2/18
MEETING NOTES

- Download solstice for next time
- Use case 3.1 has a mismatch w/ the assumption of UI layout for terminal
- Use cases should be more orthogonal with each other
- Will need to decide on a UI interface
- We are missing the purpose of how each use case helps in achieving the full scenario - need to fix this
- Think as if we are selling to investor
- Make sure there is a bridge between use cases and full scenario
- 1st use case - after testing this will allow us to add a UI which is relevant no matter what
- 2nd use case - completing this successfully allows us to move on to use case 3
- Use top-down to sell work, bottom-up to describe work
- Final presentation should be top-down
- Make sure we position our use cases properly in the big picture scenario
- Start mapping out functional and nonfunctional requirements
- Figure out our individual roles -
- Goce says that postgres is a good choice
- Figure out what data we will be storing - files, email, username,
- Write down formally why we chose postgres and any questions that we have so that we can answer them
- Make it explicit as to why use cases are necessary and how they fit into bigger picture
- Have a set timeline of when we will make decisions
- Foresee scenarios for individual testing vs integration testing - come up with at least one example for next week
- Come up with representative architecture, use case, conceptual diagrams

OBJECTIVES FOR NEXT MEETING
- Make diagrams - architecture, use case, concept
- Come up with individual testing vs integration testing example
- Justify ourselves as to what our project has to offer
- ** one thing we could do is go to git and find who has similar projects to ours - what could sell us is changing the experience of something that is already there - add new features that will make the old thing better
- Come up with a more formal project description - enable some type of matching that we are not able to do in git - “which pairs are most similar?”
    - Basically git currently enables us to have a single similarity set, but for our project we want to “join” them to differentiate ourselves. Join a project database with itself and find similarities - find a pure match of tokens
    - In this case we can assume that every project has a description about what it is.
    - The problem is how do we define similarity (pick simplest possible solution - parse one, parse the other, do they have words that match?”

    - Then do the join
- Explain / justify how we link the simple use cases with the big picture - identify our roles within the big picture - switch assumptions to functional/nonfunctional requirements, make samples of unit tests vs integration tests that we foresee. Then we will move onto diagrams and identifying timelines.