# Tech stack summary and decision log:

Selected options are in **bold** next to each bullet point

**Context that affected our decision making:**

- Our team is concerned about the time that this course will take up, so any reductions in development/setup time is appreciated.
- Our team is worried that our workload from other courses will be high.
- Our team is already familiar with certain tech stacks and would prefer to use existing knowledge if applicable to save time and minimize risk.

- Frontend: **N/A**
    - Will not be required as our project is just an API for other apps

- Backend: **Node.js**
    - *Option 1:* Node.js
        - Every team member has prior experience
        - Easy integration with MongoDB
    - *Option 2:* Flask
        - Used only by James in the past
        - Unfamiliar with DB integrations, possible risk
    - Decided to go with Node.js as we felt more comfortable with it + our database integration over Flask.

- Database: **MongoDB**
    - *Context:* We will only need to save ActivityDefinitions and PlanDefinitions into a database
        - id properties aren't unique across both activity and plan definitions, so we will create separate DB schemas for each
        - Our database will only store JSON objects, so it's not necesary to use a relational database
    - *Option 1:* MongoDB
        - Free to use
        - Fits with our data well (ie. json files).
        - Mongoose allows easy integration with Node.js
        - Team has familiarity working with mongo in the past
        - Easily define schemas for backend use

    - *Option 2: Postgresql*
        - Didn't fit our data as well.
    - Decided to go with MongoDB due to the json files being a better match for non-relational db, less risk in familiarity and easy integration.

- Hosting service: **Heroku**
    - *Option 1:* Firebase
        - Free to use
        - Has its own DB and authentication
        - Limited flexibility, forces the use of a proprietary DB (no mongo allowed)
    - *Option 2:* Heroku
        - Free to use
        - Prior experience for team members who took CSC309
        - Compared to Firebase, Heroku is more flexible and easier to integrate with other services
    - We had no need for Firebase's own DB and authentication as we are already familiar with MongoDB. In addition, Heroku is more flexible and easier to migrate in case our code needs to be used by Alex.

- Container service: **Docker**
    - *Option 1:* Vagrant
        - Can run in non-linux environments because it functions as a VM
    - *Option 2:* Docker
        - Team has prior experience with Docker
        - Requires less resources and setup since we only need to load libraries
    - App will be run in a linux environment, so we're choosing docker because there's less risk with unfamiliarity.

- Testing: **Jest**
    - *Option 1:* Jest
        - Self contained
        - Most simple to use
    - *Option 2:* Mocha
        - Does not have its own assertion libraries, involves more dependencies
            - Chai is required for assertion
            - Sinon is used for spies/stubs
    - *Option 3:* Jasmine
        - Unable to run tests without a third party
    - Decided to go with Jest because the scope of the project isn't large enough to justify the more heavyweight options