ARTIFACTS TAB

Now, this presentation is just a small bit at the end to explain a little on the behind the scenes, much like this class focused on. The goal of this small presentation is to show that we did attempt to apply some of the ideas from this class to the project and we didn't wait til the day of to finish the project. First we'll be starting with the diagram on the left or the Feature Breakdown Structure. Which is an *extremely* simple overview of the entire project starting with the product vision. Then we'll move on to the context diagram, which will have a few pieces that should be at least somewhat interesting to see.

FBS TAB

Moving on here up first we have the Feature Breakdown Structure or FBS for short, which is, as advertised, and extremely basic overview of the whole project starting all the way from the vision.

I'm sure you all know what the vision is so I'll just skip that and move on to the Goals & Outcomes.

The Goals & Outcomes group is also relatively basic however it needs to answer the all-important question of how exactly the vision is going to get from writing to functionality? That is why the split of the structure starts here rather than the vision since we need the yeast and the dough to make the bread.

The Epics group breaks the Goals & Outcomes down into something a programmer could work with. Changing the goal to a deadline that needs to be completed. Pretty basic here but I'm sure you could

The Features tab are the ingredients to bake the cake.You need a login for students? Well you'll need a login for returning students and a registration for newcomers. Same idea as the selective messaging.

CONTEXT DIA TAB

Ok, moving onto the Context Diagram, there are a lot of moving parts here, but everyone's projects most likely have these whether you planned them in a similar way or imagined them differently. For these reasons, I'll only be touching on some of the important pieces and some interesting ones as well.

REG NEW USE TAB

First up we have the registration of new users. There isn't much to be said about this part, but the interesting part is how the profile pictures are made. Since we were not able to completely figure out how to give users the ability to provide profile pictures and for us to store them, we opted for using a randomizer along with profile pictures stored online. The pictures are then **COME BACK TO THIS!!!!!!** Other than that, we move on.

USER DB INT TAB

Up next is the User Database Interactor, which has the logic for querying our database, confirming user identities, and adding new students to the database. For the verification, like I'm sure you'd expect, it grabs the username & password from the login page when the user asks for it, queries the database and runs the username & password against the stored items for a match. If it doesn't have one, we return an error message and the user can try again. Registration acts in a similar manner instead of logging in, it simply checks to see if their username matches someone else's and provides an error if so.

USER DB TAB

This tab is another we can essentially skip over, but I thought it important to refresh some of those who aren't familiar with, didn't use, or were not a part of the creation of their group's database since this will be important to understand the next tab.

Users are as stated, and are provided to the database by the interactor to be stored for future use.

The ID is a unique identifier or the "key" that allows for unique storage so that other tables and other backend elements can access user information. The User-ID relationship is as simple as that, each user or value has their respective key, forming the most basic of key-value pairs.

MSG DB TAB

Now, it was important to have some understanding of databases because we're going to be talking a little about relational databases.

This database stores messages similar to the user database. However, what happens when we want the messages between who the student selects and their partner? That is where relational databases comes in.

A relational database is essentially a database that stores both ids from database A for access inside of database B. An equivalent of what we are doing here might be how a bank stores your money (or Database B) and they require your PIN or ID(Database A). In this case, when you deposit money, the money is deposited into database B and can be accessed with database A’s ID.

Now, the reason as to why our Message Database is relational is the exact same as the example I provided above. As you can see on the context diagram, when the student selects student2, a query is made to the database with the IDs of both students. The database uses this to confirm the users and accesses the messages between them, returning them back to the frontend through the message DB interactor. I thought this was pretty interesting so I hope you did too.

MSG DB INT TAB

I made that last slide sound like the conclusion, but I figured since I mentioned it in the last slide, I might as well talk about it. This interactor is pretty much the same as the user DB interactor, but even simpler as it simply needs to verify both users exist, and query the messages between them. With that, I think we’ve reached the end of our project and I thank you for your time.