve with more experience of the system.
- Anyone with a valid company ID could join a meeting and submit feedback if a meeting code was shared outside of a meeting. This is not a big problem as we filter offensive feedback out, but even this more secure method of joining allows the host to receive feedback from non-participants.
- The system does not prevent the meeting host from joining their own event and give feedback, so it is at present not advised to put in place a reward system for highest-scoring events' hosts.
- A host is unable to edit a template once the meeting has started, which is inconvenient if a change to questions or emotions is required.
- A template or event cannot be deleted. This could be an inconvenience for a host of many events.
- As with almost any sentiment analysis system, sarcasm is not detected and may be given an erroneous sentiment rating.
- Due to the dataset the sentiment analysis model was trained on, 'average' and 'ok' are regarded as negative comments. Introducing a 'neutral' category or fine-tuning this model to our specific context would mitigate this problem.
- The graph averages the sentiment across the entirety of the meeting, meaning that even very negative comments later on might not cause a noticeable spike in the graph. The host might thus be oblivious to a problem.
- No sentiment score is attributed to selected emotions, introducing a possible negativity bias (unsatisfied attendees are more likely to give textual feedback).
- The offensive language detection model has been trained on a dataset that only in part mimicked the feedback likely to be received and thus sometimes incorrectly labels an appropriate comment as offensive and vice versa.
- Custom technical issues are not filtered for offensive content.
- No temporary data storage makes a user lose all information inputted if they leave the current screen.

## System testing

In order to ensure the finished system met the requirements we established, we thoroughly tested the system in a local environment to ensure each subsystem worked independently, and as a whole system. The test cases and results are shown in Table 1 below.

Test Description	Input data (if applicable)	Expected output/response	Pass/Fail
Host login	<ul style="list-style-type: none"> <li>- Both boxes empty</li> <li>- Only a username "John"</li> <li>- Only a password "pa\$\$word"</li> <li>- Incorrect username-password combination {"test-user", "wrong_password_4"}</li> <li>- Correct username and password {"John", "pa\$\$word"}</li> </ul>	<ul style="list-style-type: none"> <li>- No login, popup "You must enter a username"</li> <li>- No login, popup "You must enter a password"</li> <li>- No login, popup "You must enter a username"</li> <li>- No login, popup "Password is incorrect"</li> <li>- Login successful, taken to events view page</li> </ul>	PASS PASS PASS PASS PASS
Host log-out	Click 'Log-out' button	Return to host login page	PASS
Event filtering	<ul style="list-style-type: none"> <li>- Select a category to filter by from dropdown menu</li> <li>- Select an empty category</li> <li>- Deselect category</li> </ul>	<ul style="list-style-type: none"> <li>- Only events in that category appear</li> <li>- No events shown</li> <li>- All events appear</li> </ul>	PASS PASS PASS
View past event	Click on a past event that has finished	<ul style="list-style-type: none"> <li>- Summary event data shown</li> <li>- Option to download as PDF</li> <li>- 'Back' button shown and functioning</li> </ul>	<b>FAIL</b> PASS <b>FAIL</b>
Create new event	Click 'Create new event' button	Event creation screen shown	PASS
Create event	<ul style="list-style-type: none"> <li>- Start event with no title</li> <li>- Start event with "😂" as title</li> <li>- Choose start time before current date/time "20/02/2004"</li> <li>- Set event duration to negative number "-5"</li> <li>- Start event with no template selected</li> <li>- Press 'Edit template' button</li> </ul>	<ul style="list-style-type: none"> <li>- Popup "Event title necessary"</li> <li>- Popup "No emoji allowed"</li> <li>- Times before current date cannot be selected</li> <li>- Text box only accepts whole numbers greater than 1</li> <li>- Only created templates can be selected, and there is a preset default template</li> <li>- Directed to Create Template page</li> </ul>	<b>FAIL</b> <b>FAIL</b> PASS PASS PASS <b>FAIL</b>
Template creation	- Press 'Next' with no template name	- Popup "Template title necessary"	PASS

	<ul style="list-style-type: none"> <li>- Press 'Next' with "😄" as template name</li> <li>- Click on an unselected emotion</li> <li>- Click on a selected emotion</li> <li>- Select too many emotions</li> <li>- Add {"#\$\$", "😄"} as emotions</li> <li>- Add "Self congratulatory"</li> <li>- Select no end-of-event questions</li> <li>- Select more than 3 end-of-event questions</li> <li>- Press 'Save Template' button</li> </ul>	<ul style="list-style-type: none"> <li>- Popup "No emoji allowed"</li> <li>- Becomes coloured (selected)</li> <li>- Becomes grey (deselected)</li> <li>- Cannot select more than 8</li> <li>- Popup "Emotions must be letters"</li> <li>- Popup "Emotions cannot be longer than 15 characters"</li> <li>- Return to create event screen</li> <li>- 'Add new question' button cannot be clicked</li> <li>- Return to create event screen</li> </ul>	<b>FAIL</b>  PASS PASS PASS <b>FAIL</b>  <b>FAIL</b>  PASS PASS  PASS
Publish event	<ul style="list-style-type: none"> <li>- Click on scheduled meeting</li> <li>- After creating an event, click 'Start meeting now' button</li> </ul>	<ul style="list-style-type: none"> <li>- Directs to Publish Confirmation page</li> <li>- Directs to Publish Confirmation page</li> </ul>	PASS  <b>FAIL</b>
Live event	<ul style="list-style-type: none"> <li>- Press 'Start event' button</li> <li>- Press 'Show feedback' button</li> <li>- Press 'Show meeting codes' button</li> <li>- 'Click 'End Event' button</li> </ul>	<ul style="list-style-type: none"> <li>- Loads a QR code and meeting code. Displays number of active participants.</li> <li>- Hides meeting codes and shows summary feedback data</li> <li>- Hides summary feedback data and shows meeting codes</li> <li>- Directs to Host Events View page</li> </ul>	<b>FAIL</b>  PASS  PASS  PASS
Joining event	<ul style="list-style-type: none"> <li>- 'Find a Meeting' page displays option to enter code or scan QR</li> <li>- Scan QR code</li> <li>- Enter valid event code "14999"</li> <li>- Enter invalid event code "spoon"</li> <li>- Enter code in valid format to nonexistent event "13758"</li> <li>- Enter username, do not check 'anonymous' box</li> <li>- Enter username, check 'anonymous' box</li> <li>- No user name entered, 'anonymous' box unchecked</li> <li>- No user name entered, 'anonymous' box checked</li> </ul>	<ul style="list-style-type: none"> <li>- Directs to Join Event page</li> <li>- Directs to Join Event page</li> <li>- Popup "Invalid code"</li> <li>- Popup "Invalid code"</li> <li>- Popup "Invalid code"</li> <li>- Directs to Attendee Feedback View</li> <li>- Popup "Please deselect the 'anonymous' button"</li> <li>- Popup "Please select the 'anonymous' button or enter your name"</li> <li>- Directs to Join Event page</li> </ul>	PASS  <b>FAIL</b> PASS PASS PASS  PASS  <b>FAIL</b>  PASS  <b>FAIL</b>
Sending live feedback	<ul style="list-style-type: none"> <li>- Click on an emotion button</li> <li>- Click on an emotion intensity</li> <li>- Click 'Submit' after selecting an</li> </ul>	<ul style="list-style-type: none"> <li>- Popup screen to select emotion intensity shown</li> <li>- Popup is hidden</li> <li>- No response (data is sent)</li> </ul>	PASS  PASS PASS

	emotion - Click 'Back' on emotion popup - Click 'Submit' with no text  - Click 'Submit' with text - Click 'Report a technical issue' button - Click 'X' on technical issues popup - 'Click one of the issues in the popup - Enter a custom issue "Speaker is talking too fast" - Enter an empty issue and click send {""} - Enter offensive text "f*ck"	- Popup is hidden (not sent) - Popup "You must type something to submit" - Text is removed from box (sent) - Popup screen to select technical issue shown - Popup is hidden (not sent)  - Popup is hidden (issue sent)  - Popup is hidden (issue sent)  - Popup is hidden (issue not sent) - Popup "offensive text detected" (not sent)	PASS PASS  PASS PASS  PASS  PASS  PASS PASS
Sending end of session feedback	- Click 'Submit' with no feedback entered - Click 'Submit' with feedback just an emoji 😊 - Click 'Submit' with feedback "Incredible" - Click submit with offensive text "f*ck" - Click 'Submit' button for the final post-session question	- Directs to Join Meeting page  - Directs to Join Meeting page  - Directs to Join Meeting page  - Directs to Join Meeting page (not stored) - Directs to Join Meeting page	PASS  PASS  PASS  PASS PASS
Browser compatibility	- Above host-side tests carried out in Chrome on a Windows laptop - Above host-side tests carried out in Microsoft Edge on a Windows laptop - Above attendee test carried out in Chrome on an Android phone - Above attendee test carried out in Safari on an iOS phone	- Pages display and function as described - Pages display and function as described  - Pages display and function as described - Pages display and function as described	PASS  PASS  PASS PASS

Table 1: Test cases and results

Many aspects of our system were not completed due to time and complexity constraints, and these failed tests are primarily parts of the system we were not able to implement, though we would add them to a full system given more time and resources.

Event and template creation is a key feature in our system and works well. However, once an event or template is created, it cannot be edited and the details are fixed. They also cannot be deleted, which would become a problem after a while if there are many events to scroll through, or many templates to try and choose from. The ability to edit, delete and manage events and templates would be a very useful feature to add to improve usability for hosts.

Joining a meeting can only be done via a numeric meeting code, as researching libraries for QR code creation, and generating a QR code for each meeting, was not possible in the time we had. Adding the ability to join via QR code would increase the speed with which attendees could join a meeting, and would be a desirable feature if this system were to be deployed.

Our attendee login page had a couple of issues in the final system, due to the ability to remain anonymous. We chose to check usernames entered against a mock list in the database, mimicking a company's list of employee usernames, as a security measure to ensure members not part of the company could not join the meeting. This in turn made complete anonymity impossible, because attendees had to enter their name, even if they checked the box to 'Remain anonymous'. Increased security and the ability to remain anonymous were conflicting in this case. However, once an 'Anonymous' attendee joins a meeting, their feedback is no longer associated with their name, so though it may be perceived as non-anonymous because of the login page, their identity will not be shown to the host at any point.

When viewing past event feedback summaries, we initially planned to have a web page, similar in style to the host's live meeting view, with the most common emotions and overall mood graph visible, with the option to download a summary PDF afterwards. However, we decided to not add another complex page to the system and set all feedback to be in the form of a PDF for past events. Clicking on any past meeting now generates a summary of all the key information and comments, but there is no dedicated page in the web app for displaying this.

Error messages were not consistently implemented on each page of the system. The login page features an error message if login details are incorrect, but no other page has popups explaining what the user did wrong. This impacts the usability of the interface and increases the time taken to recover from an error, as most pages simply do not allow you to continue until you realise what the error is and correct it. One key example of this is that when creating an event, if you set the start date to be before the current date/time, even if all other fields are correct, clicking continue will not save the event, and it is difficult for the user to identify what the problem is.

Text field validation was also a problem. Our original plan was to filter all offensive words, emojis and non-standard characters from any input, but we only implemented this for attendees' textual feedback. Thus meetings called "Party meeting 😄", custom emotions like "Sócks 🦊", and post-session answers like "¿N! 🤔" are all allowed, and are neither removed nor flagged with an error. We decided that emojis in event names and emotions would allow for more creativity and freedom of expression, so these fields did not have emojis filtered out. A host can also decide they prefer emotions displayed as emojis rather than words. The length of feedback should also have been limited, so long custom emotions that may not fit in a button like "self-congratulatory" could not be entered, and text feedback longer than 300 characters could not be entered - but this was not implemented.

In addition, though a 60-second waiting period was implemented for regular attendee feedback, and inappropriate comments were filtered out, we decided to allow users to submit technical issues as often as necessary. This was because problems with the session should ideally be sorted as soon as possible to minimise the negative impact. However, this 'Custom technical issue' field is open to abuse, since it is not



filtered in the same way as regular text feedback, and attendees could send inappropriate or unhelpful messages through this pathway without consequence.

The attendee feedback page also has a minor issue since it does not allow you to submit an emotion without also entering some textual feedback. This makes it easy to link the context (in the form of text feedback) to the attendee's emotions, but also slows down the process of giving feedback from an attendee's point of view. We would like emotions to be submitted immediately after selecting them, but at present it was easier to bundle all feedback as one object to send to the database.

We did not carry out soaking tests<sup>27</sup> (simulating usual system pressure with hundreds of attendees and meetings happening) nor stress tests<sup>28</sup> (deliberately intense/thorough testing designed to stretch the limits of the system's capacity) due to time constraints and lack of a reliable server the system could be hosted on. If they were to be carried out, we would mainly verify if the system can handle many pieces of textual feedback submitted at once, for the duration of a full working day. Deutsche Bank has roughly 8000 employees in the UK, so we estimate an appropriate soaking test would involve 20 simultaneous meetings, and 500 attendees joining and submitting feedback at the same time. A stress test would involve at least 1000 participants and many more meetings, for a much longer time, to test the reliability of the hosting server and the system.

### **Interface usability testing**

To ensure our system was as user-friendly as assumed above, we showed our system to 12 members of our friends and family to determine whether the interface was as user-friendly as intended. These members included an 82-year-old and two 21-year-old colourblind students, whom we selected to ensure the system would be usable for both older users and those with visual impairments.

For each participant in the test, we presented them with the system and recorded the time taken for them to perform basic tasks such as logging in, creating an event, and submitting feedback. A summary of this testing, and the responses and comments of participants, is shown below:

#### **Test #1: Logging in as a host**

All participants completed this within 10 seconds, by typing a username and password and clicking the Log In button. Some users complained that pressing the Enter key after entering a password did not submit the form, as most websites allow, but using the tab key to switch between text boxes did work as intended.

#### **Test #2: Creating an event**

Participants took between 45 seconds and 4 minutes to create a new event, including creating a new template. Everyone agreed that the interface was straightforward, and the calendar popup for date selection was praised. Almost everyone criticised the fact that when you move to create a new template, and subsequently return to the event creation page, any previously entered data is lost, so event information had to be re-entered upon return.

#### **Test #3: Starting an event**

This was very straightforward and all participants immediately clicked on the event to try and start it. This took under 5 seconds as it only involved two button clicks. Participants liked the start event confirmation page as a prevention method for accidentally starting a meeting.

#### Test #4: Joining a meeting as an attendee

As with host login, this was fairly straightforward and took participants no more than 20 seconds to join. Several participants commented on the option to remain anonymous as a “nice feature” and appreciated that meeting codes were short and easy to remember.

#### Test #5: Submitting feedback

Participants were asked to submit one emotion as feedback, and enter a provided sentence of textual feedback. Timing this test was not appropriate as it relied largely on typing speed. Most participants expressed surprise when seeing a popup after clicking on an emotion button, but the ability to rate how strongly they felt (on a scale of 1 to 5) was regarded positively.

#### Test #6: Reporting a technical issue

This took under 10 seconds for most participants, but two outliers took 17 and 23 seconds to perform this task, because the ‘Report a technical issue’ button was coloured green in the prototype version used for testing, instead of the intended red. This lack of an eye-catching colour made it more difficult for participants to locate the button and complete this task. This colouring issue has now been rectified.

Collecting primary data about the speed and accuracy with which users could complete basic tasks using our interface gave us a deeper insight into the successes and pitfalls of our system. With more time we could implement some of the suggested changes, such as storing data between pages and allowing shortcut keys to move the user through pages they are familiar with.

#### **Future work**

The system we have developed is a working prototype with all the basic features necessary for it to function in a real-life meeting setting. However, our development was severely limited by the time constraints of this project, and with more time we believe many more useful features could be added to extend the functionality of the system. We have described the details of some possible areas of future development here, to demonstrate the capabilities of our system with more investment:

- 1) Improved offensive language filter. The current model reports 82% accuracy. It is accurate enough for this project as event feedback is unlikely to feature sentences that would confuse the model, but no dataset was found that could be more representative of the messages more likely to be typed.
- 2) Live quick-answer questions. Adding a feature for hosts to send out occasional questions relating to the session, and seeing the answers to that question immediately, would encourage attendees to participate more actively and feel like they are contributing more to the meeting, as well as allowing hosts to make sessions interactive. Currently communication is only one way (from attendees to the host), but two-way communication could be enabled so hosts could send questions in the middle of a meeting, and not just at the end.
- 3) Improved security. Implementing more robust security features was out of scope for this project, but there are several ways to increase security, such as by using preparedStatements for SQL queries (to avoid the possibility of an SQL injection attack).
- 4) Extension of language support. Translation of the interface to other languages would allow usage outside of English-speaking countries, by employees not proficient in English, widely increasing the

market to which this system is applicable. Flair has pre-trained models for analysing German text for sentiment and offensive content, and this could be a very effective extension of the capabilities of the system. We could also implement automatic detection of the language used in feedback, and pick an appropriate semantic analysis model based on the language, by identifying keywords and determining which language they are from.

- 5) Decrease event set-up time for familiar users. More default templates could be introduced based on the type of event to speed up event creation. We created a small number of default templates but this could be expanded. Adding keyboard shortcuts would decrease the time it takes familiar users to perform common tasks, so this would be a useful addition.
- 6) Wider variety of question types. With more time we could add the option to ask attendees questions in more formats, like polls with multiple choice answers, to give hosts more flexibility in the types of questions they can ask and receive feedback from.
- 7) Introduction of temporary data storage. Allowing a user to return to a previous page after creating a template without losing information entered would speed up workflow.

## **Summary**

The live event feedback system we have created is a complete proof of concept which could be expanded upon, but features all the necessary basic functionality needed to deploy it in a real-life scenario at present. We believe we have sufficiently met all the specification requirements given by Deutsche Bank, and we are proud of the system we have developed together. There are many areas in which we could expand our system given more time and resources, but we have reached a stage which proves that development of a system that meets the needs of our client is feasible.

## **Bibliography**

1. Lipman, V. (2016). *65% Of Employees Want More Feedback (So Why Don't They Get It?)*. Forbes. Available at: <https://www.forbes.com/sites/victorlipman/2016/08/08/65-of-employees-want-more-feedback-so-why-dont-they-get-it/> [Accessed January 25, 2021].
2. Archbold, J. (2021). *Group Software Development Project*. Online at <https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs261/project> [Accessed 29 February 2021]
3. Wroński, L., *Half of SurveyMonkey surveys are short - why yours should be too*. Curiosity at Work. Available at: <https://www.surveymonkey.com/curiosity/half-of-surveymonkey-surveys-are-short-why-yours-should-be-too/> [Accessed January 20, 2021]
4. Chacon, C, Long, J. (n.d.) GitHub documentation. Online at <https://git-scm.com/doc> [Accessed 4 February 2021]
5. Sahagun, B. (n.d.). Label as placeholder with animation. *CodePen*. Available at: <https://codepen.io/benedicksahagun/pen/WwaKEL> [Accessed 4 March, 2021]
6. Mozilla (2020). CSS Developer Documentation. Online at: <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference> [Accessed 23 February 2021]
7. Unknown (2021). React Documentation, Facebook Inc. Online at: <https://reactjs.org/> [Accessed 4 March 2021]
8. React Training (2020). React Router training. Online at: <https://reactrouter.com/web/guides/quick-start> [Accessed 21 February 2021]
9. MIT (2020). Axios Package. Online at: <https://www.npmjs.com/package/axios> [Accessed 28 February 2021]
10. Basques, K. (2020). *Simulate Mobile Devices with Device Mode in Chrome DevTools*. Online at <https://developers.google.com/web/tools/chrome-devtools/device-mode> [Accessed 02 March 2021]
11. Unknown. (2021). Insomnia Documentation, Kong Inc. Online at: <https://insomnia.rest/> [Accessed 27 February 2021]
12. Akbik et al. (2018). *Contextual String Embeddings for Sequence Labeling*, 27th International Conference on Computational Linguistics (Coling 2018, pgs. 1638-1649)
13. Loria, S. (2020). *TextBlob: Simplified Text Processing*. Online at: <https://textblob.readthedocs.io/en/dev/> [Accessed 2 February 2021]
14. Loper, E., Bird, S. (2002) *The Natural Language Toolkit*. (CORR, 2002).
15. Acheampong, F.A., Wenyu, C. & Nunoo-Mensah, H. (2020). Text-based emotion detection: Advances, challenges, and opportunities. *Engineering Reports*, 2(7)
16. Garain, A. & Basu, A. (2019). The Titans at SemEval-2019 Task 6: Offensive Language Identification, Categorization and Target Identification. *Proceedings of the 13th International Workshop on Semantic Evaluation*
17. Pennington, J., Socher, R. & Manning, C. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*
18. Miaschi, A. & Dell'Orletta, F. (2020). Contextual and non-contextual word embeddings: An in-depth linguistic investigation. *Proceedings of the 5th Workshop on Representation Learning for NLP*.
19. Horev, R., (2018). BERT Explained: State of the art language model for NLP. Medium. Available at: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270> [Accessed March 7, 2021].

20. Sanh, V. et al.,(2020). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv.org. Available at: <https://arxiv.org/abs/1910.01108> [Accessed March 7, 2021].
21. Sommeville, I. (2016). Software Engineering, Pearson. 9th ed. (Ch. 2.1.1, pp. 30-32.)
22. McLay, M. (1994). *If Guido was hit by a bus?* Python Legacy mailing list. Online at: <https://legacy.python.org/search/hypermail/python-1994q2/1040.html> [Accessed 19th February 2021]
23. Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 152-158)
24. Nichols, D. (n.d.). Coloring for colorblindness. Online at <https://davidmathlogic.com/colorblind> [Accessed 19th January 2021]
25. Graham, L.(2008). *Gestalt theory in interactive media design*. Journal of Humanities & Social Sciences, 2(1).
26. Soegaard, M. (n.d.). Affordances. *The Interaction Design Foundation*. Online at <https://www.interaction-design.org/literature/topics/affordances> [Accessed 25 February 2021]
27. Anon (2020). What is Soak Testing? Katalon Solution. Available at: <https://www.katalon.com/resources-center/blog/soak-testing/> [Accessed March 5, 2021]
28. Gheorghiu, G. (2005). Performance vs. load vs. stress testing. Agile Testing. Available at: <http://agiletesting.blogspot.com/2005/02/performance-vs-load-vs-stress-testing.html> [Accessed March 5, 2021]