Due: Monday, October 23, 2017, 8:00 AM (submit to BlackBoard, under Assignments)
File Type: Microsoft Word

Team Name: 10
Team Members and email addresses:

- Josiah Gray <j305g268@ku.edu>
- Shaina Krumme <s.e.krumme@gmail.com>
- Ethan Ward <ethanandrewward@gmail.com>
- Li Yehan <y494340206@ku.edu>

Team Meeting time: Friday 4:00-5:00 pm
Lab Meeting time: Wednesday 4:10 pm
Contact: Shaina Krumme <s.e.krumme@gmail.com>
Project Sponsor (if any): none

Project Description:

- Why is the project being undertaken? Describe an opportunity or problem that the project is to address. What will be the end result of the project?

  Augmented reality is on the forefront of technology. In augmented reality, you see both the world around you and computer-generated content! You can view AR through technology you already own (such as your cellphone or tablet), or get the complete experience by using a specialized headset.

  InterpretAR will make it easier for you to learn and retain foreign languages and to communicate when you don't know the language of the people around you. Since the app translates objects within view of your camera, all of the translations are relevant to what you are doing at that moment. You will save time and communicate faster since you no longer have to look up words in a book or over the internet!

Project Milestones ("Deadlines," not starting dates):
First Semester:

a) Get familiar with Apple ARKit and Swift. (October 13th)
b) Finalize design of app (October 20th)
c) Create a basic iOS app with interface (November 3rd)
d) Make the iOS app able to recognize different objects in real world (November 24th)
e) Basic incorporation of language features (December 8)
e) Rough draft of documentation (December 8)

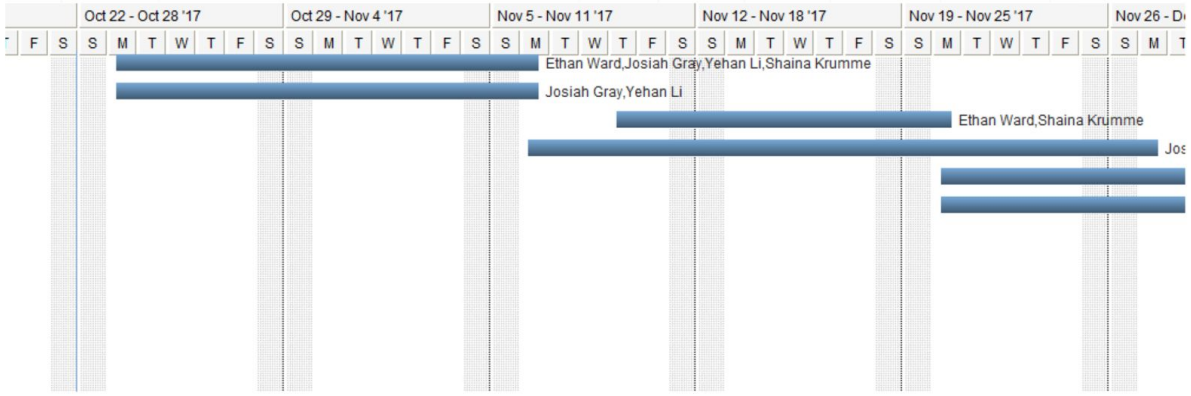
Second Semester:

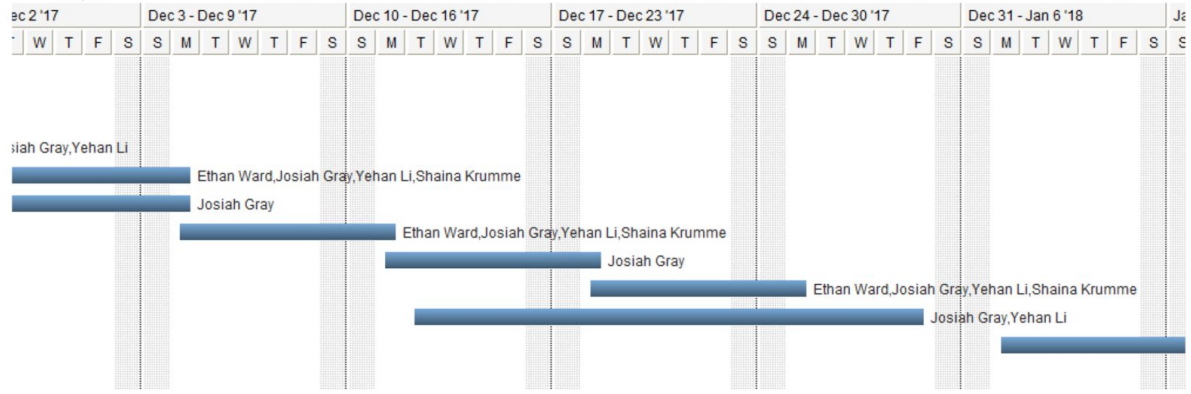
a) Ensure that the app can translate languages. (January)
b) Improve the performance of the app (February)
c) Consider incorporating additional features, depending on progress (February)

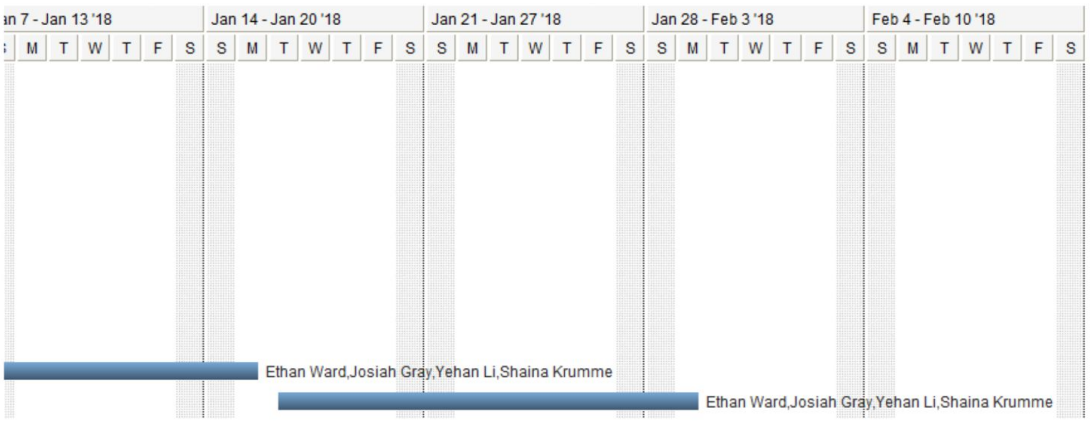
d) Finalize documentation (March)

e) Test app on real users and potentially release on App Store. (April)

	Name	Duration	Start	Finish	Predecessors	Resources
1	Project Proposal Video	11d	10/23/2017	11/06/2017		Ethan Ward,Josi
2	Get familiar with Apple ARKit and Swift	11d?	10/23/2017	11/06/2017		Josiah Gray,Yeh
3	Start planning ML model structure	8d?	11/09/2017	11/20/2017		Ethan Ward,Sha
4	Create a basic iOS app with interface	16d?	11/06/2017	11/27/2017		Josiah Gray,Yeh
5	Make the iOS app able to recognize different objects in real world	11d?	11/20/2017	12/04/2017		Ethan Ward,Josi
6	Basic incorporation of language features	11d?	11/20/2017	12/04/2017		Josiah Gray
7	Rough draft of documentation	6d?	12/04/2017	12/11/2017		Ethan Ward,Josi
8	Ensure that the app can translate languages.	6d?	12/11/2017	12/18/2017		Josiah Gray
9	Improve the performance of the app	6d?	12/18/2017	12/25/2017		Ethan Ward,Josi
10	Test app on real users and potentially release on App Store	14d?	12/12/2017	12/29/2017		Josiah Gray,Yeh
11	Consider incorporating additional features, depending on progres	11d?	01/01/2018	01/15/2018		Ethan Ward,Josi
12	Finalize documentation	10d?	01/16/2018	01/29/2018		Ethan Ward,Josi







Project Budget:

- **iPad ($429)**
- **Apple developer license ($99)**
- **MacbookPro ($1300)**
- **Translating fee for Google Translate API ($20 per 1 million characters, ~$40-50 for app testing)**

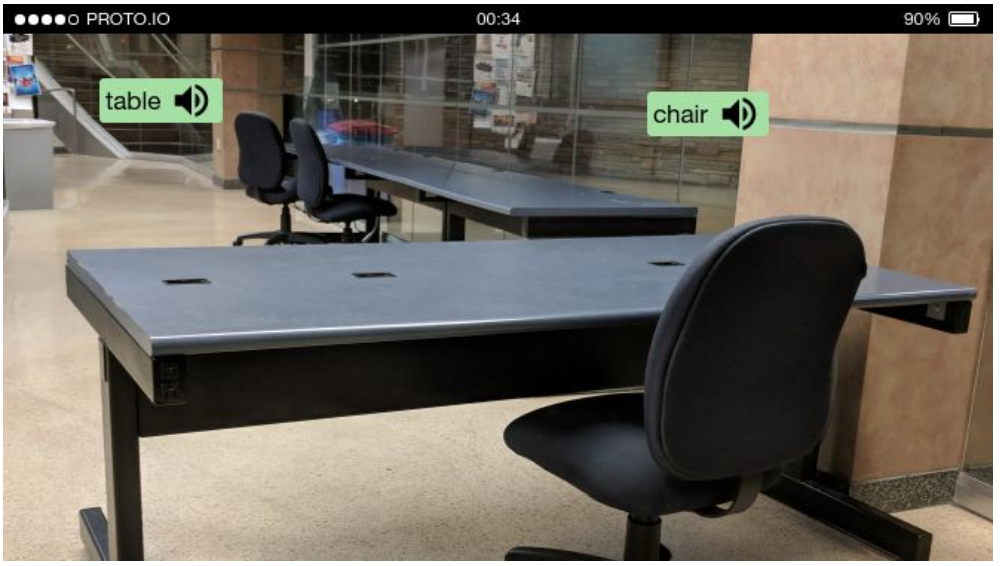
- Special training
    - Self-learning via ARKit tutorial (no cost)
    - Self-learning Swift for iOS development (no cost)
    - Github (for team members who aren't familiar)
- When they will be required?
    - As soon as possible

Work Plan:

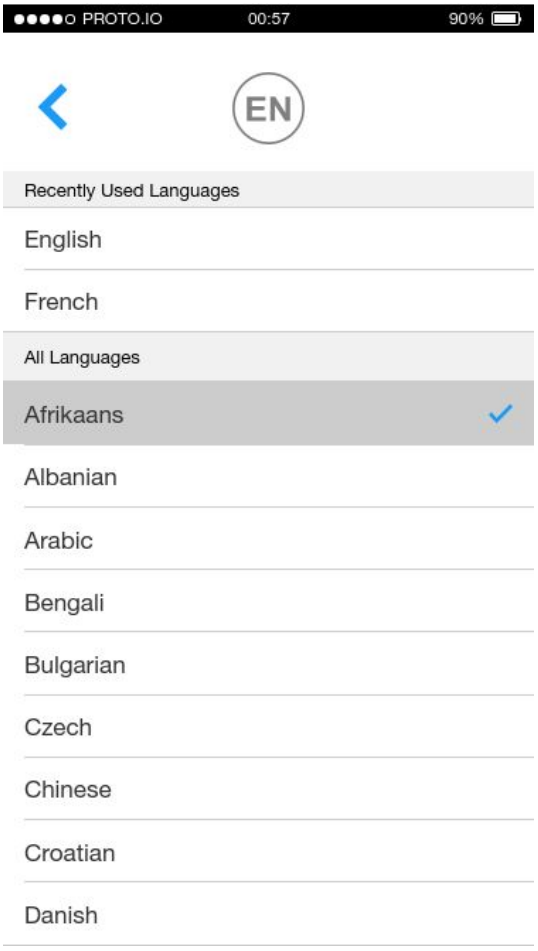
- Individual Focuses (may change)
    - Machine learning components: Shaina and Ethan
    - iOS front-end development: Josiah and Yehan
    - iOS back-end development: All
    - ARKit: All
    - Language Translation: Josiah and Yehan

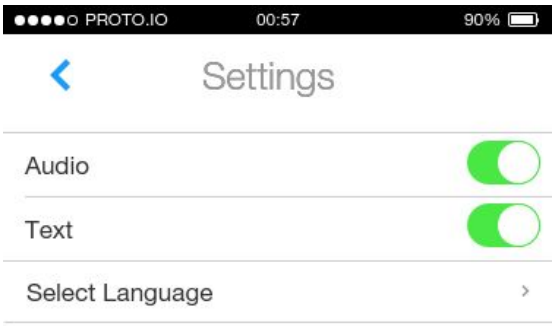
Overview

InterpretAR will be an iOS application that uses object recognition to translate objects within view of your camera. There are two ways that the app will translate objects, and the user can select one or both of those settings. By default, both settings will be selected. The first setting is to have a label placed above the object. The second setting is to have a microphone icon placed above the object. When clicked, the microphone icon will play the audio translation of the object. On the main screen, the user can select which language they want to use by clicking on an icon that has the language code on it (i.e. “en” for English).



Figures 1 and 2, showing a mockup of the planned UI in the main screen of the app.





Figures 3 and 4, showing a mockup of the planned UI for changing languages and settings.

Specifics

1) iOS UI and libraries (Swift)
    a) We are planning to design an iOS app that works on the iPhone X, as well as some previous models. Just like most of the apps on the market, we plan to design an attractive and intuitive user interface (UI) for it, which means designing good UIViewControllers in the code. The Swift portion of the project will also be responsible for activating the camera to allow the app to be able to detect objects and place AR elements, ie the translated labels for objects. Also, our app need to be able to switch between different UIViewControllers smoothly. This will require the use of different ViewControllers, and UIButtons that work with these ViewControllers.
    b) Additionally, Apple allows machine learning with CoreML. We may use this to train the app to recognize classes of objects based on an observed sample set. Our section on machine learning and object recognition has more details.

c) Another useful feature of Swift is the functionality that deals with animation. Apple is especially good with user interface animations. We can use this functionality to put emphasis on creating smooth animations on our UI.
d) Finally, we need to make our program compatible with different devices (iphone6s to iphoneX/ipadPro). To do this, we will need to learn more about the differences between different architectures of different devices as we become more familiar with Swift code.

2) Apple ARKit for AR
    a) We plan on using ARKit to enhance the users' experience by incorporating augmented reality elements into the translations. This tool requires a processor that is A9 or higher, which means that we need to make our app compatible with iphones with versions 6s to X. We will design the app in a way that will automatically match the aspect ratio of different devices. ARKit includes many features such as motion tracking, scene processing etc, and we likely won't be able to use all of these features. For now, we are only going to use its camera scene capturing and processing features in our app such that it can at least recognize the objects in the camera scene.
    b) ARKit also allows developers to do 3D modeling in the camera scenes. For example, a user can place a 3D model in the camera scene and rotate the view point without changing the shape of the 3D model, which means that the 3D model won't rotate synchronously with the viewpoint of the camera.We've thought about potentially using this feature in our app to help users better visualize the object.

3) Google Translate API for translation
    a) Once objects are recognized and a label can be assigned to them in the AR space, we need a way to translate the recognized object name into another language. For this, we plan on using Google Translate's API as a sort of digital dictionary that is handled automatically in the background. From the user's perspective, all they see is that an object in view of the camera is recognized by the app, and then appropriately labeled with a textbox in the language they chose. Additionally, we may include a feature where users can enter specific words they want to know into a text box and have them translated, similar to a digital pocket dictionary.
    b) The main benefit to using Google Translate's API is that we would not be limited in the number of languages our app could support; we would have access to all the languages Google currently supports, possibly even allowing our app to be universally marketable, allowing anyone to use the app, regardless of their native language, provided the user interface is designed to support their language.
    c) The main things to consider when using the Google Translate API are cost and attribution. Google charges a fee of $20 per 1 million characters when translating using their API, charged monthly and scaled by usage (so translating only half a million characters would only cost $10 at the end of that month). The Ethical and Intellectual Property Issues section below goes into

more detail regarding attribution details.
See the link below for more information regarding attribution:
https://cloud.google.com/translate/attribution

4) Object Recognition
    a) An integral part of this app is in actually recognizing the images that we get from ARKit and turning them into recognizable text. This will be done using a machine learning model that takes in images and tagged text corresponding to what the image shows and learns to produce accurate text from an image. This is an area that we haven't completely worked out our design for. One possibility is using our experience with python to train an object recognition model in Python, and then integrating that into to the main core app. We've done some research into this, and we've found a good database of tagged images called ImageNet that we should be able to use for this training process. Although this plays best into our expertise, it is possible that it may be tricky to integrate this model into the app. Another possibility is using Apple's machine learning library, called Core ML, to train our model. This would require more learning on our end but may be worth it to have a simpler integration with our app. A third option may be using a built-in feature of ARKit that has object recognition. We still have to do testing to see how good this feature is, and it is likely that it won't be suitable to our needs.
    b) Another part of the machine learning model is a planned feature allowing users to opt-in on allowing images that we fail to recognize to be tagged by the user and then sent to our machine learning model. We can then retrain and improve our model based on this new information. This brings up ethical issues, which are discussed in the next section.

Ethical and Intellectual Property Issues:

1) Translation from Google:

One potential issue is in using Google's Translate API to translate what we recognize the object to be. We could have issues for properly attributing our use of it, and there could be privacy concerns for users. Looking at Google's FAQ for attribution, we need to clearly show users that the results from the translation are provided by Google Translate, preferably using the logo. We should also clearly state that it's being used in any description of our app. These are important guidelines that we should adhere to in development, as it is unethical to use software without respecting the desires of the developers. Google also says that they do not look at, use, or share the text sent to them via their API, meaning that there are no privacy concerns with using their API.

2) Using images to send to training:

Another issue we see that could be seen as both an intellectual property and ethical issue is using users' images for retraining the model, if we end up going that route. As the images taken with a phone should belong to the user, we'll have to work out some kind of agreement for letting users give us rights to use their images to improve the model. This will probably take the form of some kind of of opt-in menu where users can agree to let us use

their images. If we didn't have something like this, it would clearly be unethical to use their images without their consent, especially if we didn't inform them about it. It would be disrespecting our users by essentially lying to them and taking things that belong to them. In order to respect their privacy, we should also strip any identifying information off of the image.

Change Log

- Added description of InterpretAR audio feature. We plan to allow the user the option to play an audio recording of the translated words to help facilitate learning by receiving an additional sensory element and to teach proper pronunciation.
- Updated the team roster to reflect changes in personnel.
- Updated the budget to include Google Translate pricing. Upon further investigation into the use of Google Translate's API, we found that Google charges a fee of $20 per 1 million characters given to translate. This will be important to consider when testing our app, as well as planning the future sustainability and possible launch of the app.
- A Gantt chart was added to the milestone session. This gives us a clear visual representation of our goals for the project as we move forward.