

CS3216: Assignment 3

Confused

<https://confusedsession.vercel.app/>

Group 6

Didymus Ne	A0218159Y
Duan Yu Hang	A0201815R
Hong Ai Ling	A0221725M
Toh Kar Wi	A0217996J
Zhang Ziqing	A0223145R

Table of Contents

Table of Contents	1
Phase 1: Design	3
Milestone 0	3
Milestone 1	3
Milestone 2	6
Milestone 3	11
Milestone 4	12
Milestone 5	15
Phase 2: API Server	16
Milestone 6	16
Phase 3: Mobile Client	17
Milestone 7	17
Milestone 8	20
Milestone 9	27
Milestone 10	27
Milestone 11	28
Milestone 12	30
Milestone 13	36
Milestone 14	39
Milestone 15	44
Phase 4: Coolness Factor	44
Milestone 16 & 17	44

Phase 1: Design

Milestone 0

Describe the problem that your application solves. (Not graded)

While we have seen many classroom web apps (such as Kahoot, or PollEv) promoting *interaction* between the teacher and the student, there are not many prominent classroom web apps that focus on helping the student *communicate* their level of understanding of the material in real-time, especially for in-person lessons; same goes for apps that enable the teacher to receive *feedback* on what the teacher has spoken about *during the lesson*, which would allow the teacher to immediately address any confusion or concern from the student.

Zoom is one tool that can potentially act as a real time communication channel between teacher and student, however it is first and foremost a video conferencing app suited for a *remote learning experience*; using zoom to facilitate an in-person class may not be the most suitable.

Note: Professors and teaching assistants who are our primary target users will be collectively referred to as “teachers” or “instructors” for the rest of the report.

Milestone 1

Describe your application and explain how you intend to exploit the characteristics of mobile cloud computing to achieve your application's objectives, i.e. why does it make the most sense to implement your application as a mobile cloud application?

Description of Application

Confused is a Progressive Web App that aims to help university students inform their classroom teachers when they are confused or clear about the material taught in real-time, without having to face the pressure of speaking in front of their peers. By connecting students to the teaching session using nicknames, they can notify their teacher anonymously when they do not comprehend the material. This provides a communication channel for students who refrain from asking questions due to fear of

what their peers may think of them. Furthermore, teachers are presented with a real-time summary of how many students are confused as well as the topics they're confused about. This helps the teacher to adjust the pace of their lesson, in order to address any concern or confusion raised by the student, improving the learning outcome of the in-person classroom.

Mobile Cloud Computing Choice

While it is common for SoC students to use their laptops during lessons, this observation is not entirely applicable to the wider student community. Students in other faculties often bring only a tablet (e.g. iPad) or phone to their classrooms especially if it's a lecture or seminar style class that does not require much typing. In addition, students may forget to charge their laptops which results in a laptop that has *run out of steam* by the time class starts. In both these scenarios, students will only have access to a mobile device.

Hence, we designed this web app to facilitate a *mobile-first* experience for both the teacher and the student, to increase *accessibility* and *usability* to a wider community of students. Even for students who do bring their laptops to class, having this app on their mobile phone frees up their laptop screen for taking notes and doing research, instead of having to dedicate screen space just for this web app. In this sense, they are able to make use of a normally-dormant screen during class, increasing productivity (just like how having a second screen greatly boosts productivity). In addition, with a mobile-first design, QR codes can be used to facilitate easy joining of the session, whereas a desktop-first design will require users to enter session PINs manually or even have to use an invite link.

For the teacher, while both desktop and mobile use makes sense, we believe that being able to use their mobile phone as a dedicated "question tracker" is helpful. The teacher most likely will be using their computer/laptop for lesson content; having the app on their phone means they can keep it on all the time and keep a close eye on the confusion level of the class as an *accessory*, instead of having to switch back and forth between their PPT slides and the app. Nonetheless, our app is *responsive* and works well on a desktop too.

Exploiting Mobile Cloud Computing Characteristics

1. Utilising the cloud server to run heavy lifting workload so that the mobile web app experience runs well

The real-time aspects of our app necessitates the use of cloud computing to store and transmit the information from student to teacher. By offloading the more-intense heavy-lifting and workload to the server (e.g. generating the questions to display by filtering by session), the web app can consume minimal local resources (e.g. there's no need for the client-side to handle any business logic such as sorting, but rather just handle the display), resulting in a great user experience. After all, nobody likes using an app/website that drains their battery fast!

2. Utilising the cloud to encourage stronger Platform Independence and Faster Built

Since students and teachers have different devices, platform independence is crucial for us to reach a larger audience. Through mobile cloud computing, our app is "serverless" (in the sense where we do not need to manage servers) and can run on any device and operating system. We are thus not constrained by the devices' processing or storage limitations, as all data-intensive processes can run from the cloud. Since we also do not need to manage servers, we can build the app faster and more flexibly.

3. Sharing of information across the Instructor and the Student

Through the cloud, instructor and student views can be updated on the information without the need to "export", "download" or "email". Mobile cloud computing thus lets our Instructor and Student users have a way to securely and quickly integrate information and information updates.

In a nutshell, we exploited the key characteristic of convenience and ease-of-use of mobile cloud computing, and a progressive web app that is mobile first but also scales well to desktop fits this purpose exactly.

Milestone 2

Describe your target users. Explain how you plan to promote your application to attract your target users.

Our target users are 1) Instructors, and 2) Students, with their respective user personas outlined below.

Target User 1: Teacher



Dr Sharon Tan

Lecturer at NUS

About

Age:	35
Gender:	Female
Occupation:	Lecturer
Location:	Singapore
Desire to know students' level of understanding in real time	Medium to High

Behaviours and Needs of Target User 1

- They want to gauge an **overview** of how their students are understanding the lesson material (in terms of whether students are confused or clear) in **real time**.
- They want an immediate record of all the questions students have, without having to disrupt their flow of teaching at any point in time.

Target User 2: Student



Marisa Maldonado Santana

CNM Student at NUS

About

Age: 20

Gender: Female

Occupation: Student

Location: Singapore

Level of Shyness Medium to High

Behaviours and Needs of Target User 2

- They want to indicate to the Teacher that they are **confused** about the teaching material **in real time**, in hopes that the teacher would adjust the pace of lesson for their confusion, **without** having to let others know that they feel confused
- They prefer not to disrupt the overall lesson, despite having a question, or feeling confused.
- They are afraid, or shy, to ask questions during class or in front of the class.
- They prefer to ask questions the moment they *think* about it, instead of having to defer the question to the end after the Teacher has finished teaching

Marketing and Positioning the value of our App

Online teaching via Zoom has introduced newer behaviours to learning that were not evident before (e.g. having a Zoom Chat to be able to ask questions immediately without being blocked by others, and students being able to indicate “YES”, “NO”, or “Go Faster” and “Go Slower”). These indications are only available to **offline learning** today by having to physically disrupt the lesson if the student wants the teacher to

register his/ her reaction or question in real time. Confused thus helps to mitigate this gap for **Teachers and Students**, especially as lessons are “going back” offline.

As students ourselves, we are all too familiar with the problems that exist in the classroom, and we believe we are well-versed to tackle this problem from the ground-up.

Objective

Our goal is to focus on **acquisition** of new users. Given limited resources, our promotion strategy should focus on acquiring **teachers**; after all, by acquiring 1 teacher, we can likely also acquire 20-50 students, who will use the app because their teacher is using it.

In the pilot stage (**Phase 1**), we aim to acquire 2-3 teachers and gather detailed user feedback over a span of 3-4 weeks to improve and iterate upon our app. Thereafter (**Phase 2**), we plan to get at least 1 teacher from each faculty in NUS to use the app, which can help us to plan new features or tailor existing features to faculty-specific needs. Finally, we plan to spread awareness throughout NUS (**Phase 3**) and get as many teachers onboard as possible

Strategy

To achieve the objectives mentioned above, we will take a three-pronged, three-phased approach, outlined below

1. Phase 1~2 - Reach out to School Administration

Work with the school administration and faculty deans to trial and promote the use of Confused. This can include organising an **Ask Me Anything** during NUS Forums or Discussions with the NUS President (or any Key Opinion Leader in the Education space, such as Prof Ben Leong)

Impact

With the school administration convinced of the usefulness of Confused, they will be more likely to promote our app, not only by recommending it to teachers but to also grant us some air-time, be in in an AMA (as above) or via other platforms such as

instagram posts, promotional posters around the school and so on. This makes it easier to coordinate the use of and gather feedback for our app, as we will be doing it officially via the school instead of as a third-party.

Additionally, a Key Opinion Leader (in the Education field) will be able to help increase visibility of our app and convince other teachers to try it out, perhaps at far greater effectiveness than us students would be able to.

Feasibility

The key challenge to executing this promotion would be having the school administration on-board with the app, and be willing to promote its use. To achieve this, we can schedule a live demo to demonstrate our apps' effectiveness and ease-of-use. Given the openness NUS has displayed towards student-run initiatives such as NUSMods, various NUS Telegram groups and bots, as well as student-led activities (such as exam welfare packs and student life fair), we believe that the school will similarly keep an open-mind to a tool which can enhance classroom learning.

2. Phase 1~3 - Offline posters

Large standing posters displayed across areas of school where teachers are most likely to walk past, for example near staff offices. This poster will highlight the "real-time"-ness of the app as well as the problem it solves (that student sentiment in-class is hard to gauge, that students typically don't ask questions out of shyness). Since this is mainly aimed towards teachers, we will also highlight the ease-of-use, for instance how easy it is to create a session and get students to join (via QR code).

Impact

The aim of this poster is to call attention to an existing and unsolved pain point from the teacher's point-of-view. As such, upon first glance, the teacher will be able to resonate with it which draws them in to find out more about our solution. Moreover, as most teachers are already familiar with online learning tools like Zoom, our app will not be a foreign concept and it should be straightforward for us to communicate the value and seamless-integrability of our app.

Feasibility

Putting up posters around the school only requires a minimal investment (to print the posters and have someone to place it down physically); plus approval from the school, which shouldn't be too difficult to obtain. The poster will stay up and remain visible throughout the (marketing) campaign period, and serves as a low-cost-yet-effective way of increasing visibility of our app. People who walk past the poster will minimally take a passing glance, even if out of curiosity, and our app's strong selling points will do the rest.

3. Phase 2~3 - Promotional emails

School-wide email blasts to showcase the key features of our app, which will focus on the unique selling points (USPs) of our app, the real-time aspect and the anonymity of asking questions. Given, most people receive tons of emails everyday and likely do not have time to carefully read through or even open some of these emails. Hence, we will condense the USPs in a one-page virtual pamphlet, designed such that readers immediately understand the core problem that our app can help solve.

Impact

Email is one of the more effective ways of outreach and speaking awareness about our app. Emails sent to school email inboxes (i.e. e1234567@u.nus.edu or matabc@nus.edu.sg) typically have a much higher *email open rate*, as it often contains information useful to school life, unlike personal emails where we receive all kinds of spam and promotional emails. The email campaign is purposely scheduled around Phase 2~3, when we have gathered user feedback from users of the app. This helps ensure that, as much as possible, the User Experience is optimised and painless, which helps to prevent a situation where a user tries the app, finds it to be clunky or not-very-useful, and then disregard it from that point on. In other words, we want to make sure that our app is polished and ready for primetime before marketing it more extensively

Feasibility

It goes without saying that emails cost almost nothing to create and send; and have a potential to achieve school-wide outreach. Initial investment may be minute, but the potential gains are immense.

Although a majority of people receiving the email blasts will be students, we believe that outreach to students will be beneficial, as they can potentially inform the teacher

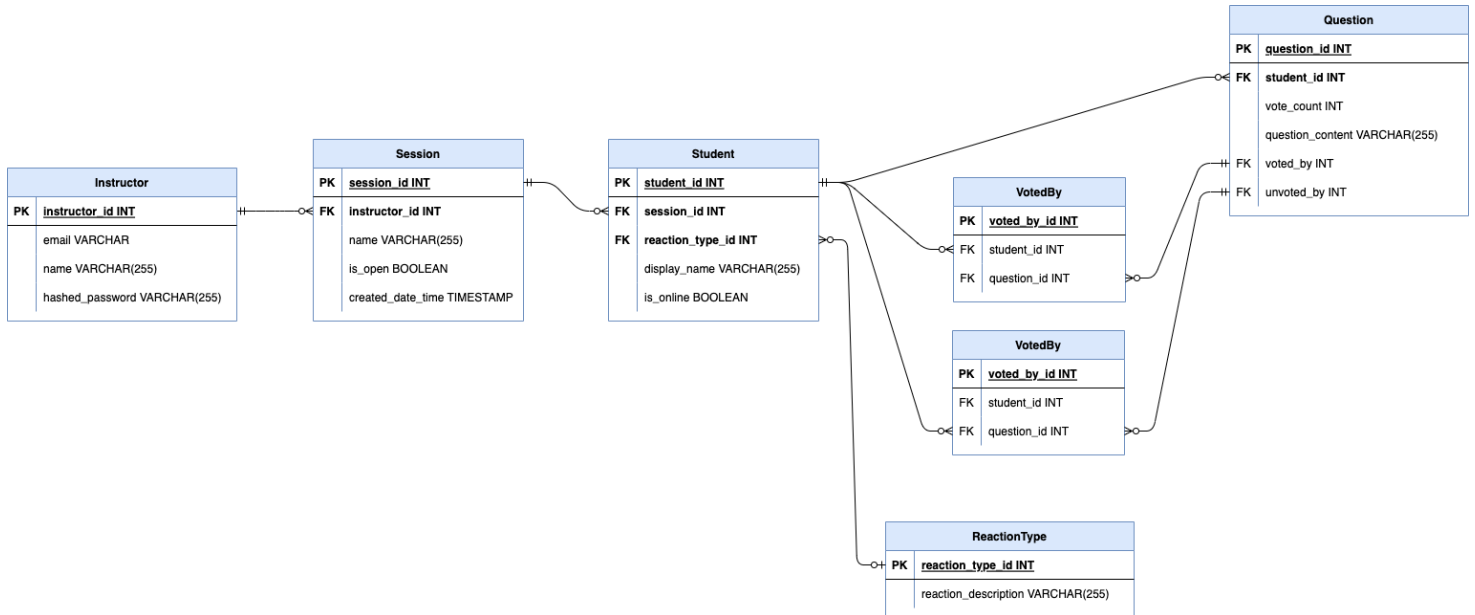
on this new tool. In addition, undergraduate and graduate teaching assistants can potentially pick up this new tool and apply it in their tutorials, which can then induce the professor/module coordinator to start using it as well. Not to mention that teachers will definitely be receiving our email, and we believe that our value proposition is strong enough to hook them in.

What comes after

For user retention purposes, we will not utilise promotion strategies, because we think Confused will need to focus more on acquiring **new users** given that it is still early in its product life cycle, and is trying to find its product-market fit. Instead of promotion strategies to retain users, our strategy would be to utilise a consistent iterative product, making use of Google Analytics, elaborated more in Milestone 14. We believe that once users (especially teachers) start using the app and seeing its benefits, they will naturally build a habit of using this app for their classes, just like how Zoom was a staple for the past 2 years.

Milestone 3

Draw an Entity-Relationship diagram for your database schema.



Milestone 4

Explore one alternative to REST API (may or may not be from the list above). Give a comparison of the chosen alternative against REST (pros and cons, the context of use, etc.). Between REST and your chosen alternative, identify which might be more appropriate for the application you are building for this project. Explain your choice.

In this milestone / segment, we evaluated our design of APIs by dividing the app into two core sets of capabilities, and exploring the design that we felt best suited each set of capability. The two core sets of capabilities are:

1. Basic CRUD operations on the Instructor / Student
2. Real-Time In Session capabilities and communication between the Instructor and the Student

Evaluation for Capability 1: Basic CRUD Operations

The key features that we require from the Basic CRUD operations for our MVP are as follows:

1. Create an Instructor Account
2. Read details of an Instructor Account
3. Update details of an Instructor Account
4. Create a Session
5. Read details of the Session (note: this does not refer to the **in-session** details, which is separate)
6. Update details of a Session
7. Delete a Session
8. Create a Student and Join an existing Session

Context of use	REST	GraphQL	Better for CRUD in Confused?
Architecture focus	Server-driven	Client-driven	Equal
Message format	Can support multiple data formats	Returns the data in a JSON format	Equal

Type	Weakly typed	Strongly typed	GraphQL
Scalability	Easy to scale software through REST, because of the decoupling of the client and server (clients and servers can scale independently of each other)	Queries are executed within the context of a particular system	REST
Performance	REST APIs can encounter the problem of <i>overfetching</i> and <i>underfetching</i> , so the client-side needs to filter out unnecessary data, or make multiple requests to stitch the responses together	Being a query language, GraphQL is more flexible in fetching data from the exposed endpoint, so client can fetch the <i>exact data</i> it needs without over or under fetching.	GraphQL
Communication / Error reporting	Communicates via HTTP status codes to inform on success or failure / erroneous API operations	Always return HTTP 200 (OK) status codes, regardless if the query is successful or not	REST
Caching	Support	Difficult to support	REST
Learning Curve	Easy	Less easy	REST

After our research and comparison, we ultimately felt that REST is more appropriate for the app that we are building for the following reasons:

1. Better decoupling between Client and Server

Since this app we are building has multiple clients (the teacher and the student), we want to focus on decoupling the client and the server, so that clients do not need to know the implementation details from the server. With an independent server, we felt

that it would be easier to scale upon this server and build multiple clients. Hence, in this context, REST allows for easier adaptability to GraphQL, therefore we chose REST.

2. Error Identification

REST has an advantage over GraphQL in communicating errors, as REST is built upon HTTP and uses HTTP Status codes to identify errors, while GraphQL returns a HTTP status code 200 (OK) for every request. This makes error identification in GraphQL more complex than in REST. Given our short timeline, we felt that REST would be a better option as it is easier for the communication of errors.

3. Simple Application

In capabilities 1, our functional requirements are mainly basic CRUD operations. Given the simplicity of our core set of functional requirements, we felt that REST helps us not to overcomplicate the development of the application, especially since we do not foresee the need for much nested data to be fetched, and that our *basic* operations mainly involve CRUD operations.

Evaluation for Capability 2: Real Time Transmissions

To evaluate our choice of APIs to support Real Time transmissions, we analysed WebSocket and Long Polling.

Characteristics of WebSocket API vs Long Polling

Context of use	WebSocket	Long Polling	Better for RealTime?
Number of calls required to constantly get updates	1 to establish connection, another to disconnect	Multiple GET requests (after a period or if data is received)	WebSocket (because the cost of communication is lower)
Load / Scaling	Performance of WebSocket is better with high loads	Performance is challenging to maintain with high loads	WebSocket
Communication	Bi-directional	Uni-directional	WebSocket

			because clients can perform actions in a single communication, and still receive updates
--	--	--	--

Given our evaluation metric, we decided to go with WebSockets for the following reasons:

1. One call to the server (WebSocket) vs multiple calls to the server (REST)
 - a. Less costly and more performant
2. WebSocket is more ideal for our “in-session” scenario where high loads are required since there are many students in one class.
3. Bi-directional communication

Thus, our API design chosen for Confused includes:

- 1. REST**
- 2. WebSocket**

Milestone 5

*Design and document all your REST API. If you already use Apiary to collaborate within your team, you can simply submit an Apiary link. The documentation should describe the requests in terms of the triplet mentioned above. **Do provide us with an explanation of the purpose of each request for reference.** Also, explain how your API conforms to the REST principles and why you have chosen to ignore certain practices (if any). You will be penalised if your design violates principles for no good reason.*

Our API document can be found here: <https://confused.docs.apiary.io/#>

Phase 2: API Server

Milestone 6

Share with us some queries (at least 3) in your application that require database access. Provide the actual SQL queries you use (if you are using an ORM, find out the underlying query and provide both the ORM query and the underlying SQL query). Explain what the query is supposed to be doing.

Reading sessions belonging to an instructor:

Django ORM - ViewSet	<pre>self.queryset.filter(instructor=self.request.user).distinct() .order_by("-id")</pre>
SQL	<pre>SELECT DISTINCT * FROM Session WHERE instructor_id = 1 ORDER BY id DESC;</pre>

Creating a session:

Django ORM - ViewSet	<pre>serializer.save(instructor=self.request.user)</pre>
SQL	<pre>INSERT INTO Session (instructor_id, name, is_open) VALUES (1, "CS3216", FALSE);</pre>

Updating a session:

Django ORM - ViewSet	<pre>self.update(request)</pre>
SQL	<pre>UPDATE Session SET name = "CS3217" WHERE id = 1;</pre>

Deleting a session:

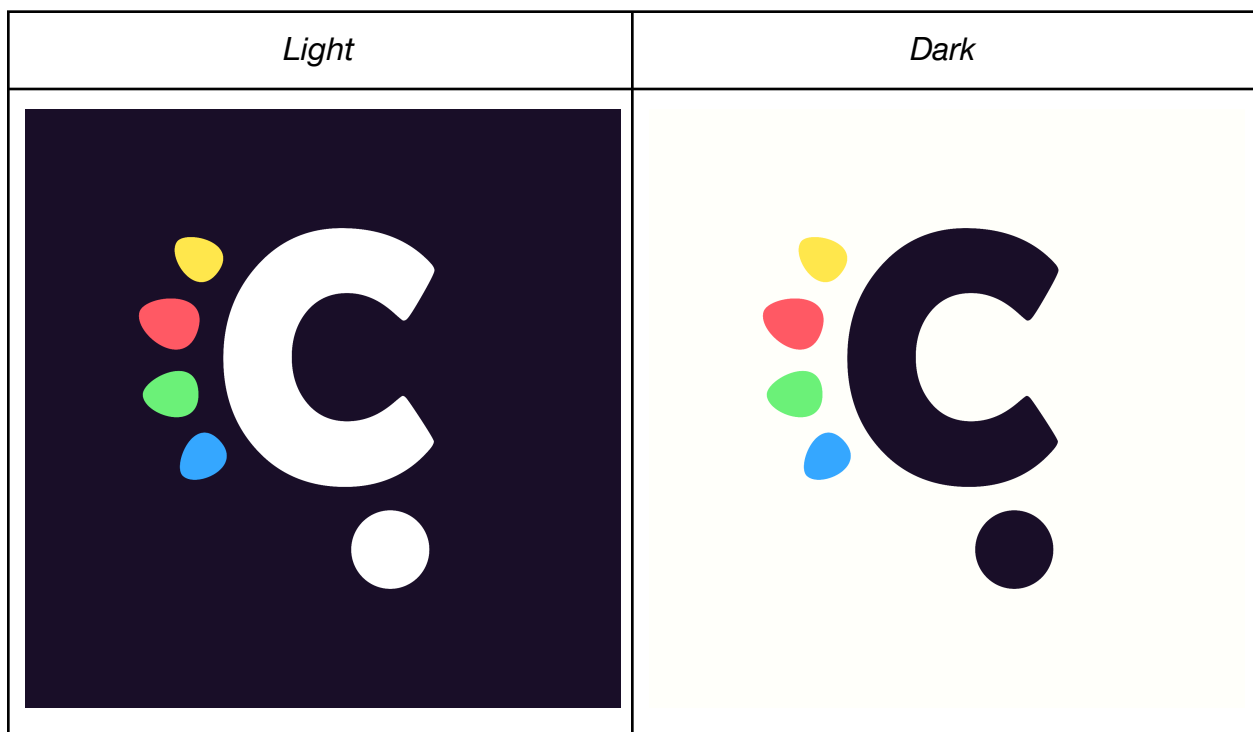
Django ORM - ViewSet	<code>self.destroy()</code>
SQL	<code>DELETE FROM Session WHERE id = 1;</code>

Phase 3: Mobile Client

Milestone 7

Create an attractive icon and splash screen for your application. Try adding your application to the home screen to make sure that they are working properly. Include an image of the icon and a screenshot of the splash screen in your write-up. If you did not implement a splash screen, justify your decision with a short paragraph. Add your application to the home screen to make sure that they are working properly. Make sure at least Safari on iOS and Chrome on Android are supported.

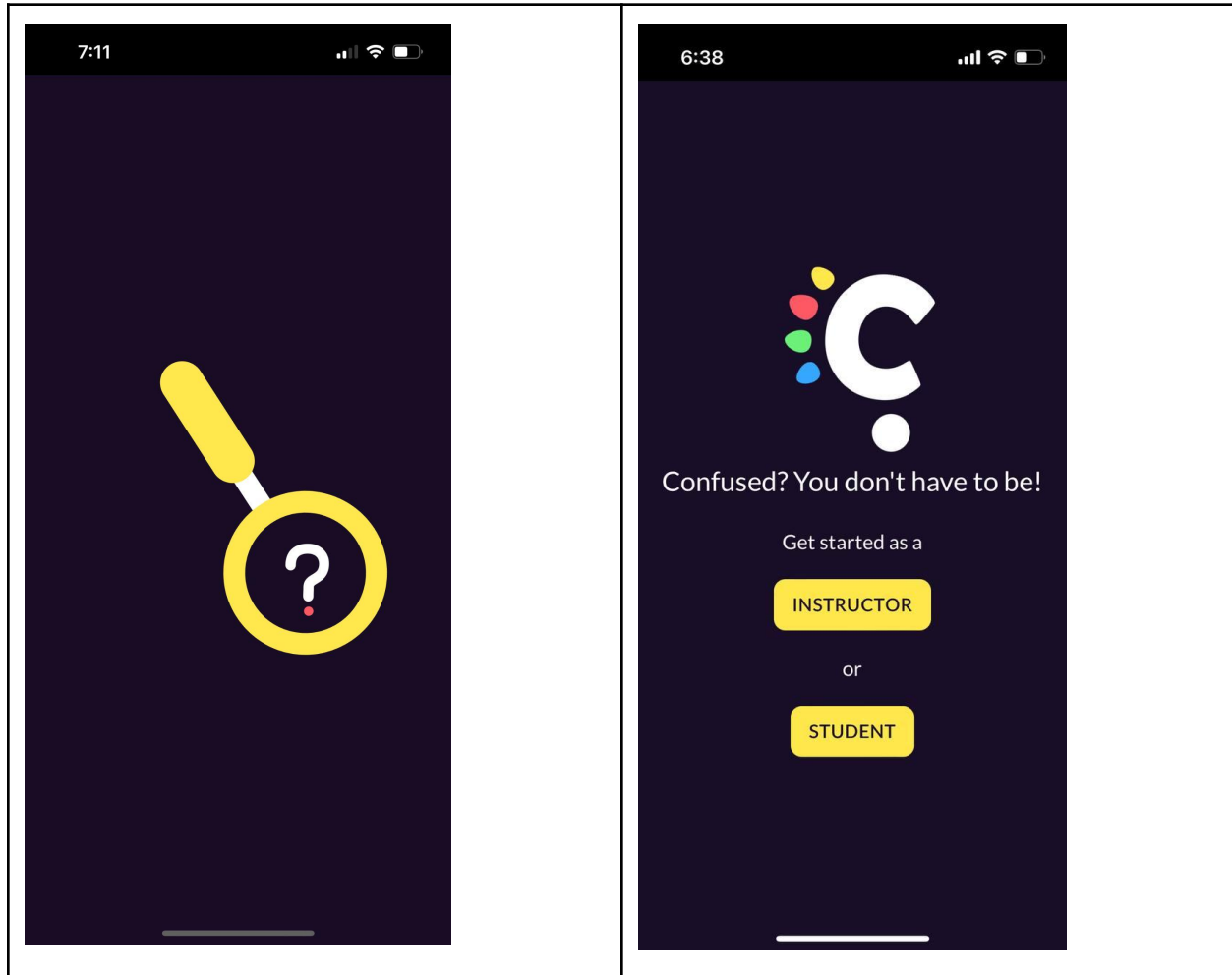
Our app icon is as follows:



Since the name of our app is Confused, we think emphasising the starting alphabet “C” helps users to immediately **recognise** the app. More importantly, we designed the icon to resemble a question mark, in order to communicate the app’s **purpose** - to be the “go-to” app for expression confusion and asking questions during a class, as well as ultimately getting those queries addressed by the teacher.

We then added a palette of spirited and lively colours – blue, green, red and yellow – to portray the app as a vibrant and safe platform for students to ask questions. The app itself similarly follows this palette of colours in its design for a consistent look.

Our splash screen is as follows:



We decided to include a splash screen with animations to communicate through our use of the Magnifying glass and the Question Mark that learning can be fun, even if confusing. More details regarding the animation can be found in **Milestones 16&17**.

Milestone 8

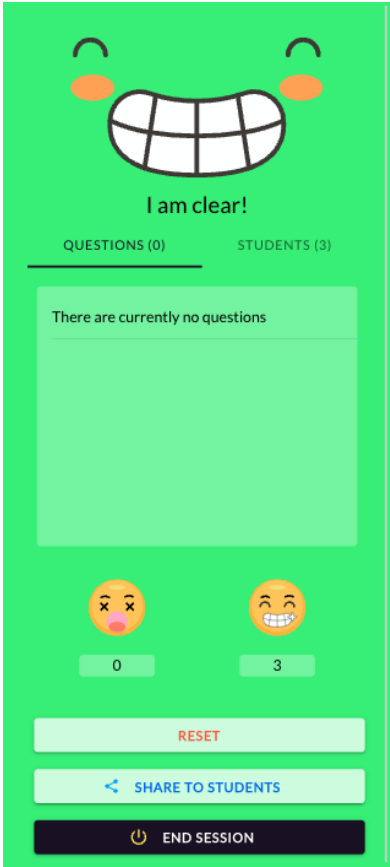
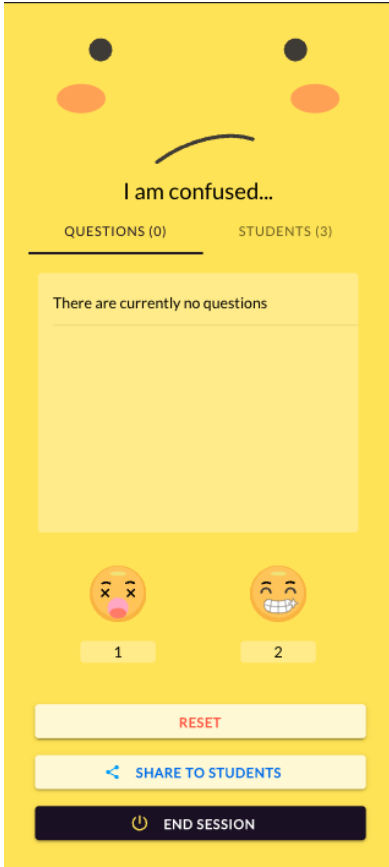
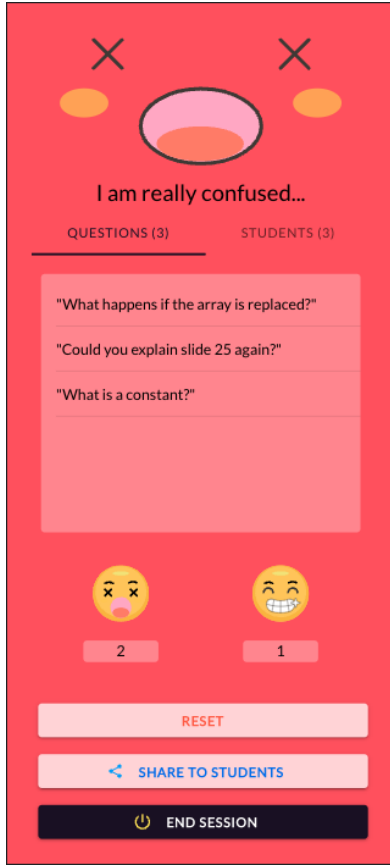
Style different UI components within your application using CSS in a structured way (i.e. marks will be deducted if you submit messy code). Explain why your UI design is the best possible UI for your application. Choose one of the CSS methodologies (or others if you know of them) and implement it in your application. Justify your choice of methodology.

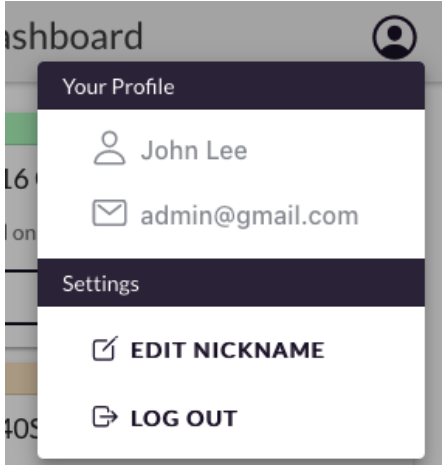
A. Justification of UI design

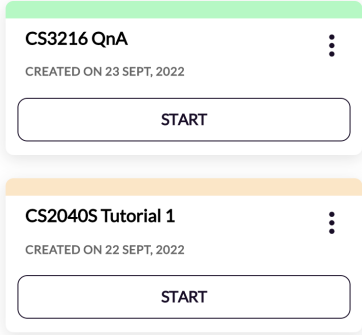
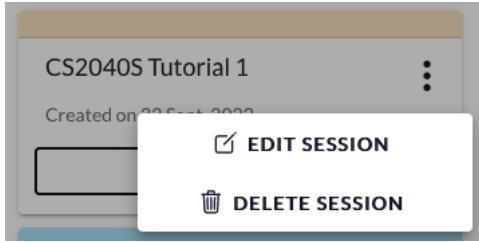
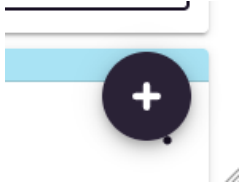
A.1.Instructor

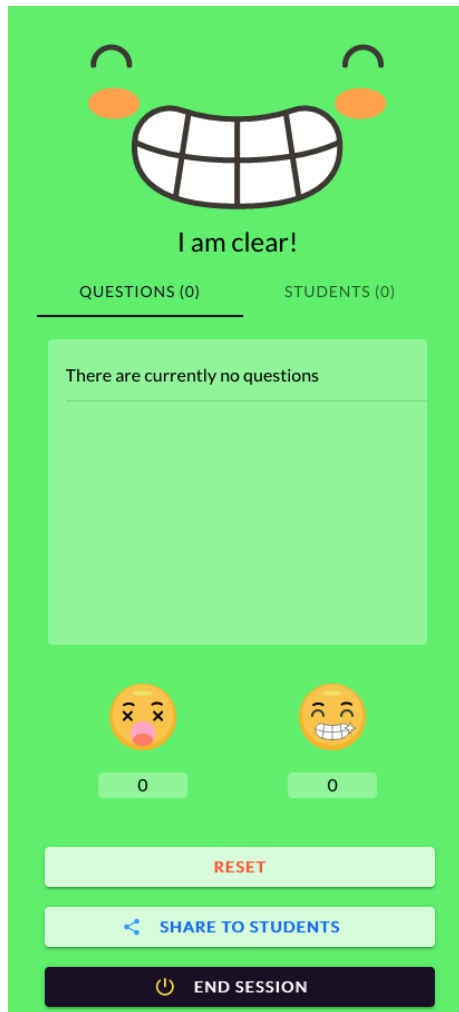
Our UI design was focused on designing for the **user experience**. To improve the in-person learning experience, we added a touch of **gamification**.

Instructors are more incentivised to “keep their screens green”, since it indicates a clear lesson with no confusion amongst students.

CLEAR	CONFUSED_1	CONFUSED_2
		

UI Design	Rationale
<p>User Menu</p> 	<p>Clicking on the profile icon gives details of the instructor's personal details such as displayed name and email. It also intuitively provides functionality of editing nicknames and log out.</p> <p>Moreover, by separating the profile and settings into two sections, it presents the information in an organised manner that is in line with our theme colour.</p>

<p>Session Card</p> 	<p>The name of the session and the date of creation is displayed, as these are the two most crucial pieces of information that need to be shown to an instructor (as opposed to other non-essential information such as number of students in the session, number of reactions etc.). The start button was also designed to be easily accessible and clickable as that is the primary action for this page.</p>
<p>Edit/delete Session</p> 	<p>The Edit and Delete actions were placed in a separate menu as these actions are not commonly taken, and will also cause a significant change to the session. Placing it on the session card directly might lead to unintended misclicks that ultimately result in user frustration.</p>
<p>Create Session Button</p> 	<p>By using a floating action button as the entry to creating a session, it makes sure that instructors could create a session no matter where they are scrolling to in the dashboard. It is also easy for a phone user to click on, helping them to create a session easily.</p>
<p>Create Session Form</p> <p>What is your session name?</p> <p>Fill in a descriptive name for your session!</p> <input data-bbox="228 1436 664 1503" type="text" value="e.g. CS1101S Studio Group 8"/> <div data-bbox="535 1524 662 1577">SUBMIT</div>	<p>The submit button is arranged at the bottom right corner compared to the text box. This suits users' habit of writing texts from left to right and then submitting on the right hand side.</p>
<p>Instructor In Session - Question</p>	<p><u>Top: Big Emoji</u></p> <p>Together with the background colour, this gives users a quick impression of how much students are confused during a lesson. As instructors rarely have time to look into details at a screen, such UI</p>



quickly signals the most important information - are the students confused - to them.

Middle: Question / Student Tab

The switch of the tab is by default displaying questions from students. The tabs allow instructors to have more nuanced information from students such as how many students are in the session and what questions they have currently.

Middle: Emoji and Count

The emoji and count below provides a detailed breakdown of student reactions at a particular time. Placing them side by side helps instructors to quickly compare the ratio of confused students with those indicated they are okay.

Bottom: Button groups

The below part places important settings that are within the reach of their fingers. The positioning helps users to conveniently control the session. The colour and icons of the button help users to differentiate and comprehend the meaning of each action. For example, using a black background for "END SESSION" signals the difference and seriousness of this action. Moreover, placing the "RESET REACTION" button the furthest away from "END SESSION" prevents mistakes in pressing wrong buttons.

The Sharing Page

The sharing page is a sliding up window in the ongoing session, as this allows instructors to dismiss the window easily by sliding down and implies that the session is still ongoing.

Below the title is the session code, where the code is bolded so that it is clear enough.

The QR code is the most prominent element in the page, as it is one of the core features that makes Confused easy-to-use. The size is also



Share the session

Session code: **305**



Scan this QR code via the student page, or with your preferred QR scanner

<https://confusedsession.vercel.app/student/session/305>

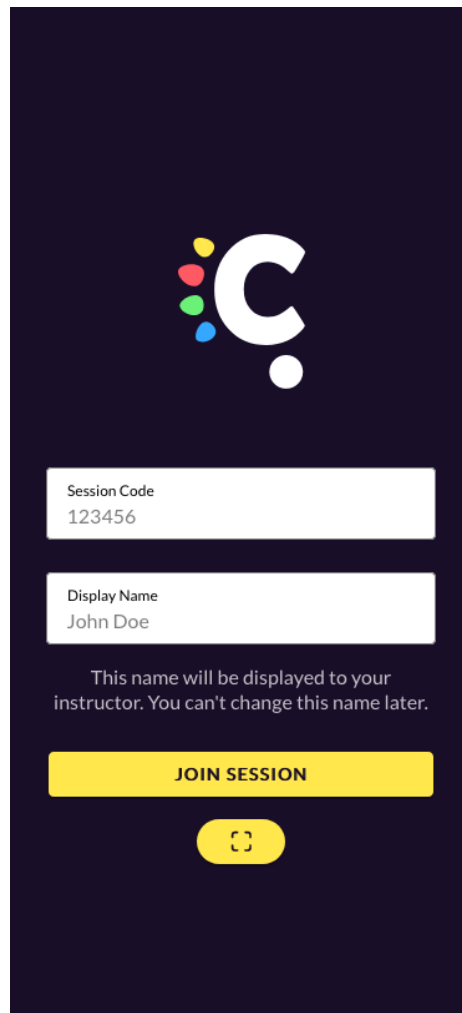


sufficiently large such that students can scan the QR code even from a distance away.

Besides that, we placed a “Copy URL” button such that the teacher can conveniently send the link via Zoom or email if they so wish.

A.2.Student

Student Join Session Page



The image shows a mobile app interface for a student to join a session. It features a dark purple background. At the top center is a logo consisting of a white 'C' with four colored dots (yellow, red, green, blue) to its left and a white dot below it. Below the logo are two white input fields. The first field is labeled 'Session Code' and contains the text '123456'. The second field is labeled 'Display Name' and contains the text 'John Doe'. Below these fields is a line of text: 'This name will be displayed to your instructor. You can't change this name later.' At the bottom of the form is a yellow rectangular button with the text 'JOIN SESSION' in black. Below the button is a yellow rounded rectangle containing a QR code icon.

Session Code
123456

Display Name
John Doe

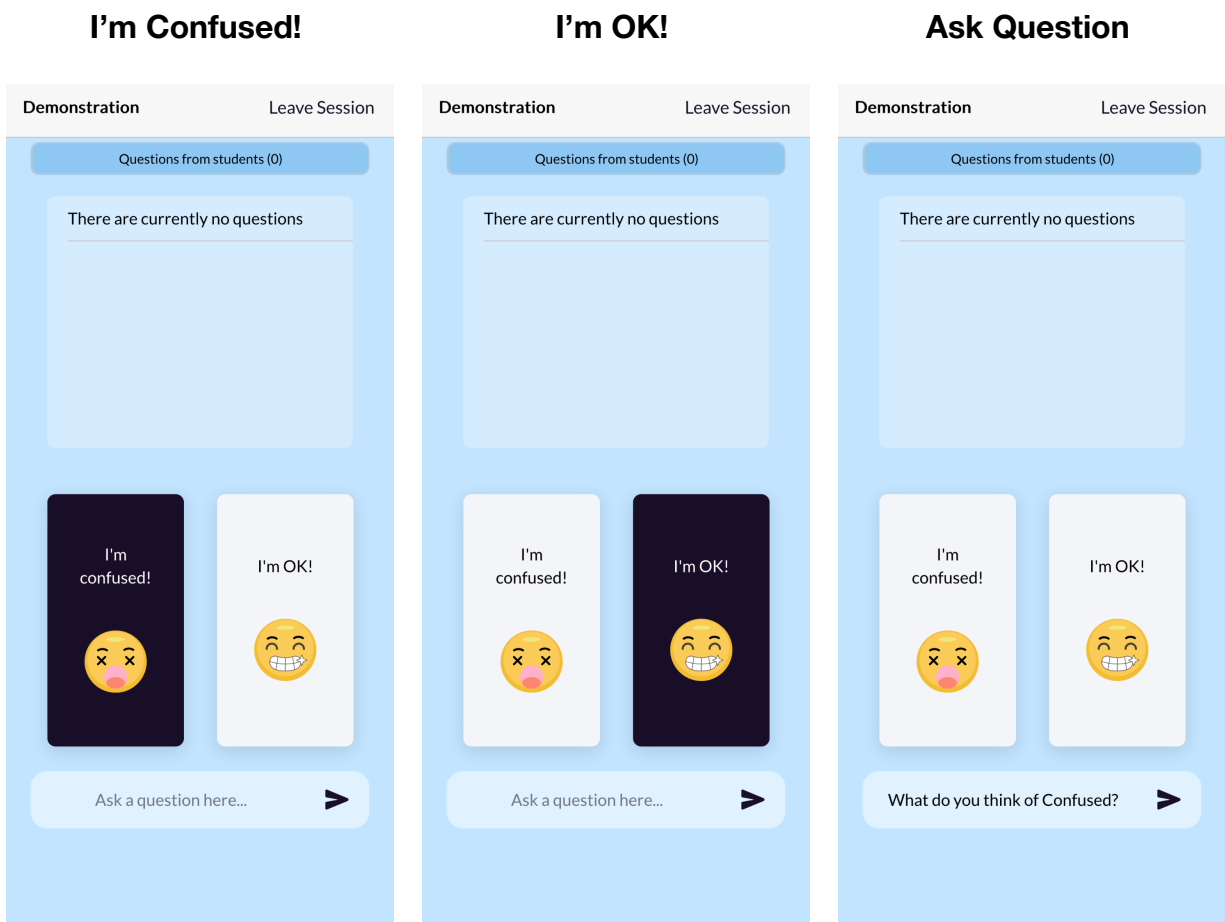
This name will be displayed to your instructor. You can't change this name later.

JOIN SESSION

QR Code

The join session page for students uses a dark background to attract them with the gamified feeling. The action buttons are placed at the bottom, making it easy for them to join the session via QR code as well.

Student In Session



As we can see above, the reaction buttons are placed at the area that is the most comfortable for students to click using their hands. The reaction button is big enough for them to click easily. Moreover, placing the asking question text box at the bottom of the page uses the familiar layout of social media messaging apps (such as Telegram), creating an easy feeling for the UI. In addition, the "LEAVE SESSION" action is placed at the top right corner as that is not the hot area of a phone, preventing students from leaving a session accidentally.

The simple and quick way to tap on the reactions in the Student In Session page mimics a GameBoy device, which would reduce the barrier to voice out one's concern and encourage students to be more open to sending feedback. To this front, the reaction buttons are designed to be large and are a prominent aspect of the page design.

B. Choice of CSS Methodology: BEM

BEM

- BEM provides a modular structure. Because of its unique naming scheme, we won't run into conflicts with other CSS names (Decreased likelihood of class name overlap are some of the biggest benefits of using BEM)
- BEM also provides a relationship between CSS and HTML. Ambiguous names are hard to maintain in the future.
- Easy to use class names and reduction in deep CSS selectors.
- One great thing about BEM is specificity control. By using the naming convention stated above, we are automatically flattening our specificity to .class single selectors, or [0 0 1 0].

Easier to quickly understand what is going on with BEM. All these help make BEM helpful for debugging. Hence we chose BEM.

Object-Oriented CSS

- Separate content from container, which allows for one React component to have one css file
- Reusable visual styling codes, location flexible codes, reduction in deep nested selectors.
- Without a fair amount of repeating visual patterns, separating structure and visual style codes seem unnecessary.
- Common benefits of design principles, reduce cost of debugging
- Reduced dependency across other components
- Negatives: can lead to inconsistency across components
- Very prone to conflict across different components (e.g. button colour)
- Overkill for small projects, very tedious because need to define for every single component

SMACSS

- Although SMACSS is regarded as more scalable, our team thinks that it is more complicated compared to BEM, adding on to the learning curve. Therefore, for easier maintenance in the future, SMACSS might not be the best choice.

Milestone 9

Set up HTTPS for your application, and also redirect users to the https:// version if the user tries to access your site via http://. HTTPS doesn't automatically make your end-to-end communication secure. List 3 best practices for adopting HTTPS for your application.

1. Enforce proper key management by keeping private keys of the server securely. Private keys should be generated in a secure environment such as during deployment. When the certificate is no longer safe to use, it should be revoked.
2. The server should be configured with secure cryptographic algorithms and up-to-date SSL/TLS versions which minimises the risk of security vulnerabilities. In addition, tools such as TLS server scans can help audit server configurations and identify poor practices
3. Implement a SSL expiration monitoring tool to maintain reliability and safety of the website data; such tools can help automate the renewal of SSL certificates that are close to expiry. Most of the Certificate Authority (CA) supports auto-renewals of certificates to ensure a new certificate is installed before the old one expires. Not doing so can have drastic consequences, at best blocking access to certain parts of the app/website and at worst resulting in data breaches that cause sensitive information to be leaked, such as in the [Equifax case \(2017\)](#) which resulted in the personal data of 148 million users being accessed illicitly.

Milestone 10

*Implement and briefly describe the **offline functionality of your application**. Explain why the offline functionality of your application fits users' expectations. Implement and explain how you will keep your client synchronised with the server if your application is being used offline. Elaborate on the cases you have taken into consideration and how they will be handled.*

When user is offline

Users (Instructors and Students) will receive a toast alert when they are offline and when their network connection recovers, so that they are aware of the network connection status.

When instructor disconnects

When instructors are offline, they will still be able create, update and delete sessions, and edit their nickname. When they return online, their activities will be synced up with the online server.

This is achieved by storing queues of POST, PUT and DELETE requests in local storage when users are offline. When users reconnect, the request generated during the offline period will be processed in sequence from the queue and subsequently deleted from local storage.

Moreover, when creating a session, a session id is not known to users until the response to POST request is sent back. To address this problem, we have also locally stored a list of "fake ids" for sessions created offline, which will be updated to real ids when users reconnect. The list of "fake ids" takes the maximum of existing session ids plus 1000 to ensure any fake id does not clash with real session ids. This design ensures the usability of the app while following the POST request format for session.

When student or instructor disconnects from an ongoing session

When a Student faces latency issues due to wifi and goes offline while **in a session**, they can still update their reactions or send questions. Once the wifi connection recovers, the reaction updates or new questions will be registered and updated. This offline persistence is also applicable to the Instructor In Session.

This offline persistence is due to the WebSocket connection established. WebSockets will keep the unique connection open while eliminating latency problems with long polling.

Milestone 11

Compare the advantages and disadvantages of token-based authentication against session-based authentication. Justify why your choice of authentication scheme is the best for your application.

Session-based authentication

The server maintains user state, which can help in user personalisation and keeping track of user activity. They also mitigate the risk of client-side Javascript from

manipulating behaviour. In addition, cookies do not take up much space and are simple to store.

However, this method is vulnerable to cross-site request forgery attacks, where malicious users can make use of the cookie stored in the user's browser to carry out illicit operations. Another disadvantage is that all the sessions have to be stored in the server's memory, which increases strain on the server, especially if large numbers of users are concurrently using the app.

Token-based authentication

Users do not have to send their credentials over and over to the server. Moreover, servers that use tokens can enjoy improved performance, because they do not need to continuously look up session details to authorise the user's requests. In addition, servers also do not have to maintain a long list of sessions since they simply need to generate and verify tokens (which are stored on the client device). Tokens are also more secure as the token has to be attached to each request for the server to accept the request.

However, the authentication details are stored on the client, so the server cannot perform certain security operations as in the session method. The server does not authenticate the user, so linking a token to its user can be more difficult. If a hypothetical attacker manages to get a valid token, they may have unlimited access to the server databases. If the server generates keys using older algorithms, these keys can be breached.

What makes sense for Confused

For Confused, we believe token-based authentication makes the most sense, because in a session, there can be up to a few hundred students connected simultaneously. Multiply this by the number of concurrent lectures all around NUS (and even other universities), and you get a large number of sessions that need to be maintained (if we go with session-based auth). As mentioned above, this can put a large strain on the servers. Tokens are simply much more scalable and put less strain on server-side hardware.

Milestone 12

Justify your choice of framework/library by comparing it against others. Explain why the one you have chosen best fulfils your needs. Lastly, list down some (at least 5) of the mobile site design principles and which pages/screens demonstrate them.

A. Benefits of Ionic

We chose Ionic framework because it provides a native look and feel, that also implements native functionality and interactions. This minimises the development time as we can share a single codebase for all platforms (with only very minor platform-specific tweaks required), while making users feel like they are using a real native app. Not only so, the web-based nature of Ionic means that standard HTML, CSS and even Javascript can be used to provide styling and functionality. In addition, we can make use of the huge assortment of React libraries and components, which is phenomenal as React has been (and still is) the most popular front-end framework in recent years.

Comparison to Native & Cross-platform-native

Native is obviously the most optimal in terms of performance and achieving an overall feel congruent with the rest of the OS. However, it requires maintenance of separate codebases for different platforms, not to mention the web-based implementation that is necessary if we want to enable desktop use.

On the other hand, cross-platform-native frameworks like React Native and Flutter are incompatible with web, similarly requiring a separate web-based implementation for desktop use. React Native also does not feature great support for animations, which goes against one of our initial design principles of using animations to gamify the experience. For Flutter, it requires learning the Dart language, which is too time-consuming for the purposes of this project.

Obviously, neither of these types of frameworks can satisfy our constraints of limited development time and lack of experience for some of the languages/platforms (e.g. Dart, Kotlin etc.)

Comparison to other cross-platform web frameworks

Ionic is one of the most popular cross-platform web frameworks as compared to alternatives such as Framework7 and Vuetify, which means that it is easier to obtain

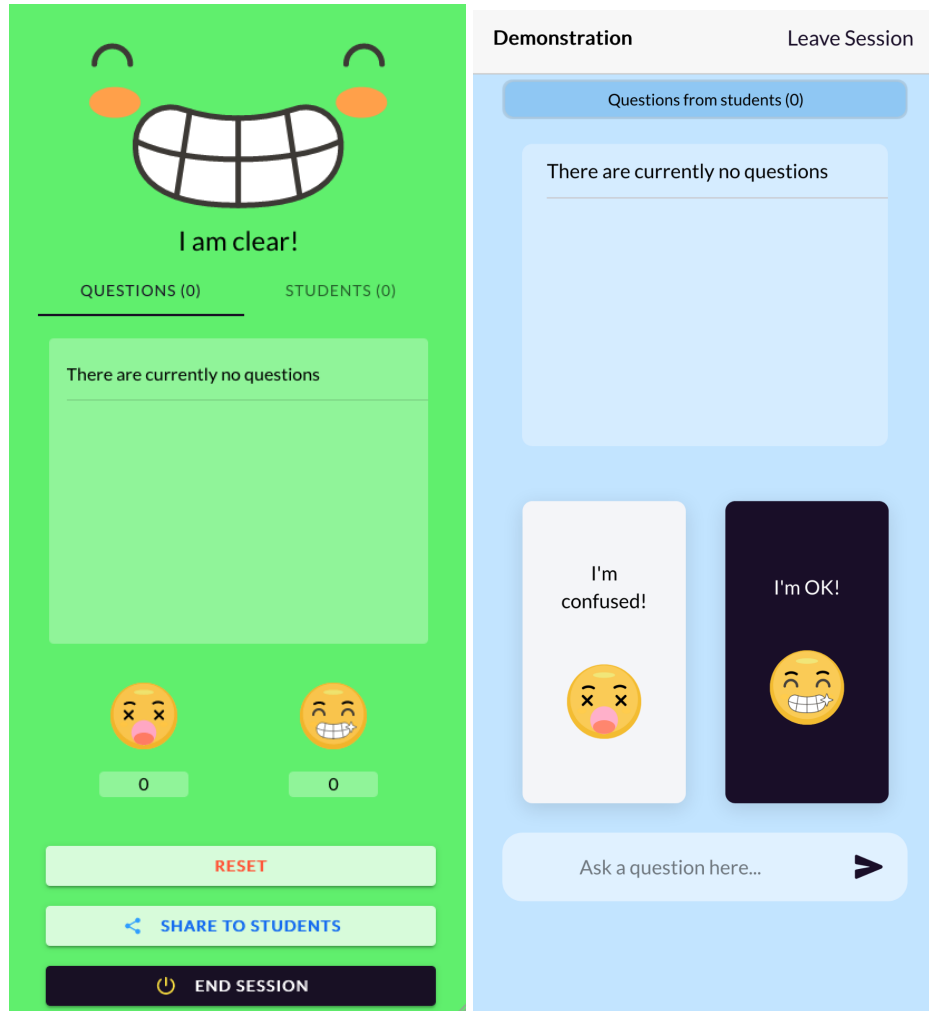
help for and find solutions to problems faced during development. In addition, its compatibility with Capacitor and Cordova provides a gateway to the vast selection of plugins which helps to add functionality to the app; this level of versatility is one of the best, if not the best amongst cross-platform web frameworks. Its active development and strong community ensures that new features are actively implemented, bugs are regularly caught and squashed, and compatibility with external libraries and frameworks (such as Capacitor and React) will be maintained.

Material UI is another strong contender, however it heavily draws upon Google's Material Design principles, which may not be a good fit for iOS; Ionic on the other hand has components that adapt to the platform and is more optimal at providing a native feel. Material UI is also more of a component library than a framework, which means that it is easier to add functionality while remaining performant using Ionic than Material UI. In addition, there have been reports of Material UI having poor documentation on custom style components as well as poor performance with long lists. Ionic, on the other hand, is known to have comprehensive and in-depth documentation, which reduces the time spent *pulling our hair out*.

B. Mobile site design principles

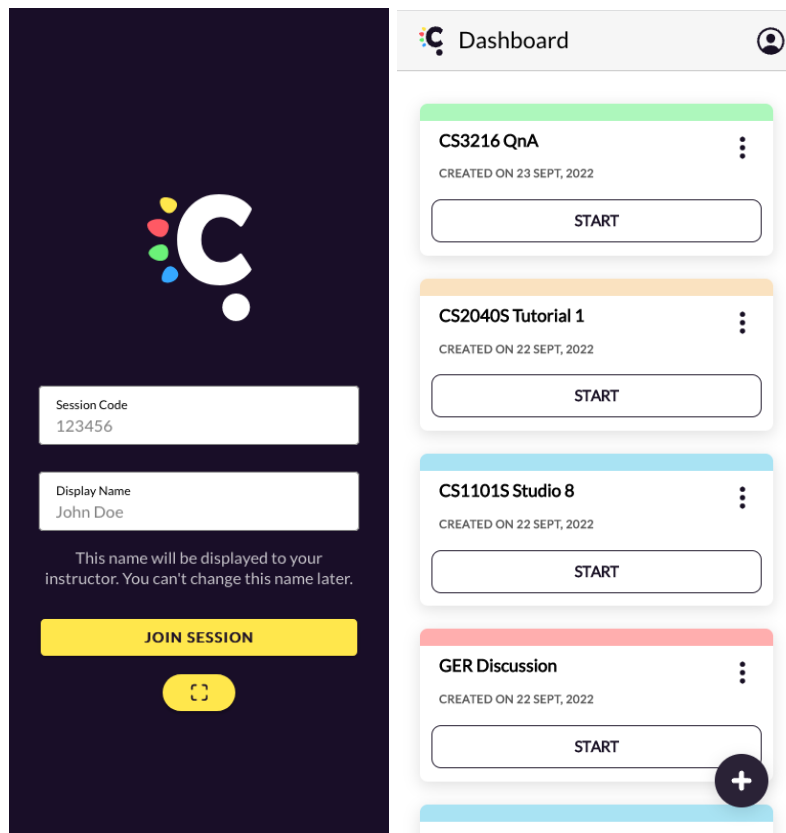
1. Simple page layout

Single column layout with no horizontal scrolling necessary, and free from distracting content



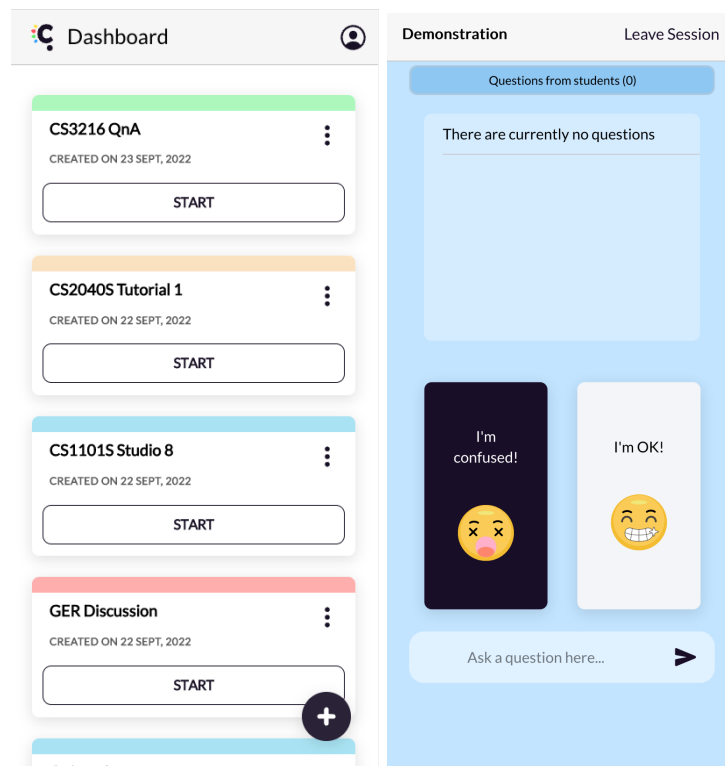
2. Readable text with large touch targets

Font size is readable on mobile screens, and buttons are large enough for fingers to tap on. As shown below, the JOIN SESSION button and the START button are big enough for fingers' convenience.



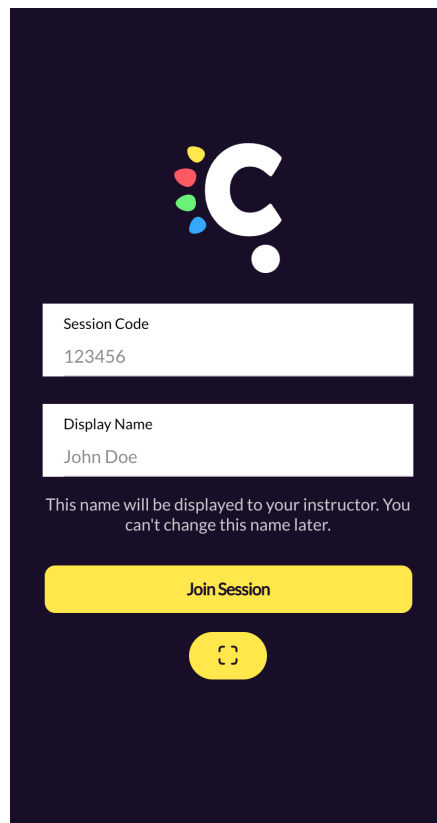
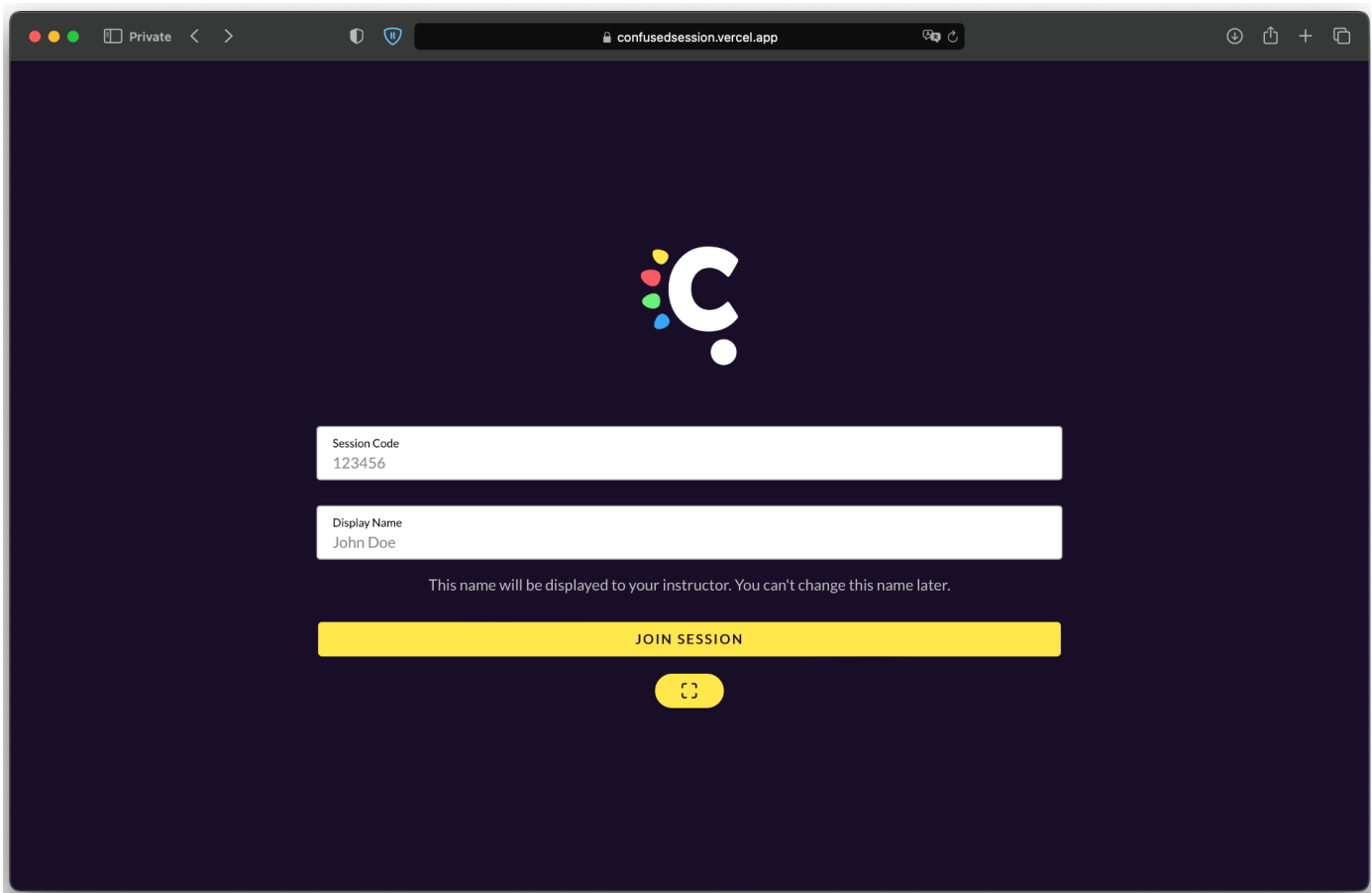
3. Designing touch targets with the “thumb rule” in mind

Most people use their thumbs to interact with their phones, which means clickables near the middle are far easier to access than clickables at the corners. We designed commonly-clicked components to be near to middle, while clickables that are more “final” and not commonly clicked (e.g. “Leave Session”, or the Profile) are placed further away from the middle.



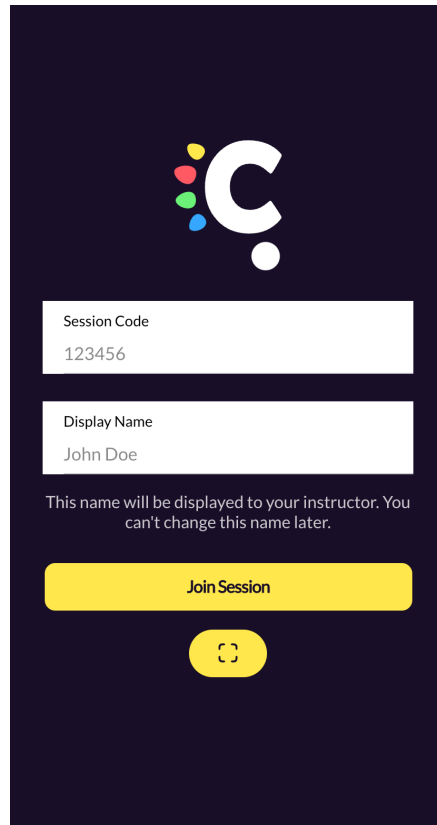
4. Visual and functional consistency between mobile and desktop

Desktop and mobile sites should have a similar look and feel. Users must be able to achieve the same things on mobile as they can on desktop.



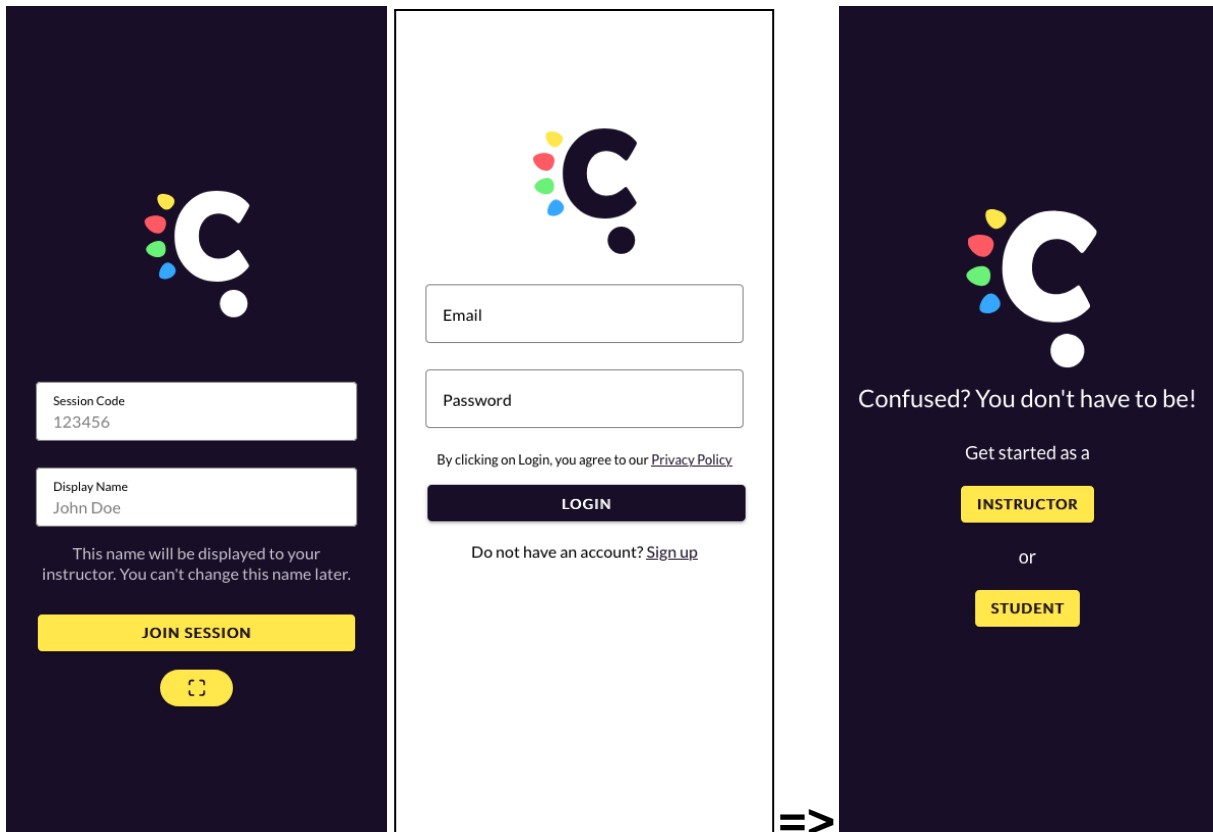
5. Simplify input fields and introduce efficient data entry

Filling out detailed information on mobile is less convenient than on desktop. Hence, we included the “Scan QR” functionality so users do not have to enter the session code manually.

A mobile app login screen with a dark purple background. At the top center is a logo consisting of a white 'C' with four colored dots (yellow, red, green, blue) to its left. Below the logo are two white input fields. The first field is labeled 'Session Code' and contains the text '123456'. The second field is labeled 'Display Name' and contains the text 'John Doe'. Below the second field is a line of text: 'This name will be displayed to your instructor. You can't change this name later.' Below this text is a yellow button with the text 'Join Session'. At the bottom center is a yellow button with a QR code icon.

6. Make it easy to get back to the homepage

Users tend to click on the logo of a mobile site when they wish to return to the home page, so we added links to navigate to the homepage. When users click on the logo in the two pages below, they will be directed back to the Start screen



(From left to right) Student join session page, instructor sign up page, homepage

Milestone 13

Describe 3 common workflows within your application. Explain why those workflows were chosen over alternatives with regards to improving the user's overall experience with your application.

Workflow 1: Instructor checks students' confusion level

On the dashboard, the instructor can start any session that has been created by tapping on the “start” button. This leads them to the instructor-in-session page and sets up the session immediately. The instructor can then share the session with the students by clicking on the “share to students” button. Once students join the session, they can express their confusion at any point of time. The instructor can gauge students' level of confusion via 2 ways: colour of the screen and the reaction statistics. If all students are clear, the screen will appear green with a happy face. If at least one student is confused in class, the screen will appear yellow along with an unhappy face. If more than half of the class are confused, the screen will appear red with a defeated face. Besides, the instructor can also look at the number of “confused” and “clear” reactions shown below their respective emojis. Once the instructor has clarified the confusion of the students, they can tap on “reset” to reset all students' reactions.

One alternative is to show student's confusion using a barchart, where it indicates the level of confusion via the height of the bar of the “confused” reaction, compared to the “clear” reaction. We think that using common colours that signify good (green) and bad (red) clearly, would be more intuitive and usable for our users. As our target users are instructors, they will likely be occupied with teaching most of the time. Using a barchart will not be as effective in catching their attention as compared to using full-screen and bright colours. Instructors can leave their phone on the table and rely on the screen colour changes to notify them of students' confusion. Furthermore, the usage of faces will be more intuitive for the instructor, as the faces reflect the student's emotions.

We also considered alternatives where the instructor will receive push notifications whenever students react that they are confused, but we think that it will likely be too disruptive to the instructor, and may spam the instructor if there are many students reacting at a time.

Workflow 2: Student expresses confusion

Students can express their confusion by first joining the session setup by their instructor. They can click on a link, scan a QR code or enter a session code to join a session (after

entering their display names), without needing to sign in. In the student page, they can tap on the “I’m confused!” button to indicate confusion, or the “I’m OK!” button to indicate that they are clear with the content taught. The reactions selected will stay selected until the instructor resets the reactions, or until the student untaps the reaction. These two buttons are purposefully placed near the bottom of the page and occupy a big area so that users can easily reach and tap the buttons with their thumbs.

First of all, one alternative is to authenticate students before allowing them to join a session. This allows the instructor to identify students via email, and possibly restrict access to a session to those in the respective class only. We think that the main purpose of our application is to allow students to express confusion freely with anonymity, so identifying a student is not crucial in achieving our primary goal. Although authentication for students can be a good nice-to-have feature in future iterations, we excluded this feature due to limitations in time.

For reactions, an alternative that we considered was to let the students send consecutive, unlimited number of reactions that are temporary and time-limited, as we imagined that students will be eager to express their confusion and notify the instructor of their confusion. However, this requires the instructor to keep track of the reactions on the screen, which disappears after a short period of time. In a real classroom setting, it is more likely that instructors will not be able to keep looking at their app screen most of the time, and only look at it when they finish a part of the lesson. This means that instructors may miss most of the reactions of the students by the time they check their app. Hence, we thought that a more efficient solution was to persist the reactions of students until the instructor decides that they have addressed most of the confusion from the previous part of the lesson, and can reset the reactions to keep track of confusion from the next part of the lesson. With this, students can rest assured that their reactions will be reflected on the instructor’s screen until the instructor explicitly clears the reactions.

Furthermore, we also considered different actions that students can do to send a reaction, such as swiping upwards, double-tapping, or tapping on a “send” button before sending the reaction. After careful consideration, we decided that actions such as swiping and double-tapping may pose usability issues as they may not be intuitive to all users. A “send” button also makes it more tedious for a user to simply react. This decision is based on the choice of allowing users to unsend reactions easily, so users can simply undo their selection if they have tapped on one of the buttons by accident.

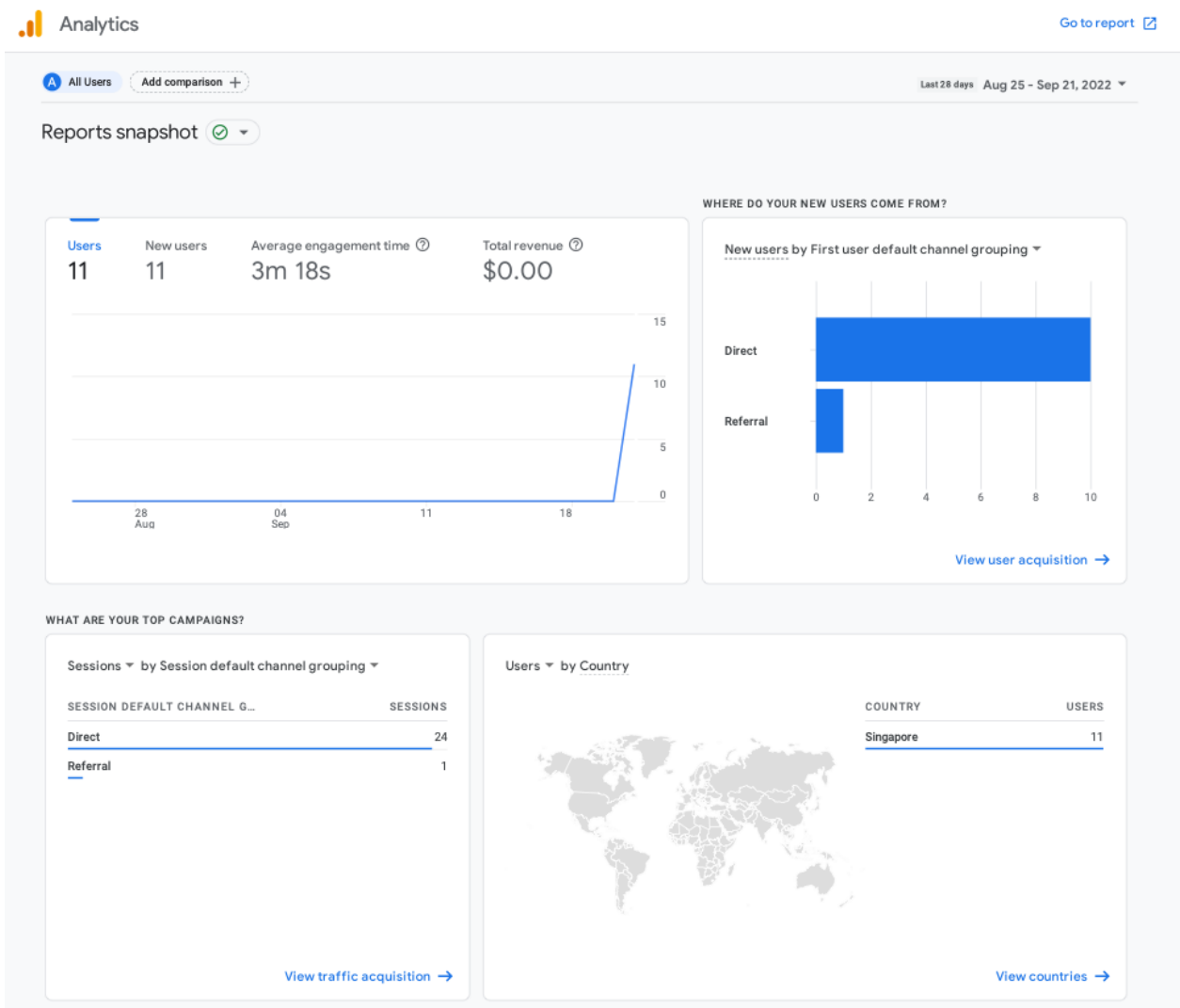
Workflow 3: Student asks questions

Similar to the expression of confusion, students will first need to join a session. In the session, students can type their questions in the input field at the bottom of the screen. They can then tap on the send icon to send their questions, which will be reflected in the question list in the top half of the screen.

An alternative that we thought of was to allow users to tag their reactions with a particular preset keyword, which may help instructors identify the section of the lesson that students are confused about. However, this means that students are restricted to expressing confusion about a general keyword, and cannot ask specific questions about a topic. So, instructors may not know why students are confused. Moreover, we think it is reasonable to assume that instructors using our app will stop every now and then to check on the students' confusion level and reset it, assuming that their main goal is to make sure that students are following the lesson. So, any confusion that is expressed would be related to the content that has just been delivered. For this, a reaction button without any tags will be sufficient to convey students' confusion about a general topic. If students have more specific questions, they can send them to the instructor so that their doubts can be directly clarified.

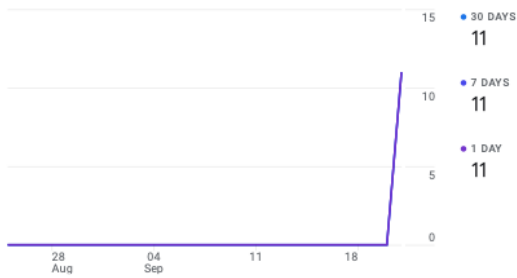
Milestone 14

Embed Google Analytics or equivalent alternatives in your application and give us a screenshot of the report. Make sure you embed the tracker at least 48 hours before the submission deadline as updates for Google Analytics are reported once per day.



HOW ARE ACTIVE USERS TRENDING?

User activity over time



HOW WELL DO YOU RETAIN YOUR USERS?

User activity by cohort

Based on device data only

	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5
All Users	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Aug 7 - Aug 13						
Aug 14 - Aug 20						
Aug 21 - Aug 27						
Aug 28 - Sep 3						
Sep 4 - Sep 10						
Sep 11 - Sep 17						

6 weeks ending Sep 17

[View retention →](#)

WHICH PAGES AND SCREENS GET THE MOST VIEWS?

Views by Page title and screen class

PAGE TITLE AND SCREEN CLASS	VIEWS
Confused	315

[View pages and screens →](#)

WHAT ARE YOUR TOP EVENTS?

Event count by Event name

EVENT NAME	EVENT COUNT
page_view	315
scroll	89
user_engagement	64
Logged in With Refreshed Token	46
session_start	25
Clicked Button	19
first_visit	11

[View events →](#)

WHAT ARE YOUR TOP CONVERSIONS?

Conversions by Event name

EVENT NAME	CONVERSIONS
No data available	

[View conversions →](#)

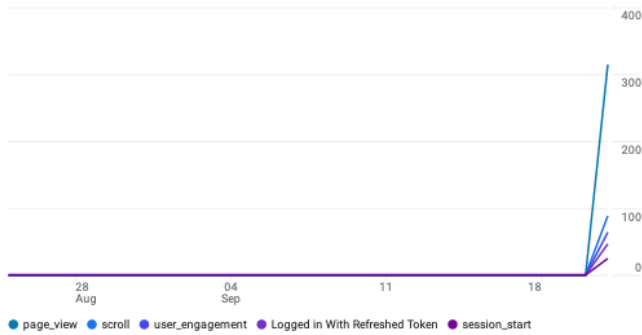
All Users [Add comparison](#)

Last 28 days Aug 25 - Sep 21, 2022

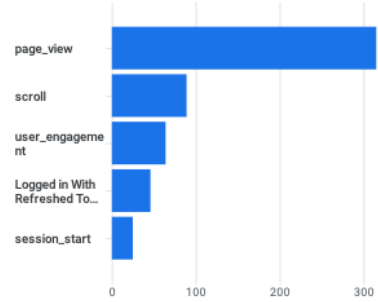
Events: Event name

[Add filter](#)

Event count by Event name over time



Event count by Event name



Q Search...

Rows per page: 10

Go to: 1

< 1-10 of 12 >

Event name		+	↓ Event count	Total users	Event count per user	Total revenue
			584	11	53.09	\$0.00
			100% of total	100% of total	Avg 0%	
1	page_view		315	11	28.64	\$0.00
2	scroll		89	11	8.09	\$0.00
3	user_engagement		64	9	7.11	\$0.00
4	Logged in With Refreshed Token		46	5	9.20	\$0.00
5	session_start		25	11	2.27	\$0.00
6	Clicked Button		19	8	2.38	\$0.00
7	first_visit		11	11	1.00	\$0.00
8	Logged in		6	5	1.20	\$0.00
9	Updated Session		5	3	1.67	\$0.00
10	Logged Out		2	2	1.00	\$0.00

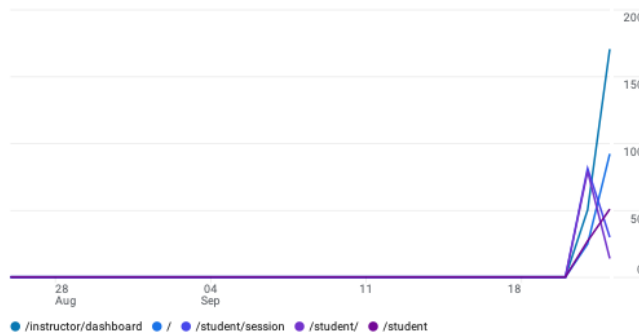
All Users [Add comparison](#)

Last 28 days Aug 26 - Sep 22, 2022

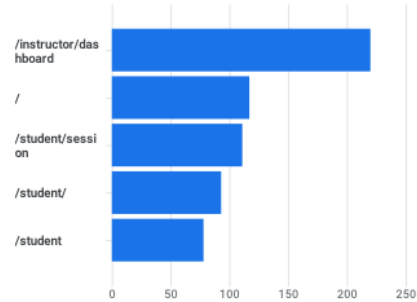
Pages and screens: Page path and screen class

[Add filter](#)

Views by Page path and screen class over time



Views by Page path and screen class



Search...

Rows per page: 10 Go to: 1 < 1-10 of 37 >

Page path and screen class	Views	Users	New users	Views per user	Average engagement time	Unique user scrolls
	822 100% of total	32 100% of total	32 100% of total	25.69 Avg 0%	2m 43s Avg 0%	32 100% of total
1 /instructor/dashboard	220	18	3	12.22	1m 12s	7
2 /	117	30	25	3.90	0m 14s	30
3 /student/session	111	7	0	15.86	2m 21s	3
4 /student/	93	3	0	31.00	2m 54s	3
5 /student	78	16	2	4.88	0m 50s	5
6 /instructor/session/3	61	1	0	61.00	10m 05s	1
7 /login	46	19	0	2.42	0m 06s	3
8 /student/scanner	12	8	0	1.50	0m 10s	0
9 /instructor/session/10	10	1	0	10.00	0m 46s	0
10 /instructor/session/create	9	5	0	1.80	0m 15s	0

With Google Analytics 4 embedded, we can understand our user behaviours such as page views and penetration rate of a particular feature better, so that we can better improve **current features** and have a more quantitative validation on **new features**.

For **current features**, we would like to see the penetration rate of instructor sign up / log in page view to the instructor dashboard, and compare that with the penetration rate from instructor dashboard to create session pages. This helps us to see what UI arrangements could be improved and what features are not used as frequently as expected. Moreover, comparing the number of clicks on the "Share to Student" button in an ongoing session with

the number of clicks on "Copy link" button in the share page tell us how frequent sessions are shared through urls. We could add more native social sharings if that is frequently used.

For **new features**, we want to use Google Analytics to help validate our ideas and support our User Interviews. For example, we want to use Google Analytics to carry our **Fake Door** Experiments on the following features (that we did not include in our MVP):

1. Students voting for a question asked
2. Instructors being able to **customise** reactions they want to receive from the class

Through "Fake Door" experiments and Google Analytics, we can track the number of clicks on our "fake door button" (Event on Google Analytics) to gauge the "interest" of these news features. With a high number of clicks, we can be more confident in validating our ideation. GA thus allows us to iterate with a quantitative perspective in addition to the qualitative User Interviews.

Milestone 15

Achieve a score of at least 8/9 for the Progressive Web App category on mobile (automated checks only) and include the Lighthouse HTML report in your repository.

The lighthouse HTML is in the submission folder:

<https://github.com/cs3216/2022-a3-2022-a3-group-6/tree/main/submission>

Phase 4: Coolness Factor

Milestone 16 & 17

Being an education-focused app that is most suited for a classroom-context, we felt that features like Social Integration or Geolocation were not features that would help deliver or enhance the value of our app. Instead, we implemented “cool” features that are beyond the basic requirements as outlined by the milestones, which are described below.

1 Real Time Capabilities

First and foremost, we implemented WebSockets, which forms the backbone of our **real-time** interactions with the Student and the Instructor. This avoids having to do multiple, frequent HTTP requests. The technical details (and reasons why we chose it) are as outlined in Milestone 4, but it is cool simply because it adds another layer of depth and versatility to our app. For instance, in the future, we can potentially add a functionality to allow teachers to send messages to the student, which could be useful if the teacher wants to let the student know that their question will be answered later (e.g. a reply of “I’ll address this during the break” or something similar). Going even further, we could even have chat channels and threads like Slack/Discord!

2 Enjoyability

(Note

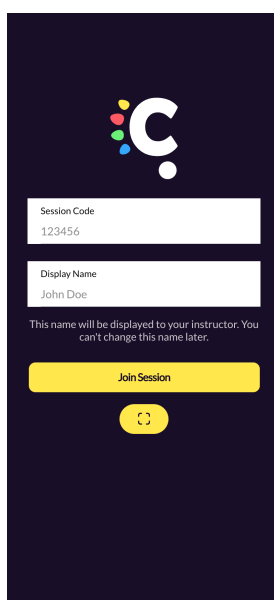
Secondly, just as Kate shared in the Week 2 UI/UX workshop, an important yet oft-neglected aspect of UX is enjoyability. Having a useful and usable app is, of course, a priority, but enjoyability is a cardinal aspect that keeps users engaged and inclined to use the app everyday. To this front, we deliberately implemented theming, custom animations and custom assets that makes the app more “fun to use”, some of which you’d have already seen above in the UI design section.

For instance, upon loading the app, users are greeted by a cute animation featuring a magnifying glass and a question mark; this is perfectly aligned with what our app helps its users achieve – asking questions and having them addressed.

Subsequently, the start page materialises, housing a dark indigo background and yellow buttons. What’s more, the “Confused” logo appears to be levitating above the words and

wiggling about! Compared to the utilitarian white-and-blue theme of Zoom's login/join session pages, "Confused" bears more resemblance to the main menu of a game, inviting its users to click in and experience a magical journey.

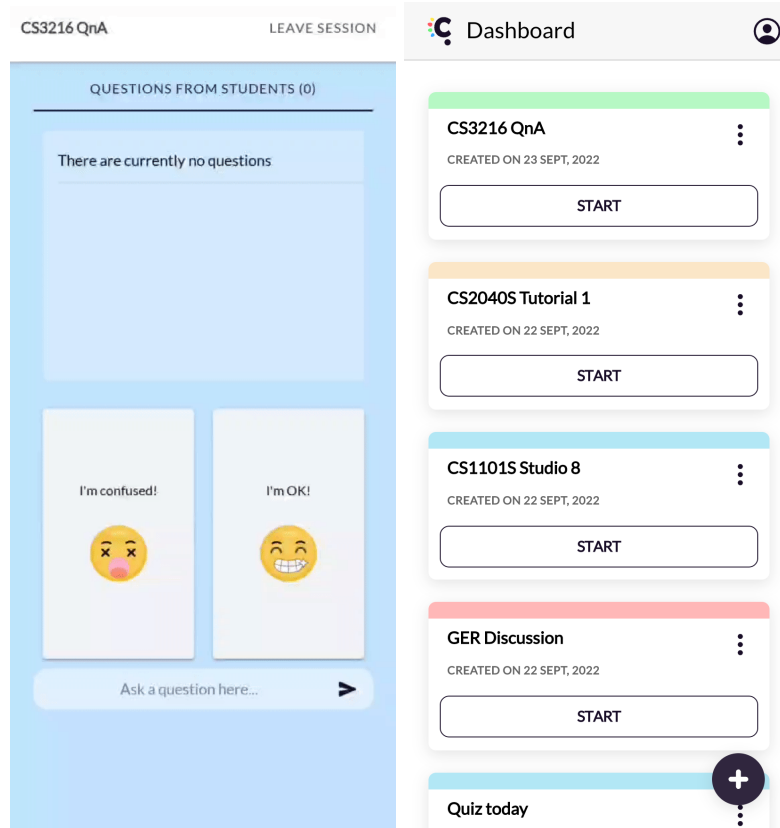
The instructor login/signup page reverts to a more formal white-and-black theme, as its goal is fore and foremost to be functional, clear and uncluttered; to get instructors through the purely-administrative task of login/signup to unlock the full potential of the app. Instructors will rarely view this page, instead spending a majority of their time in the actual session within the app. On the other hand, the student "join session" page sticks to the game-ish and fun theme of the start page, as students will be seeing this page every time they join a session.

A screenshot of a mobile app interface for joining a session. The background is dark purple. At the top center is a logo consisting of a white 'C' with four colored dots (yellow, green, blue, red) to its left. Below the logo are two white input fields. The first is labeled 'Session Code' and contains the text '123456'. The second is labeled 'Display Name' and contains the text 'John Doe'. Below the second field is a small white text note: 'This name will be displayed to your instructor. You can't change this name later.' Below the text is a yellow button with the text 'Join Session'. At the bottom center is a yellow circular button with a white double-headed arrow icon.

Student Join Session Page

The fun doesn't just end there. When students are in a session and click on any of the reaction buttons, it expands and shrinks, to provide a clear visual indicator that the expression has been selected and toggled on.

As for instructors, after login they are presented with a multicoloured list of sessions (assuming they have created some beforehand), once again a stark contrast to the boring black-on-white theme of Zoom's "Sessions" page.



Student clicking reaction and Instructor viewing session

During an actual session, the background turns yellow, and then red, as more and more students are confused. This is accompanied by cute expressions which change alongside the background colour. Combined, these elements act as an endearing yet visually-noticeable and impactful indicator to the instructor that, *“Psst! Some of your students are confused!”*. We specifically designed in this way to ensure that instructors wouldn’t miss out on any confused students the same way that they might miss out on Zoom “raised hands” or “yes/no” reactions. After all, a background change is far more noticeable than a small pop up or small icons beside students’ names.

In essence, the whole app was designed with enjoyability in mind, rather than disregarding it completely or shoehorning it in as an afterthought, and this presents itself from the colour palette down to the cute expressions used.

3. QR Code

As we were building this mobile cloud application, we were thinking of cool mobile-specific features that were appropriate and functional. QR codes seemed to fit this bill pretty well.

At this stage, Singaporeans are no strangers to QR codes, having many-a-time scanned a TraceTogether QR code to gain entry to a mall or shop. In the school context, QR codes have been used to mark attendance as well as quickly and conveniently disseminate a URL to a large group of people. In fact, our mobile-first application fits the second use case almost to a T. By making use of QR codes, we can ensure a painless and efficient way for students to join the session – no more having to enter complex meeting IDs or having to search for that meeting link *you just know is in your inbox somewhere...*

On the technical front, a QR code scanner makes use of the camera-related APIs built into modern browsers to capture the QR code, analysing the video feed using QR detection libraries or even native QR detection APIs to hunt for information embedded within. The related APIs include `navigator.mediaDevices.getUserMedia()` to grab the camera feed and analyse the image with the `zxing-js` library. Some browsers such as Chrome even support the [Shape Detection API](#) (QR code scanning portion was [launched in 2020](#)) which natively detects QR codes and barcodes without needing external libraries. These APIs enable web apps such as Confused to behave more like native apps, and to reduce the need for a dedicated QR scanner app.