Jason Jiang, Asfandyar Khan, Ivana Sanchez Diaz
Group Project, Final Report
CMSC436

# ClouText

## Evolution of ideas

The inspiration for this app came from group member Jason. When we were meeting to discuss what idea we should come up for the project we were very close in choosing something else until he mentioned the Dark Souls messaging system. In the video game, Dark Souls, a player is able to go anywhere within the map and leave personalised messages that other players can read when they are around the location the message was created. If the player is close enough to a messaging bubble, they are able to tap it and read it. This inspired us to try to make an app based on location and the user would be able to use their camera, walk around, and discover 3D messages around them.

In the first few weeks we only researched on the original idea we came up with. The idea was for the app to have a single view that worked with multiple button components. The user would have a map as their home screen so they are able to see where a nearby message can be. They are able to click on a cloud object and it would take them to the camera view where the message will appear in front of them. The user does not have to be in the same location as the message to see the message, they just have to be close enough so the message appears in the map and then the message will appear in front of them in 3D form. The location is relevant to where the message was created.

The goal of the messages is so users can see quick and simple texts that relate to their surroundings of the location they created the message. It is not a forum or meant to take in long paragraphs. The idea was to leave “comments” on a 3D space.

The app can be used by anyone, but the focus group was for users who are related or affiliated to the University of Maryland, College Park campus. Our goal was for students to use this app to leave messages relating to the university so other students can read it. In our first few meetings, we already found much of the knowledge we needed in class lectures and online resources that would help us code our ideal program. At this point, our ideas and the information we found seemed to match up so we assumed that it would be going very smoothly, but that is never the case when it comes to trying to build a well-functioning app.

During the starting implementation of the app, we had tweaked some ideas, mostly in the design section. We decided that the buttons would go at the bottom and there would be other pages besides the map such as a “Global Chat” and “Profile” section. Starting early in the semester to implement simple features was key for us because we changed a lot of components from our original idea to make the application more user friendly. We did not want to do the bare minimum for the application, knowing that the appeal of the app to the user is also

very important. The team decided to add a Global Chat because it would make the users, and honestly our lives as developers with recently learned Swift knowledge, much easier.

The idea and evolution of the Global Chat section is just an easier way to see all of the messages that all users have created in the app. This means the user does not have to go and click on every cloud to know what is being talked about in that particular location. They can easily scroll in the Global Chat section and not miss out on what is being talked about in. If the user wants to see the 3D text in front of them, they can leave the Global Chat to go to the map and then click on any cloud icon within the map so that particular 3D text appears right in front of them. We kept this in mind so messages have an even easier way to be accessed by the user.

The next evolution of our idea was to create a Profile page. Personalization is important in every app, so we decided that the application would have a better look and feel if each user has their own profile. This meant that they would only be able to create, edit and delete their own messages. Their app “name” would be their own email address that they use to sign up into the app. The idea of having profiles tied to messages means that they give messages a source of credibility in our end and users who want to read each other's messages. This can also add a sense of comfort to users within the app, knowing that a person is actually tied with the email that is being shown.

Shifting gears from the Profile Page, we were thinking about how we wanted our AR Object to look like. There were several iterations of ideas we went through for this. At first we were thinking of an actual cloud shape. It would be easy since we could arrange different sizes of circles together to make it look like a cloud. At the bottom of the cloud a white, flat rectangle would be attached to the cloud. This would make it seem like the cloud is holding a piece of paper, and the paper would have the user’s message. This was possible, but the issue with this was that it made the camera page view really slow because it was trying to generate all those objects with the text, compared to just having the text. It was probably the data that the whole app held or the fact that the iPhones we were testing in were not the newest models, but it did not give a good feel when it took the object to appear after 3 minutes of waiting. In addition, the rendering we were currently doing at that time for the object was not the final one, so adding more detail or touch ups was going to make it even slower. The final evolution for this idea was to just generate the message in 3D in front of the user whenever the clicked on a specific cloud icon from the map.

# Process we went through

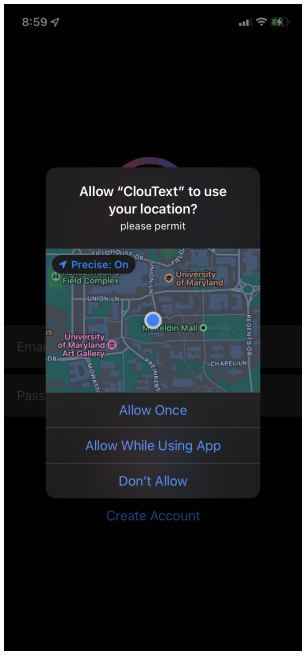
After going through several phases of constructing a more solid and realistic app idea, we went into the more critical components of the app that needed to be worked on right away. This team knew that we were all beginners in Swift and we had to take this into account.We had to think about that since certain features we worked on would end up becoming too slow or messy and had to pivot on the original idea to a similar more manageable one. As explained earlier in the report, we wanted the user to have an AR Camera view when they clicked a cloud from the

map view. The easiest way to get the AR Camera view to be integrated into our other views that have been created was to code with a Storyboard file. Of course, there are other ways to do this but the most effective way we found through research was to use a Storyboard file. For certain issues that will be talked about later on in the report, we were not able to use Storyboard. To go back to our button features, it was easier to use TabView and NavigationView in SwiftUI than Storyboard components in order for the user to go through the different pages we created (Map ( AR view page inside map page), Global Chat and Profile). The final decision was to keep the tab views at the bottom of the application and use NavigationView so the user can go to the camera view by touching a cloud icon from the map page through SwiftUI.

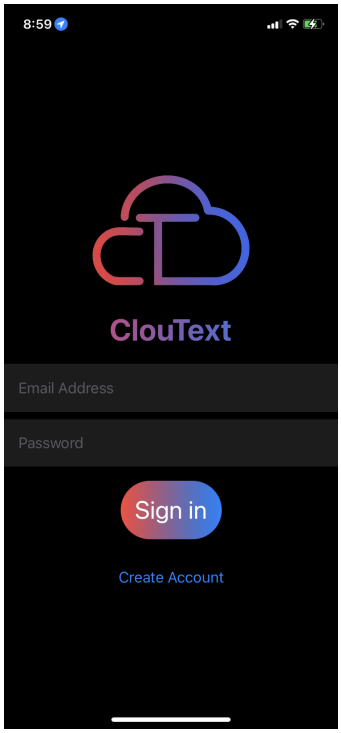
One of the most complicated processes while trying to make the app was trying to merge the code from the AR view with the firebase data so that it would appear in an AR Object in the camera view. Most of the base code and tips found on the internet were using Storyboard to achieve data functionality with AR but we had many different libraries we needed for other components. Trying to connect the Storyboard feature with the View we had already created was a challenge. The only solution that connected an AR Object to SwiftUI properly was using a UIViewRepresentable struct. It worked great and it was efficient but in terms of how more functional and faster we could have coded our app, Storyboard would have been better. This connection we were not able to make with our pre-existing files created a bump for us in the AR side, since almost all internet sources on AR components make the use of Storyboards. The UIViewRepresentable struct was not bad to use, but seemed outdated compared to how much more and how faster one can work with Storyboard. It was difficult at first, but it did not stop us from trying to find as much information as possible for UIViewRepresentable in order for us to meet our AR goal. Regardless of that issue, we were successful and managed to display AR Text from the messages saved through firebase.

Some of the aspects we added in a smaller scale to make the app feel more user-friendly, was to add a sign in and out feature as well as having created a logo to fit the color palette of the app. Small adjustments such as these go a long way into how polished the product looks with added functionality. A user could have another account and want to switch accounts, someone else may want to use their account through someone else's phone and have to sign out, etc. so there is an option for that. We also took into account how the features look. We managed to use outside images, SF symbols and picked a color theme red and blue. We also created our own CloudText logo that appears when the user wants to sign in as well as the thumbnail of the application.

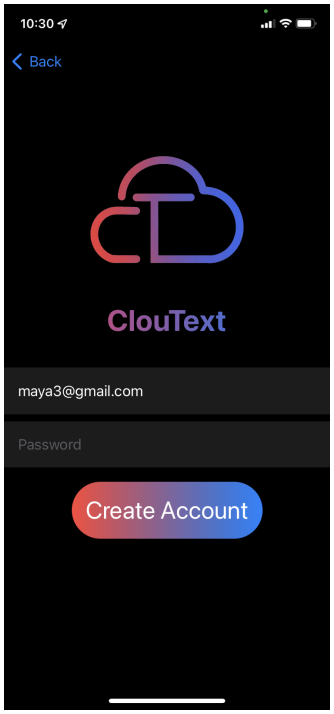
# Final App Walkthrough



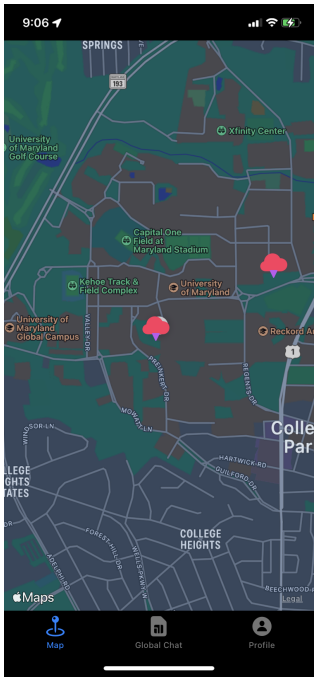

-- When you open ClouText for the first time, you will be asked to allow location tracking. For ClouText to work you can either choose “Allow Once” or “Allow While Using App”.



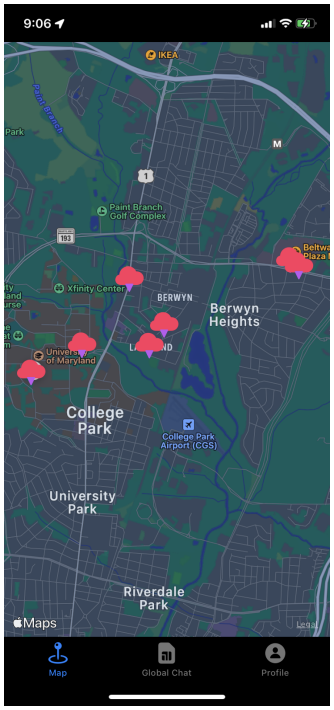

-- You will be greeted by the Sign In page. If you already made an account you can sign in, but if you don't have one you can click Create Account below.



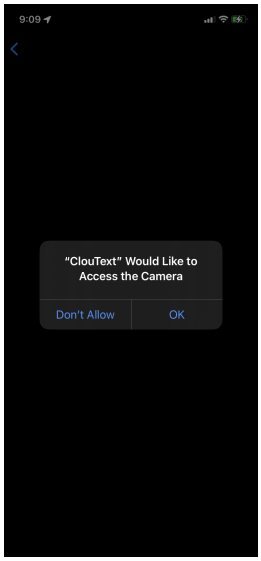

-- You can put any email address as your username, and any password. NOTE: The password must be at least 6 characters long for ClouText to register you in as a user. Please make it 6 characters or longer.




-- After you put your email and your password and click “Create Account” you will be taken to the homepage of the app, the map page. In this page you are able to see the messages that have been created around your location.

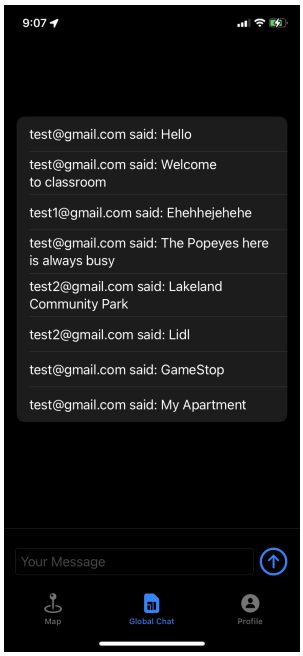



-- You can also zoom out more and notice if there are more message clouds around your area. When you click any of the clouds there, you will be taken to the camera view which will show you the text message that was created in the location that you pick and will be shown to you in 3D.

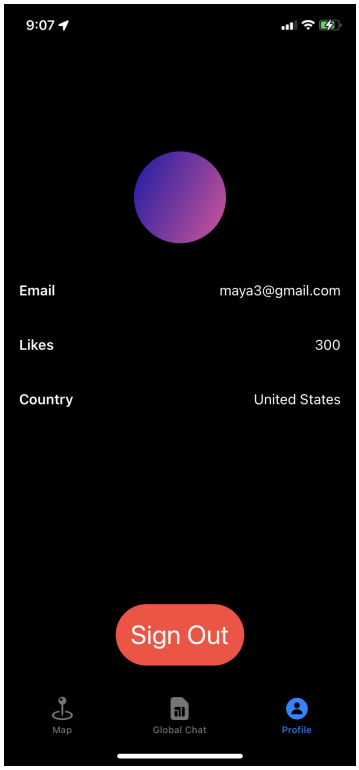



-- When you click on a cloud for the first time you might get this permission, you can click “OK” to allow the app to access your camera to see the 3D text.

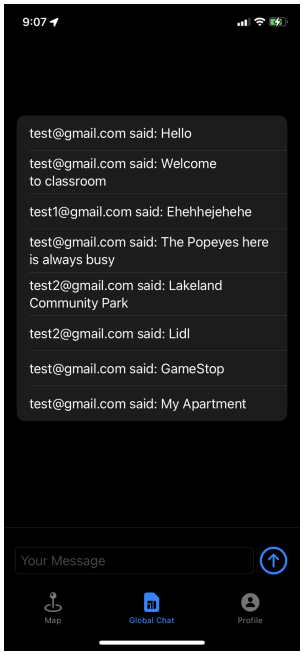

-- Above you can see one of the clouds that I clicked right in front of “University of Maryland” in the map says hello!



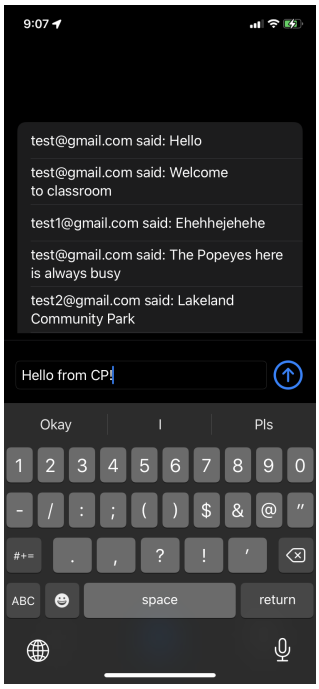

-- The map was part of your homepage, but if you look at the tabs on the bottom you can see that there are other pages besides the map view. The middle tab says “Global Chat” and this section has all the messages that have been created and the users emails that have created them.



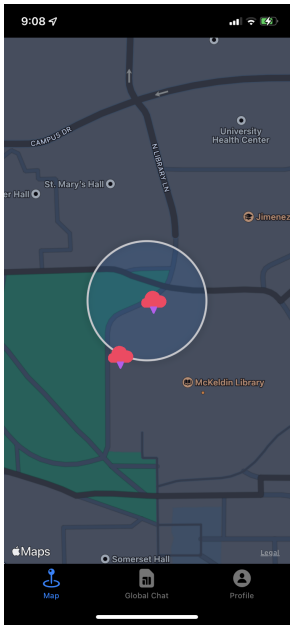

-- The last tab that says “Profile” has saved information about the user such as their email, number of likes and the country that they are in. They are also able to sign out with the “Sign Out” button below.



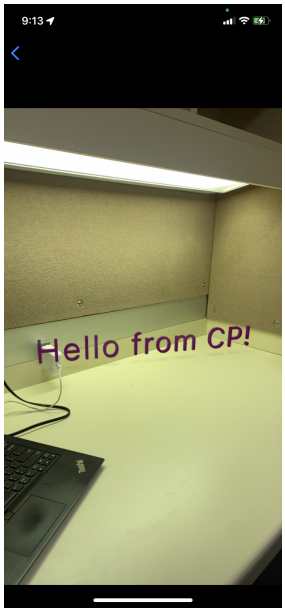

-- Let's go back to the Global Chat view for a quick explanation on how to create your own 3D message. At the bottom you can tap on the “Your Message” bar to start typing your message.




-- I will type “Hello from CP!”, but you can type anything you want. After we are done typing you can hit the blue arrow in the circle at the end of the text bar to send your message. After that you can click return in the keyboard to hide your keyboard again.




-- You can go back to the Map view. The blue marker is where you are currently, so you can see the message you just created by clicking on the nearest cloud. Click on it.




-- Here you can see my text appears in 3D, and yours did too!