- ***A high-level description (about one paragraph) of the software you are building and who the target user groups are**. Separately, explain how each user group is expected to use the software differently. (Recall that in order for two users to be considered as being in two different user groups, they must use the system differently either via a functional or non-functional system requirement.)*

We are building a mobile friendly website to automate the process of obtaining campfire wood for the general public. On the website, the user will choose the location (from a list of available locations) and the number of bags of firewood desired. Then, they will automatically receive an invoice that is produced based on pricing and quantity ordered. The website then makes payment or retains an invoice for cash payment upon pick up. The receipt is generated automatically.

- ***An overview of your system architecture presented as a data flow diagram (DFD), at both levels 0 and level 1**, along with accompanying descriptions for each diagram. You must be very specific in this DFD so to design it in a way that supports an incremental and iterative development process. You can read an intro on DFDs and adopt its notation. Remember that a process does something, so be sure to use a verb to label your processes. In describing your DFD, **highlight clearly which components will be delivered for each of the following milestones: peer testing #1, peer testing #2, and the final product**. You may additionally wish to include extra components and label them as bonus features that may be developed if time allows.*

Okay

- ***A detailed list of functional requirements for each milestone.** Be sure to separate out the functional requirements for each system component based on the target milestone.*

Peer testing 1 Milestone: We should be hosting a mobile friendly store front with minimal backend interaction

Peer testing 2 Milestone: We should have a GOOD looking mobile friendly store front with full back end integration, fully fledged database schema with the purchase form filled out (no actual payment / order fulfilment)

Final deliverable: All previous milestone goals as well as true payment processing and invoice generation fully fledged

***A detailed list of non-functional requirements and environmental constraints** for the overall project. The environmental constraints are similar to non-functional requirements and explains the constrains related to system's environment. Some examples are interfaces to other systems, COTS components,* *programming language, system's location, size, power consumption, humidity, maintenance accessibility, ...etc.*

The user interface looks nice and is easy to use for both old and young people. The site stores information securely. The website is compatible with all popular browsers (desktop and mobile).

- ***Identify the tech stack you plan to use****. If this is the tech stack required by your client, then state that in the document.* ***If your client is flexible, you must indicate a clear rationale of the choice made****. This means an extra page with a table of at least 3 options and documenting all pros and cons for each before letting your client make a choice.*

MERN (MongoDB, Express, React, NodeJS). We made this decision because it will fit the clients needs quite well in a modern standard. MongoDB allows the database and its objects to be easily understood by both us as developers and the client. Express and Node to serve the front end and communicate with the database because javascript is a very easy language to make fast prototypes with and is a perfect fit for the interactivity that the client needs. React will be used on the front end due to the client wanting a mobile friendly site, React will allow us to do this with ease.

- ***An explanation of how you will test the developed features of your system (in accordance with the tech stack chosen) as well as the method you will adopt to ensure continuous integration****. Don't just use a technical term to describe the method; write a few sentences to explain the tools and techniques that will be involved. Read up on regression testing, and develop a feasible plan that ensures when new code is added to your project that your project didn't break anything that was working before.*

The stack we are using gives us lots of flexibility for testing based on the availability of testing libraries for JS. On the backend we can utilise unit tests to ensure certain aspects of our API work. On the front end we can use chrome's developer tools as well as run regression tests with different browsers to ensure nothing breaks, and if something does, we can isolate that incident. We will use postman, a REST client to ensure our API is secure and can’t be spoofed by hackers who would steal information.