Some requirements from video about midterm

Find out what modules team has done to see what language and route we can do, technical part it important and does need to be workable.

General notes:

- Need to understand the nature of the problem we trying to solve.
- How and why we trying to solve a type of problem
- Defining a concept, and clearly identifying how and why we going to develop this thing
- Note other similar software that exists, and then why a particular software would be interesting, different, useful to someone, valuable, add utility to someone's life, or improve accessibility or usability.
  Describe how it would do these things also
- Explain why we using a specific language, create a full stack application, front end and backend
- Github logs, they use it to verify, though most process must be documented in the report. There might not be code for the midterm, though we can do version control for like resources and mood boards and images etc. you don't have to for the midterm though.
- Talk about how we resolve different ideas
- How we communicate, as how we work in slack
- It's ideal that the midterm is to be certain, you don't want to change things when doing the end year, as the midterm is to prepare for this. Only smaller changes. Don't throw away ideas. The scope should be pretty direct
- Gannt charts, with mile stones, what we need to do before something else

Project:

- Deliverable components
  What we are actually going to be building
  The work required to do to build this project to completion (think this can be like what type of research, and random things)
  What process we are going to take
  How we going to build it
  What things we are going to need to do
- Breakdowns from month to month, week to week, day to day
  Defined time scale of work
- Formal specification of a system
  Telling them what we are building, in the forms of a model, in technical terms
  Using UML techniques
    - Functional
    - Technical specs
- Scope
  Clearly defined scope, What we are going to be doing, and what we are not going to be doing.
  So if we building a part of a system we need to define what parts we will build and what parts we won't be building (IMPORTANT). So e.g. if we doing language translation, we might only translate English to Spanish, though we won't do German or French, and we won't be using machine learning.
- Requirement illustration
  Going out and getting what stakeholders want from the system, through research methods and market analysis, Looking at other applications, asking questions, interviews etc.
  User centered Design,
  Evidence based reasoning, Show why we are using a particular method and why it is the best method.
  Maybe passwords, we might need two factor authentication and why, or why not
  There could be data and legal requirements specified

- Description of your approach
  Evidence of how we are doing things, and how we working as a team, and how we coming together. Talk about the process of working in a team in a way, how doing things in a team might have worked well and things that didn't work well (so maybe like splitting 5 interviews, didn't work well because ended up interviewing the same range of people creating a bias approach, something like that not that your teams bad)
- Prototypes
  Some basic prototypes, pen and paper is fine. And maybe some small tests of the prototypes. And shows utility.
- Assumption testing
  How can you use feedback and resources from your analysis to make something more valuable (I think called assumption as we won't know if it's more valuable until it's been made and evaluated)
  Though put the prototypes in front of a user and try gain evaluations from users. Make sure we building the right form of system. Buttons are in the right place, how long it takes for user to do it, etc. Do they work or don't they work.
- Critical evaluation of the project in the current state. And proposed software project
  So at the end of all the stuff we need to state this is what we going to build, and this is how we will do it (IMPORTANT)
- Clear structured document

Deliverables

It should be clear from your proposal:

. What you are delivering
. Which set of processes are useful
. The timescale/dependencies relationship
. The specification (specs) of your system
. The scope, limitations and risks
. Identification of requirements
. Where your tool sits in a marketplace
. Evidence of prototyping and iteration
. Some simple tests/evaluation steps
. A strong rhetoric that informs and describes

NOTES from example project

- Description of who everyone is, maybe note who would be working on different components and why.
- How did the group come up with proposals and ideas?
- Why did we lean towards a specific idea?
- Defining the scope, note papers that we read into that might say what is being done and what isn't being done. So I think focus the scope on a section of an app that hasn't been done yet
- Descriptive questions, to figure out aims. So for e.g. how can we make it easier to make payments in restaurants, or how we can make shopping in stores faster. Categories like reliability, accessibility, adaptability, etc.
- Who we are designing this system for. E.g. a music app could be
  - **Passive listeners** – no interest in engaging with music other than listening "passively." Possibly non-technical. Possibly non-technological.
  - **Non-passive listeners** – desire to interact with music. Either technical (musically) or technological but not both.
  - **Engineers** – Existing workflow of engaging with production tools and technologies. Both technical and technological.

So we need to figure out which groups we want to design for.
Also how we identify the target group

- How existing interface can be improved
- How we build one-size fits all, like a universal design
  focusing on familiar features (like the pause button) and research what they are
- What type of data do we need?
- A design goal could be trying to make users feel they have some ownership in the app, like different colour themes
- Would there be benefit to adding further functionality, and what's the cost implication