# SNAP & GO EXPO SCRIPT

## INTRODUCTION (RYAN SPEAKING):

Hello, when we came up with the idea to create the snap and go crafting app,
we wanted it to be the all in one companion for any type of crafting needs So, some of the problems we wanted to solve were, customers having to search for prices,
the availability of items, and coming up with ideas for crafting projects…
That can be really time consuming for the customer. This can hurt the bottom line of Online business owners, and really take away time from working on projects by hobbyists and people Who use crafting projects to make a living. So the solution we came up with was to make an App that uses images to get prices and project ideas related to crafting items To save people time. This app will really save hobbyists and craft shop business owner's time and money by quickly finding them relevant results that they may need.
Our target audience is really people that do arts and crafts as a hobby and also those that do it as a business. People of all ages fall into this category.
According to statista, arts and crafts is a growing market, with its worth predicted to reach $52 billion by the year 2024
According to ironSource, on average most apps make $.02 per video add. If at least 20,000 Video adds are shown in a day, then that would amount to $400 per day
Revenue for us developers. That would be equal to $146,00 a year from our app,
If it were to contain real ads and be successful, the way we want it to be.
As far as project architecture and external resources go, our group used REACT Native and Expo frameworks to really make a cross platform app.
That would be usable on iOS and Android, and these architectures allow us to seamlessly blend and create for both app simultaneously.
We used google vision to identify objects and images and return that identifying string back to the app as sort of an AI way to identify products.
That customers may be wanting to find out more information about. We have a web crawler Built with crawler node package that will grab all relevant results and pass them to the front end.
For our database we used mongoDB, with no SQL database, so we don't have to use any type of SQL commands, to store queries and user information.
We integrate a GIF as a fake advertisement to allow the web crawler time to return results and demonstrate how financial revenue can be generated.
Now the combination of these parts working together deliver results to hobbyists and business owners quickly, simplifying their lives, and allowing them to complete their tasks much faster than if they were to search all on their own, through the internet, or worse, driving around and finding prices at stores themselves.
As far as implementation goes, you can look at the following graph here:

## FRONT END (JOSEPH SPEAKING):

Hi, my name is Joseph Shafer and I was primarily a front end developer for the Snap and Go app.

Some of the things I worked on with the navigation scheme, the login interface, the Google Login implementation and the results interface.

For navigation what we used was the React Navigation Library which gives us access to a stack navigator, a bottom Tab navigator, a top tab navigator, and we're able to nest navigators within other navigators.

For stack Navigator: This is an example of a user clicking forgot password, and then clicking back to login.

For top Navigator, a user is on the results screen and they can click between products and projects.

For bottom Navigator, a user can click on the user interface or the camera interface.

This shows also generally what our login looks like and what the interface for the camera looks like.

This is a map of the general paths a user could go down for navigation.

So, from the top level from Navigator into bottom Tab Navigator by default it'll land them into the user screen within the bottom Tab Navigator.

It'll check if they're logged in or not.

If they're not logged in, it'll take them to the login screen where they are able to sign in through Google, create an account, start a process for putting in their email.
If they forgot their password that they can then take them to a reset password screen.

If they are logged in, it'll take them into a different stack Navigator for a logged in user where they can land on a home screen.

Then go to past searches they've done, which if they click on one of their past searches it'll take them into a top tab Navigator for results that display products and projects.

If from the top, going from navigator to bottom tab, they click camera, it'll take them to a camera stack where if they take a picture it'll request to Google Vision a string identifying the image from vision.

It'll return that to us.

We call a web crawler on the identifying string and that returns results to the user and display things related to the picture they've taken.

It will take them into a top tab Navigator for displaying those results on products and projects.

These are some login interfaces we worked on where I designed the snap and go front page and then this is also the implementation of the Google login.

Where a user can log in through two different ways.

This is the results interface where basically, after we get the list of strings from a web crawler, I create a title out of that string and then make a button that you can click on that takes them to that link within the app.

Yeah, that is basically what I worked on for the last year.

# CAMERA FUNCTIONALITY (JUAN T SPEAKING):

Hello, I'm Juan Trigueros and the main contributions that I made to the project are the camera interface,
a transition from the camera interface to a simulated ad before loading results, and overall general user
interface work. I'll start by talking about the camera interface and its importance in our application. In
order to make our app convenient for users to search for items or ideas by taking a picture, we needed a camera
interface that would take the best pictures possible. After testing a few camera APIs, which are application
programming interfaces, we found one that worked well with both iOS and Android devices, after getting their supported screen and camera aspect ratios. Taking good pictures is important because we then compress the image
to decrease it in size and then encode it so that the image recognition portion of the app can take its guess.
Secondly, transitioning from the camera interface to a simulated ad before loading results was important in ensuring
that the web crawler had enough time to grab results and return them before the results screen loaded. I was
responsible for getting a gif to simulate an ad that lasts for a few seconds, allowing the app enough time to
grab results. Lastly, I contributed with general user interface work, such as helping get screen navigation working
or creating a screen that grabs a user's search history from our database, but I focused more on testing the

implementations on Android to make sure they worked the same as on iOS devices. To get our app to work on both
operating systems, we used the React Native framework alongside the Expo framework, both of which are platforms
that intend to make creating multiplatform apps easier. These played an important role since we wanted our application to be usable by almost anyone. While testing, there were issues that appeared on Android but not iOS.
I found that the main issues that occurred were due to the fact that there are so many different Android devices.
Therefore, I implemented conditions to check whether the user device is running iOS or Android in order to grab
important information related to their device. Hopefully this information provides you with a better understanding
of how the contributions I made tie into the Snap & Go project as a whole.

# WEB CRAWLER (JUAN V SPEAKING):

Hi there everyone. My name is Juan Villasenor and the main components of project Snap & Go that I focused on were the web crawler it uses and the connection between our app and Google's image recognition framework known as Google Cloud Vision AI. I will first be discussing about the involvement of Google's image recognition framework and its importance in our application. My team and I thought it would be pretty neat to allow our users to conduct all of their searches by simply taking a picture and we needed a framework that was able to accurately predict the objects the user is taking a picture of. After some testing with multiple frameworks, we saw that Google's image recognition software would just work best with our application. Having accurate predictions is important because we then use the AI's strongest guess as a target search term to traverse through various sites and bring back all related results. This then brings me to the next topic which is the web crawler. A web crawler can be defined as software that rips the contents of various web pages from a website. I was in charge of building and maintaining the web crawler we used for project Snap & Go. This plays a very important role in our application since this is how we are able to show our users where to get arts and crafts materials that they need as well as show them some arts and crafts projects that they can do with the materials they have. Users snap a picture of any material, we then use Google's image recognition software to predict the object, and we finally pass the strongest prediction to the web crawler to bring back all related results. After some testing with the web crawler searches, I found that conducting these searches for every request would definitely cause heavy traffic buildup. To counter this, I also implemented a caching system. A cache could be thought of as a shortcut where we conduct one search and save those results in a file. Then when another user wants to search for the same search term, we resend that stored data and they would get their results back instantly. After some time has passed, the software will automatically update the cache and all of the results. Hopefully you guys now have more of an understanding of how these components I worked on tie into the Snap & Go project as a whole.

# DATABASE (JACQUELINE SPEAKING):

Hi. I'm Jacqueline Peralta and I was primarily responsible for developing the sign-up process, and setting up
user authentication and the database. MongoDB Atlas is the cloud database service we are using for our project.
Two reasons why we chose MongoDB was because it enables developers to build easily and the document query language
is simple compared to SQL databases, therefore it was easier for me to design collections as a first-time user of
MongoDB. When a user signs up for the first time, the required information is sent to the backend and stored into
the database. If someone with the same username and/or email has already registered, they will be notified as those
values are unique in our database. For user authentication, I used express session. It's a middleware to validate users.
What this does is that a session is going to be stored on the server side of our application and we are authenticating
into the session with a secret key. A session is used to store information about a particular user moving around the
application. In addition to setting up user authentication, I thought it was important that users had the option to
reset their passwords in the event that they were to forget it. The way I approached this was by implementing Nodemailer,
a module for Node.js designed for sending emails. One of its main features include attaching HTML content in those emails.
When the user hits the forgot password route on the backend with the email address they have entered, the first step is
to check if that email exists in our database. A token is then generated that can be attached to the user's account and
setting a time limit for that token to be valid. I created a gmail account for Snap & Go and from this email we will be
sending the password reset email link to our users. The link in the email will redirect users back to a screen on the app
where they could reset their password. Redirecting users to the reset password screen was a challenge because the token
had to be passed to the app. If successful, their passwords in the database will be updated and they can log in with their new password.
These links expire after a day, so if a user takes no action and does not attempt to reset their password,
they're password will not change.

# SECURITY (RYAN SPEAKING):

Hello, my name is Ryan Wallace, and I was responsible for setting up the security of the application. In this section I will be going through the process I followed for each of the

modules of our application, as most of this work is behind the scenes it was a goal for me that the users of the application never run into any issues and the process of using the app is intuitive.

For our database we used MONGODB, I followed the 7-step process they laid out for how to secure the database. Those steps have to do with encrypting the data that is transferred into the database. We also used some tools known as bcrypt, which is used to hash information such as passwords to allow secure storage of user information. The IP addresses that are allowed to access the database right now are also set into the code, to prevent any remote access into the database. MONGODB also encrypts all data while it is in transit on its own to ensure the integrity of information to and from the database.

When it comes to the front end of the application, the portion of the app that the user will interact with, I took advantage of an API known as VALIDATOR.JS. this tool has hundreds of ways to check on any strings that are typed by the user and can check that they are the EXACT format that we want them to be, before they ever even enter our system. This tool is used for all input that the user will attempt to type in and should prevent any type of common string manipulation attacks into the application.

When it comes to logging into the application, we went through months of work on using the AUTH-0 framework, but in the end had to ditch the idea and chose the much simpler google authentication framework instead. This should be more convenient for the users in the end, as google accounts are used by the vast majority of people today and will make the flow of signing into the app much easier as the user will not need to remember too much new information to use the application. This is also very secure, as google authentication is industry leading and offers various benefits such as 2 factor authentications.

The application is segmented off as much as possible from the end users so there should be little possibility that user's information ever be at risk from malicious parties. The app easily goes above minimum expected security standards so you can complete your projects with a clear mind! Thank you for your time.

# CONCLUSION (RYAN SPEAKING):

So in conclusion, what we learned and the future of our app. Some of the things we learned were things such as how to create complex interfaces for a multiplatform mobile application using external resources.
We learned how to implement image recognition to scan an image and then use a web crawler to grab results from the internet for that specific image.
We also created two options for users to have accounts to save their search history of their images.
As far as the future for this application goes, we want to possibly get the app to search for more items that go beyond arts and crafts, so that we can expand the app to other topics.

Were looking at adding a landscape option to the camera interface to make it easier for users to take images.
Were also trying to get the app to recognize multiple items within a single image, so that they can do many searches off of a single image that they take. Were hoping to get the app ready to display multiple results within one image, allowing users to filter through them and select which items they really care about looking up.
Alright, thank you for your time, and I hope you enjoyed the demonstration of our application.