Group 4
Kawang Wong - Leader
Ricardo Deodutt - Documenter
Sai Ho Yip - Speaker
Alex Fashanu
Denzel Gustave

https://github.com/kawangwong/make-or-break-business

As a group we introduced ourselves

3:45 - 4:00pm
We set up a plan of action.
We all generated new keygen keys to share with the group leader.
So we all made a directory that is the same as the remote repository (make-or-break-business)

4:30pm
We still have errors connecting to github. We have made the Git remote repo public and some remade keys

4:35pm
Kawang is successfully connected.

4:43pm
Sai connected to the remote repo successfully from VM

4:48pm
Ricardo successfully connected to the remote repo.

4:50pm
We created a slack group and discuss different App ideas
Sai had a water drinking app idea Monolithic
Kawang had an idea of a waiting in line app Microservices
Ricardo had an idea of a banking application Microservices
Alex had a queue application where it would save peoples place in a queue.
We talk about how applications such as it should be convenient for users.

As a group we had a discussion. First we had to define what Monolithic vs Microservices was.

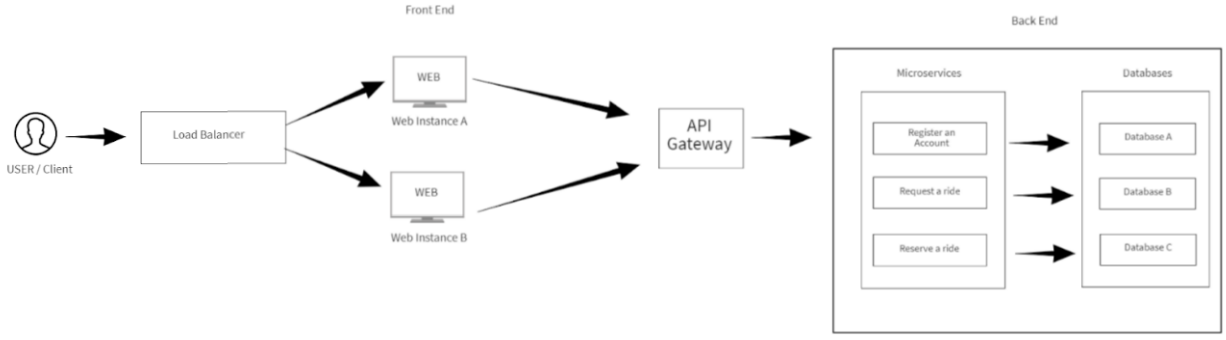
As a group we went with…
Monolithic code is when an entire application is one code.
Microservices is when monolithic code is broken down into multiple individual services.

A person would go with microservices because if one service fails, the entire application will not also fail.
Certain parts of the application can also be scaled up/down depending on the demand.
It is easier to isolate errors.
Microservices may require more resources. Monolithic services require less resources compared to microservices.
Monolithic applications can scale because there can be many instances of the code running.
The code base in monolithic applications is one uniform language.

7/20/21
5:20
Majority of the group met before class and discussed a game plan. We finally fixed Alex's issues (he has an issue with the SSH key and his using mac.). Alex got to push his initial ideas on his local repository. Alex told us his initial idea on having a queue app that is similar to Kawang's idea.

5:45
Once most of us got our ideas in, we discussed which idea we wanted to go with.

We wanted to follow this diagram of how our application wanted to run



So basically we went with Kawang's idea because it was the most unique.

The application is basically an app where basically you can have someone wait in line for you. Basically when there's a line for an item you want to get you can request for a person to come and wait in line for you to get the item. The person waiting in line for you would basically let you know when they get to the end of the line and let you know if they were able to get an item. If the person does get the item they would basically give the requester their requested item. Once everything has been done the app will grab the payment info of the requesters and process it + charge them service fee's + tax + whatever other charges that will be agreed upon before they made the request.

To Set up remote repository and let members connect on a VM
Leader creates the repository on GitHub
Leader goes into Repository settings. Then Manage access. Invite members to collaborate.

Change directory to ~/.ssh
Members create new keys using “ssh-keygen -t rsa -b 4096 -C “email@email.com””
(**Be careful copy and pasting the above code. Quotation “” get changed from Windows to Linux. Write out the code**)

Once created, share **public** key with the group leader
Group leader then puts the new public keys into his/her SSH Key settings in GitHub
(I believe the GitHub Repository also has to be public)

Once key is created, run “exec ssh-agent bash” (Run “ps -aux” to see if ssh agent is running)
Once that is ran, run “ssh-add <privatekeyname>” to add the private key to the SSH agent
Once that is finished go into working directory cd ~/Desktop/
git clone <SSH link> (ours is git@github.com:kawangwong/make-or-break-business.git)
Change directory into the cloned repository
git pull --all (this pulls all updates from all branches)
git fetch --all
touch <file>
git add .
git commit -m “inital file”
git push

Switch branches in command line using
git branch -M <branchname>

If you don’t see all files in a branch, use…
git checkout <branchname>
git pull

How to create a new branch in Command line using
git branch <branchname>

For MacOS users
To start SSH-Agent use …. eval "$(ssh-agent -s)"

If you come across this error -> This current branch main has multiple upstream branches, refusing to push.
git config remote.origin.push HEAD