Design of the product (software)

1. UML diagrams.

1.1 Class diagram- A class diagram depicts classes and their interrelationship. It is used to describe the structure and behaviour in the use cases. It is used for requirement capturing and end-user interaction.

The class diagram has been attached as a separate file.

1.2 Use case diagram- Use case diagrams provide a visual way to document user goals and explore possible functionality.

The use case diagram has been attached in SRS document.

1.3Activity diagram- This diagram describes the state of activities by showing the sequence of activities performs. It gives an indication of things that happen in parallel and things that happen in sequence.

The activity diagram has been attached as a separate file.

2. Data Flow diagrams. (Context level DFD, level-1 DFD, level-2 DFD)

Data flow diagram comes under flow oriented modelling. It is a structured analysis technique that employs a set of visual representation of the data that moves through the organization, the paths through which data moves, and the processes that produce, use, and transform data.

Diagram layering and process refinement:

Context level diagram- shows just the inputs and outputs of the system.

Level 0 diagram- decomposes the process into major sub processes and identifies what data flows between them.

Child diagrams- increase the level of detail.

A DFD that comprises of all these aspects is attached as a separate file.

3. User Interface Design (Adhering to the three golden rules suggested by Theo Mandel mention clearly on how you have adhered to these rules by adding snapshots.)

Rule 1.Place the user in control

User can be kept in control by defining interaction modes in such a way that does not force a user into unnecessary or undesirable actions. We have provided booking option in the view room availability page for the user, view room availability option in the allocation page provided to the manager. We hide the technical details of room booking procedure and provide means for direct interaction with objects that appear on the screen.

Rule 2.Reduce the user's memory load

The interface reduces the user's requirement to remember past actions and results. The user is expected to remember his username and password to login. From there on he can easily use the mailbox option to see his status of booking. The visual layout of the user interface contains well-understood visual cues that the user can relate to real world actions. The process of booking room is done in a progressive fashion. Once the user accepts the undertaking he is allowed to enter the guest details and submit the request form.

Rule 3.Make the interface consistent

All visual information is organized according to a design standard that is maintained throughout all the screen displays. Input mechanisms are constrained and that is used consistently throughout the application. The interface provides indicators that enable the user to know the context of work at hand.

4. Component design (Adherence to at least three basic design principles for designing class based components. Mention clearly by adding snapshots.)

A software component is a modular building block for computer software. It is modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. A component communicates and collaborates with other components and entities outside the boundaries of the system. SOLID principle is an acronym that stands for 5 key principles in software development. They are S – Single Responsibility Principle, O – Open Closed Principle, L – Liskov Substitution Principle, I – Interface Segregation Principle and D – Dependency Inversion Principle.

4.1 Open-closed principle (OCP) - Module should be open for extension but closed for modification. This principle is used in creation of a global user class. Presently we have 3 categories of users namely user, authorizer and manager but in future if we also needed to add admin then a fourth user called admin can easily be created.

4.2 Interface segregation principle (ISP) – Many client specific interfaces are better than one general purpose interface. Specialized interfaces have to be created to serve major categories of clients. Only operations specific to a particular category of clients should be specified in the interface. This principle is used in creation of separate classes for user, manager and authorizer. The user need not have an allocate room option because it is irrelevant to him, similarly manager need

not have options to cancel booking. So interface segregation is used so that a client's interface has only those operations that are relevant to the particular client.

4.3 Single responsibility principle- The class should have only one reason to change and only one responsibility to take care of. The principle is used in the class called guest whose only duty it to collect the guest details so that room can be booked. The guest class has a single responsibility.

5. Coupling and cohesion (Identify the types of coupling and cohesion used in development of software and support your explanation with screenshots.)

Cohesion indicates the relative functional strength of the module.

Coupling indicates the relative interdependence among the modules.

The classes in our system design and their cohesion and coupling types:

a. Global User-Functional cohesion and no coupling.

b. Guest-Functional cohesion and no coupling.

c. User-Procedural cohesion and common coupling.

d. Manager-Sequential cohesion and common coupling.

e. Authorizer-Procedural cohesion and common coupling.

f. Guest house-Communicational cohesion and no coupling.

Procedural cohesion-Module performs a series of actions related by procedure to be followed by product.

Communicational cohesion- Module performs a series of actions related by procedure to be followed by product, but in addition all the actions operate on same data.

Sequential cohesion-Output of one component is input to another.

Functional cohesion-Module performs exactly one action.

Common coupling-Two modules have write access to the same global variable.

Coding

1. Highlight 'do' and 'do not' of good coding style of the implementation issues which you have followed in your development (Explain with example.)

We have given programs with good clarity with the help of structured programming approach so that it can be easily verified. It enhances the readability of the code and helps in debugging, testing, documentation and modification of programs.

We have used single entry- single exit constructs and break, go to statements are not used.

There are no loops with five or more levels of nesting. We have used standard, agreed-upon control constructs and defined user-defined data types to model entities in the problem domain.

We have used indentation, parentheses, blank spaces, and blank lines to enhance readability (source code formatting).

We have not used null-then statements and then-if statements. There are no routines having five or more parameters and we have not used identifiers for multiple purposes. These help avoiding obscure side effects.

2. Software Quality metrics followed:
We aimed at achieving a web app that will have functionality, quality, reliability and maintainability.