1.1  
\*  
**The application of a** systematic, disciplined, **quantifiable** approach to developemnet, operation, and maintenance of software; that is the approach of engineering to software.

1.2  
CBSE: An approach that reuses components to develop a software solution  
An example of this is reusing components: Write Feedback Report + Inspect Work Done  
  
-Decreased lead Time: Reusing the receive tips components, will allow funding to occur sooner, as the system is reliant on tips to fund the system.  
-Increased ROI: The buying of or reusing of a previous notify user component will decrease development time.

**Know the 3 others:** Maintenance, quality, leverage costs: spread out costs.

1.3

**Control**How is control managed? We can see how control is managed by looking at the inspect work done and write feedback report components. In order for the Reviewer to write a report, the work must first be inspected.  
  
How is control Transferred? We can see how control is transfered by looking at the inspect work done and write feedback report components. Once the inspect work is done, the component transfers control to the write feedback report.  
  
**Hierarchy: Which components have the most control  
Topology: how arranged  
Synchronised: is it in harmony.**

**Data**  
Data Communication: Once Work is done, data is sent over to the Write Feedback Report so that it may make use of data for the report (which decides on whether the work is approved or rejected).  
  
Data Flow: We can see data flows from the inspect work to write feedback. A choice of whether the work is approved or rejected occurs. If rejected, then a request for the work to be redone occurs. This will lead to that work being inspected once finished, showing a cyclic nature of the data flow.  
 **Mode of transfer?  
Data components  
Passive or active**

**Q2**

-Aim of level (3):  
-example of consideration made (1):  
**DATA**  
**Data vs Information.** Data is raw facts, while information is processed data (meaningful in some way).  
  
**Data Continuity**  
In the system, you will look at the raw data that your system takes in, and then how that data is processed by your system into meaningful data that the user can view/use.   
What are you putting n the system, and what are you getting out? How can this be improved? What data can be represented/needed for the system to grow.  
  
**How is data represented.**Should it be represented direct or indirect? Users must easily be able to understand the data eg Tables are not a good way to ouput data to users.

**Consideration**

When looking at the Insepction subsystem, and how data is transformed from inspect work to feedback to reject work to request work. Data is transformed at each of those steps, and then viewed by the user for them to make a decision. Therefore at each of these steps, only the needed processed information should be displayed eg at Request drone work, you only see that the work was rejected, not all of the inspection data.

**ARCHITECTURE**

Overall view of solution from core level: How do the classes talk to eachother?

Defining what patterns can be used to achieve the requirements in a specific domain.

Defining the major structural elements of the software and their relationships.  
  
**Consideration**: As the components in the system require up to date information, the data stored must be active (components must know of any changes in stored data). This will effect the architectural style chosen. We can see this in the auditing subsystem. The data from receive tips and pay maintenance are very important when the allocate funds occurs, as knowing your incomes/expense/total money in bank is extremely importnatn for that.

**INTERFACE**The overall design/appearance of the system.

Definining the flow of information in, out and between the system.

Defining the   
Internal interfaces: How components interact with eachotehr  
External interfaces: How components talk to DB.   
User Interface: How the user interacts with system

**Consideration**

It is important to note that appearance is subjective, and especially seeing as how the system is funded by citizens, the citizens must like the appearance of the app. A good way to find their most preferred appearance would be to do a poll on social media with a few different colour pallets/designs.

**COMPONENT\*\*\*\*\*\*\***

Define the Internal detail of component  
  
Transforms structural elements of the software architecture into a procedural description of software components.  
  
Information obtained from the class-based models and behavioral models serve as the basis for component design.  
  
  
**Consideration**

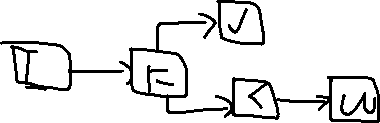
**\*\*\*\*\*\*\***

Q3.1  
  
Separation of concers: A Modularity is where a problem (Inspection subsystem), can be broken down into smaller problems (Inspection subsystem components eg Inspect Work Done), to allow reusability of components, an easier development/solving of that problem (easier to develop 5 easy things, than 1 very difficult/complex thing) by reducing complexity. As the system requires constant feedback from users resulting in updates, having modular components which can be individually updated will result in less work needing to be done compared to updating an entire eg Inspection Subsytem (if it wasn’t modular). Relates to modularity, functional independence, rerefinement.  
  
Abstraction: The hiding of complexity/unnceccesary details, allowing the implementation of a more complex system, without needing to understand how it works. An example of this is Provide Rating for the Citizen. The citizen only needs to give a 0-10 rating, which is simple. They do not need to know how their rating is aggregated or processed by the system, making it easier/simpler to use.  
  
Abstraction is broken down into  
Procedural abstraction: The sequence of steps when completing a function. (eg Allocate Funds – What actually needs to happen is total funds available must be checked, this is compared to the requested amount, )  
Data abstraction: Collection of data that describes the object (eg User – userID, userName, userPassword)

3.2

Data Flow architecture: This design allows you to go directly from the design, to implementation phase. This will speed up the rollout of the program, which is paramount as funding comes from tips – if the development takes too long, they might run out of funding.  
  
As the system a lot of interaction between components in a sequence eg After a pothole a has been fixed … system should notify… repaired. Before we can allow citizens… review… inspect… (mention the diagrams and how theyre step by step).  
\*

3.3



The data flow architecture shows the how data is transformed through as series of connected components. In the above case, we can see how data from inspecting the work done, is transferred and then transformed in the report component. The report component data will result in it being either accepted, or rejected. If rejected, the data is then transferred to the request drone work component (where there will be reasons why the work must be redone/what must be fixed/what wasn’t done ect). This shows how data from the inspect work done, is transformed through multiple components into more useful/processed data.  
  
A disadvantage of the style, is that if one component goes down, the whole thing wont work. An example of this is if the reject component goes down, requesting drone work component cannot be used.

**Each component works independently: They do not know how the other components nheiboring components work  
  
Each component expects data input in a certain form, and has an output.**