Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400050

Division: C2
Batch: C23
Roll Number: 1902112

Certify that Mr. Tushar Nankani

of Computer Engineering Department, Semester V with Roll No. 1902112 has completed a course of the necessary experiments in the subject **Software Engineering Lab (CSL501)** under my supervision in the Thadomal Shahani Engineering College Laboratory in the **year 2020 - 2021**

Teacher In-Charge
PROF. AEJAZ KHAN

Head of the Department
Prof. Tanuja Sarode

Date : 10/18/2021

Principal : Dr G.T.Thampi

List of Experiments
T.E. (Comp.) Sem V
Subject: Software Engineering Lab (CSL501)

Exp No.	Title of Experiments	Date
1	Write a detailed Problem statement for any case-study of your choice. Justify which process model would be best suited to apply on it.	22/7/21
2	Application of agile process model on the project. (JIRA)	29/7/21
3	Develop SRS document in IEEE format for the project.	5/8/21
4	Develop a Risk Mitigation, Monitoring and Management Plan (RMMM) for the project.	12/8/21
5	Identify scenarios & develop Use case diagram for the project.	19/8/21
6	Develop Activity & State diagram for the project.	26/8/21
7	Develop data flow diagram for the project.	2/9/21
8	Create a project schedule using Gantt chart.	9/9/21
9	Conduct Function Point Analysis for the project.	16/9/21
10	Application of COCOMO model for cost estimation of the project.	23/9/21
11	Case study: GitHub for version control.	30/9/21
12	Develop test cases for the project using white box testing. (Junit)	7/10/21

List of Assignments

Assignment No.	Title of Assignments	Date
1	Architectural design – Explain in detail each type with an example.	14/10/21
2	Write detailed note on Kanban Process Model.	14/10/21

EXPERIMENT – 1

Aim: To prepare a detailed problem statement for the selected mini project and to identify a suitable software model for the same.

Title: Airport Management System

Problem Statement:

Airport management systems are those systems that enhance operational performance, addressing a spectrum of processes – from data management, flight information display, billing to resource optimization, system integration and operational improvement to make collaborative decision making a reality. The data and the aspects pertaining to an airport can be overwhelming and thus, an airport management system has been put into place so that handling day-to-day activities at the airport becomes much simpler.

When an airport integrates its information systems, everyone benefits, from airport personnel to airline employees and passengers. Automatically exchanging real time information among systems results in more efficient airport operations. It speeds the flow of passengers and aircraft through the airport. Managers with access to information about all airport activity have the key to smooth, productive operations. They allocate resources more effectively and improve on-time performance. Today's airport authorities are faced with non-stop passenger growth. Short waiting times, constant access to up-to-date information and smooth processes guarantees that rising passenger demands will be met. This is true for large international airports as well as smaller regional and general aviation airports.

Thus, the overall management system consists of many entities, basically an object that has its own independent existence. For such objects, data can be stored. Those entities will have their own properties, or attributes, among which the information will be split and stored:

- Employee entity will contain the details of the employee such as empID, name, address, sex, description, salary, age, dob and employee phone number.
- The second entity is the airport entity, which will have airport_name, city and state as the components.
- The third entity, that is the airline entity, will consist of airline_id, airline_name.
- A flight entity will have details like flightno, source, destination, connected, arrival, departure, duration, status.

- Just like the employee entity, there should also be a passenger entity solely dedicated to recording the details of the passengers, such as passportNo, name, address, sex, DOB, age, TicketNo.

- Finally, even a ticket can be an entity since the ticket contains a lot of relevant information such as ticketNo, airlineName, price, seatNo, class, arrival, departure, duration, destination source, passportNo, name.

EXPERIMENT – 2

AIM: Application of Agile Process Model on the project.

THEORY:

Agile Methodology meaning a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Agile Models:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- eXtreme Programming(XP)

Agile Model is used when:

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

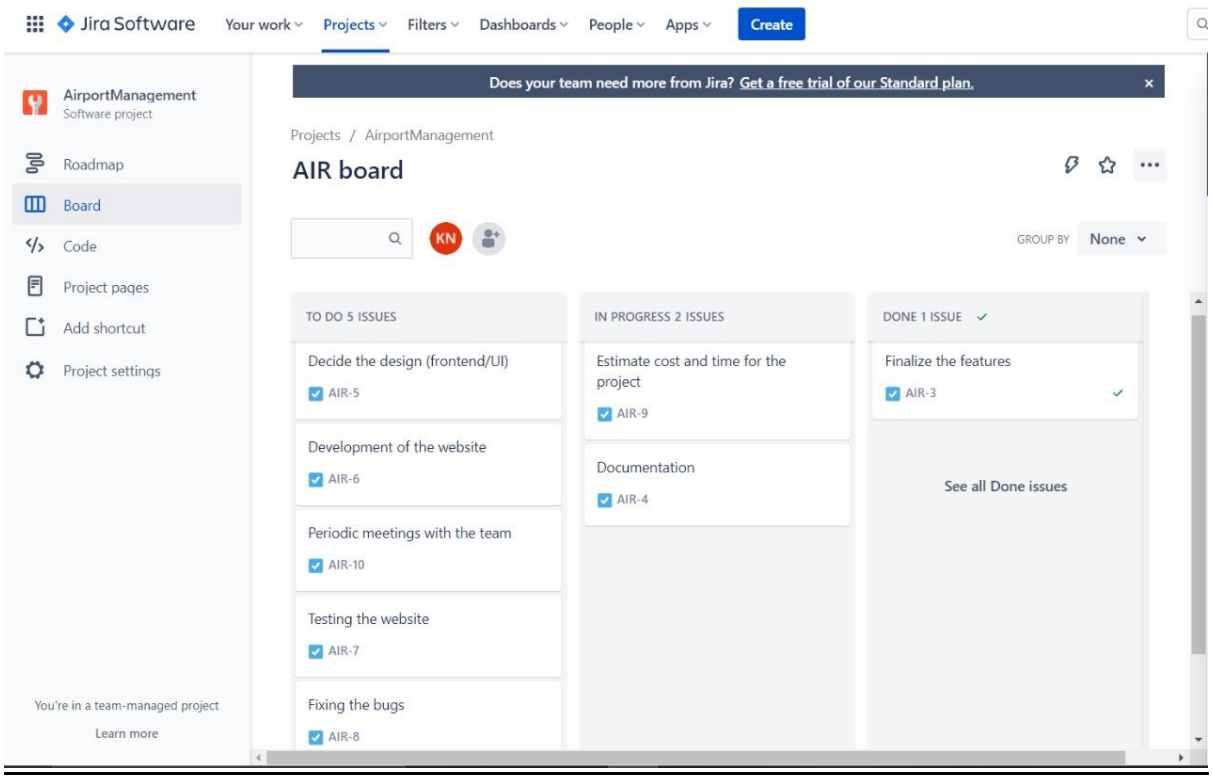
Advantages of Agile Method:

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

Disadvantages of Agile Model:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

OUTPUT:



Software Requirements Specification

for

Airport Management System

Version 1.0 approved

Prepared by

Kavya Nair
Parth Namdev
Tushar Nankani

Thadomal Shahani Engineering College

26-08-2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1.Introduction

1.1 Purpose

This software requirement specification (SRS) document describes the functional and nonfunctional requirements of the Airport Management System (AMS) release version 1.0. Business opportunity and objectives are briefly summarized followed by detailed description of the system's scope, vision, use case, features and other related requirement issues. The purpose therefore, happens to be the analysis of the system, and the need for such a system to be in place. This document will serve as a means of tracking the status of the project and intended to be used as a reference for group members.

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

1.2 Document Conventions

The font size and the font style convention that is followed throughout the document is as given below:-
Heading - 18, Times New Roman, Bold
Subheading - 14, Times New Roman, Bold
Content - 12, Arial
Here, every requirement statement is to have its own priority.

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

1.3 Intended Audience and Reading Suggestions

This software requirement specification (SRS) document describes the work flow of the airport management system, so the main audience of this SRS document are the passengers, the staff of the airport and all the software developers working on this project. The flow of this document is followed by a detailed description of the system's scope, vision, use case, features and other related requirement issues. It describes what the rest of this SRS contains and how it is organized. This begins with the overview sections and then proceeds through various sections.

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

1.4 Product Scope

Airport management systems are those systems that enhance operational performance, addressing a spectrum of processes – from data management, flight information display, billing to resource optimization, system integration and operational improvement to make collaborative

decision making a reality. The data and the aspects pertaining to an airport can be overwhelming and thus, an airport management system has been put into place so that handling day-to-day activities at the airport becomes much simpler.

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

1.5 References

[1] Airport Management Systems - Data Analytics for actionable insights on operational performance : https://adbsafegate.com/airport-management-systems/

[2] Airport Information Management System - Brochure by Navayuga Infotech : http://www.navayugainfotech.com/assets/brochure/airport-management-system.pdf

[3] Slideshare - Airport Management System : https://www.slideshare.net/VikasSingh958/srs-on-airline-reservation-system

[4] Scribd - Airport Management System : Airport Management System | PDF | Use Case | Databases (scribd.com)

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2. Overall

2.1 Product Perspective

The "Airport Management System" software is an independent application. It is a self-contained product. The system interfaces, user interfaces and hardware interfaces related with this software are defined as follows.

- System Interfaces: The client systems should be able to share the data available in the database through the network connection.
- User Interfaces: The screen formats and menu structure should be in such a way that even users will find it easy to use. The product must be user-friendly and very interactive. The functionality provided by the system like displaying error messages should adapt itself to the different users of the software.
- The system would require disk space of 10 GB and a 256 MB HDD and 64MB RAM for client systems.
- The users can first make a reservation in a particular flight for a particular date and time. The system provides the customer with a pin code which gives him access to either make any changes in his reservation or cancel reservation. These must also be backed up with data to enable easy recovery from any features.
- The "Airport Management System" software is an independent and self-contained product and no modifications are required to adapt to a particular installation.

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

The major functions include:

- Providing flight details
- Flight bookings for a particular destination, date and time and also provided with a pin code.
- Allowing the customer to modify or cancel his reservation provided the correct pin code is given.
- Displaying a report of the number of people flying in a particular flight.

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

2.3 User Classes and Characteristics

Passengers: A passenger is any person who uses AMS to book air flight ticket(s). Passengers have Internet access to AMS Internet-based user interface to book their flights.The basic needs of the passengers are:

- They may want to travel among all airports in India, prefer any date and time, select any airline as well as various demands such as arrival time, flight class etc.
- They require an easy-to-use and trustable online air flight reservation system.
- They need to be fully informed about all available flights before booking, secure assurance when making reservations and flexibility to view, modify or cancel their reservations at any time after they have done the reservations.

Administrator: An administrator may be a dedicated staff whose responsibility is managing AFIRS's back-end databases.

- They may wish to update/add/delete any information in the existing reservation database or user registration database.
- They may need to create and manage temporary views of fetched records from the databases of airlines and airports.
- They also need to be convinced that their operations will not affect the correctness of user transactions.

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

- AMS should be accessed using any popular versions of the following Web browsers: Microsoft Internet Explorer, Mozilla Firefox, Netscape, Opera and Google Chrome. AFIRS Customers Administrator search flight book flight flights manage flight information updates airport information Airline Database Airport Database Software Requirements Specification for AMS Page 4
- AMS should be able to run on Apache Tomcat Web server configured in a stable Linux/Unix/Windows machine.
- AMS works with Oracle 10i database management system.

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

- The system database design will be based on ER modeling which will in turn transfer to database schema formulated using SQL DDL statements.
- All HTML code should conform to the HTML 5.0 standard and CSS code should conform to the CSS 3.0 statements.

- Requires 256 MB on-board memory.
- Based completely on Windows functionality platform.
- The software should be portable and must be inaccessible to unauthorized users.

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

- There will be a user guide to explain to new users how to use AMS.
- New users can look for help online such as tutorials on youtube etc.
- The system will be designed as user friendly as possible.

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

It is assumed that the details of the cost of the ticket are already known to the customer. Future changes like providing different types of flights with different classes like business class, economic class will allow the customers to benefit from one facility.

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. External Interface Requirements

3.1 User Interfaces

The interface must be easy to understand. The user interface includes

- SCREEN FORMATS/ORGANIZATION: The introductory screen will be the first to be displayed which will allow the users to choose either of the two options, viewing flight detail or booking a ticket.
- WINDOW FORMAT/ORGANIZATION: When the user chooses some other option, then the information pertaining to that choice will be displayed in a new window which ensures multiple windows to be visible on the screen and the users can switch between them.
- DATA FORMAT: The data entered by the users will be alphanumeric.
- END MESSAGES: When there are some exceptions raising errors likeentering invalid details, then error messages will be displayed prompting the users to re-enter the details.

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

The system must basically support certain input and output devices. Their descriptions are as follows.

Name of Item	Description of Purpose	Source of Input/ Description of output
Key board	To accept data from user like pin code, personal details, flight details	Source of Input
Printer	To print the bookings mode E.g.: Destination chosen with date and timings	Destination of Output

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the

nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

Flight database system:

- The AMS should transmit and store the detailed properties (quantities, flight number, departure/return time/date, seat, location) of ticket items for the potential requests from users.
- The Flight database system should keep updated information about the availability of ticket and flight information to determine whether a requested reservation is available or not.
- When the AMS makes the query about ticket information with constraints, the Flight database system should provide all the information that satisfies the query and needs of the passenger.
- The Flight database system should accept information update operation flows coming from the administrator interface system.

User query/view system:

The user query/view system should communicate with Flight/ticket inventory/database system through a programming interface for the following operations.

- To allow a user to post various queries about ticket information available at present. User can use any combination available
- To show the user the resulting list of query. This should allow users to further sort the list by subfields of the ticket information (e.g. price). The view should be friendly and flexible.
- To allow a user to place a reservation.
- To detect the conflict if a user placed two incompatible reservations.
- To allow a user to view his/her reservation.
- To allow the user to modify his/her reservation.
- To allow a user to manage his/her account in AFIRS.

Administrator interface system:

The administrator interface system should communicate with the Flight database system through a programming interface for the following operations:

- To allow administrator Update/Add/Delete Flight information.
- To allow administrator Update/Add/Delete User Reservations.

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

- Every client system connected through LAN establishes a communication only with the server and not with any client system. An LAN of 10 Mbps is used.
- The AMS is a Web-database system, all kinds of user-system interaction is presented by a user-side web browser. So the communication standard is HTTP protocol.

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

4. System Features

This section outlines the low level details of each system function of the AMS Reservation System. It sets priorities of each function so that developers and AMS stakeholders know what will be accomplished first and what functions will be available for review first.

4.1 Login

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1.1 Description and Priority

The user will be prompted by the AMS Reservation application to enter a username and password on a login screen. Upon submission of valid login credentials, the data will be verified by the AMS Servers. Verification of the user is important because the devices may be lost or accessed by someone other than the authorized user. Because this application includes payment data and other sensitive materials, ensuring that only authorized users can access this data is crucial. Passengers and AMS is responsible for the protection of login credentials. This feature is of highest priority.

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

After a secure session is established, a login screen will be displayed to the user. The user will enter a username and password, and then select a button that says, “Login” to submit their credentials to the AMS Servers for validation. The application will display a message indicating the success or failure of the authentication.

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

LI-1: a login page is displayed and asks the user to enter a username and password.
LI -2: the password will be masked as it is typed into the login interface
LI -3: a touchable button submits the username and password from the login form to the AMS Server
LI -4: the credentials are compared against those stored in the AMS database
LI-5: if login information does not match, the user will be given two more chances to enter correct login information
LI -6: after 3 failed login attempts, the user account is locked and an error message is displayed instructing the user to call a 1-800 number to reactivate the account
LI-7: if the database matches the credentials, the server returns the confirmation
LI-8: after confirmation of correct login credentials, the AMS Reservation application menu is displayed

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs.Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

4.2 Search for Flights

4.2.1 Description and Priority

Upon successful entry into the AMS Reservation Application, a user can select "Search for Flights" from the application's main menu. This allows users to search for available flights. This function is one of the main motivators of the application, but it's quite as crucial as the establishment of a secure connection of login ability so it's feature is of medium priority.

4.2.2 Stimulus/Response Sequences

Alter a user selects "Search for Flights" from the AMS Reservation application menu, the search interface opens. The application accepts an entry of the departure city,arrival city, and travel date from the user and submits this request to the AMS Servers. The AMS Server then returns a listing of available fights that meet the user's selection. If no flights are available, the user will be provided with the listing of similar flights that depart or arrive at airports near to the desired destination.

4.2.3 Functional Requirements

- A search form will be provided to the user that requests the city of departure, city of arrival, and travel dates.
- A touchable button submits the data.
- Data is validated for completeness, accurate dates, and valid Gities of departure and arrival.
- If the data is complete and valid, the AMS Reservation application will request maiching fights from the AMS database.
- If matching fights exist, the AMS Reservation application will display matching flights 10 the user.
- If no matching fights exist, an error will be displayed and nearby cities of departure/arrival will be presented to the user.

4.3 Create New Reservation

4.3.1 Description and Priority

After selecting an available flight, users can create a reservation for a flight. Users will then
submit their personal data to secure their seat for the flight. Because this function is a main motivation for the creation of the AMS Reservation Application, but is not as
crucial as the establishment of a secure session or login, its priority is medium.

4.3.2 Stimulus/Response Sequences

After an available flight is selected, the AMS Reservation application displays a reservation

form. Upon submission, the AMS Reservation application submits the payment details to the Pay Processor and all other fields to the AMS Database. For entries

with successful payment, a reservation will be created and confirmation will be provided to the user. For payment failures, a reservation will not be created and an error message will be displayed to the user.

4.3.3 Functional Requirements

CNR-1: display a reservation form to users to enter their name, email address, and billing information
CNR -2: a touchable button submits the data
CNR -3: data is validated for completeness
CNR -4: if data is missing, an error message is displayed
CNR -5: if data is complete, the AMS Reservation application will submit the user's
name and email address to the AMS Database for storage upon selection of the submit button
CNR -6: if data is complete, the AMS Reservation Application will submit payment information to the Pay Processor for processing upon selection of the submit button
CNR -7: the AMS Reservation Application will display a success or failure message
provided by the Pay Processor
CNR -8: if the payment is successful, a confirmation message will display on screen and be sent to the email address provided by the user
CNR -9: if the payment is not successful, an error will display and up to two additional attempts will be permitted
CNR-10: after three failed payment attempts the user will be prompted to call a 1-800 number to complete their reservation

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- At any instant, a maximum of four nodes or users will be given access simultaneously.
- Since the program handles multiple users, if more than one person attempts the same date to the files stored in the database, the program will lock the data file using a 2-phase commit protocol to prevent simultaneous access.

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

- All the data of the passengers must be safely secured in the database system.
- Safety measures must be taken while users do the payment for the tickets online.

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

- It must be ensured that access will be provided to the authorized persons through user ID and password.
- Network security will be provided by the use of firewalls.
- Checks can be performed at regular intervals to ensure data integrity.

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

Usability

Non-technical background of a user should not be an obstacle to being able to understand and use the system.

Robustness

System should be able to display the most recent inquiry by the user in case of refreshment of the page after sudden connection loss.

Consistency

Number of available seats for a specific flight should be decreased by 1 unit once a transaction of the payment for the flight ticket is made.

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

- If the system crashes then the administrator must do the needful to get back the system running.
- Administrators can Update/Add/Delete Flight information.
- Administrators can Update/Add/Delete User Reservations.

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Other Requirements

- The database backend system in use will be Oracle 19c.
- Our development environment will be the latest NetBeans Integrated Development Environment.
- We may make changes to any of the above system requirements at any time and for any reason.

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

EXPERIMENT - 4

Aim: Develop a Risk Mitigation, Monitoring and Management Plan (RMMM) for the project

Objective:

To be able to:

- Identify Risks
- Analyze the risks and prepare the RMMM plan

Resources Needed:

- MS-office / Open office

Theory:

- Planning the Risk Management - The proactive strategy for risk estimation isused which helps in identifying the possible threats that can occur during theproject well in advance. Accordingly, steps to avoid, monitor and manage therisk are to be carried out and noted down in the form of RMMM plan.
- THE RMMM PLAN - A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and is used by theproject manager as part of the overall project plan.
- Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet (RIS). In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may beaccomplished easily.
- Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. Risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives:
  1) to assess whether predicted risks do, in fact, occur
  2) to ensure that risk aversion steps defined for the risk are being properly applied
  3) to collect information that can be used for future risk analysis.
- In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate the origin (which risk(s) caused which problems throughout the project).

Pre-Lab/ Prior Concepts:

- Risk management is the identification, assessment, and prioritization of risks(defined in ISO 31000 as the effect of uncertainty on objectives, whether positive or negative) followed by coordinated and economical application ofresources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities.
- In ideal risk management, a prioritization process is followed whereby the risks with the greatest loss (or impact) and the greatest probability of occurring are handled first, and risks with lower probability of occurrence and lower loss are handled in descending order. In practice the process of assessing overall risk can be difficult, and balancing resources used to mitigate between risks with a high probability of occurrence but lower loss versus a risk with high loss but lower probability of occurrence can often be mishandled.
- Risk Identification and Management Process comprises of following steps:
  1) Risk Identification
  2) Risk Analysis
  3) Risk Assessment
  4) RMMM (Risk Mitigation, Monitoring, Management) plan

Risk Table:

Risk Summary	**Risk Category**	**Probability**	**Impact (1-4)**	**RMMM**
Unauthorized Access	Technical Risk	50%	3	Include a login facility in the application
Incorrect estimation of the scope of the application	Project Risk	70%	4	Ensure that the requirements of the project are clearly understood
Impractical Deadline	Project Risk	40%	3	Plan as many aspects as possible in advance and decide if the project is possible in the given deadline
Reduced team member turnover	Budget Risk	30%	2	Arrange a team of substitute member who can immediately take charge
End user may not be satisfied with the features	Market Risk	60%	4	Develop techniques to get favourable responses by including features that are relevant
Less tickets available	Predictable Risk	40%	2	Show options for travel on the closest day
Sudden Flight cancellation and delays	Unpredictable Risk	80%	5	Include features to give the latest information about the flights

1: Low Impact
2: Marginal
3: Medium Impact
4: Critical
5: Catastrophic

Outcomes:

1. An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.
2. An understanding of best practices, standards and their applications.

Conclusion:

Hence understood the process of risk identification, analysis and preparation of the RMMM plan. Gained the ability to use the techniques, skills, and modern engineering tools necessary for engineering practice and understood the best practices, standards and their applications.

Tushar Nankani
1902112

Software Engineering

EXPERIMENT - 5

AIM: To identify all scenarios and develop use case diagram for the project.

THEORY:

Use case diagram is used to describe the functionalities provided by a system and the users associated with it. It helps to identify the primary elements and process that form the system. The primary elements are termed as actors and processes are called use case. Its main purpose is to help development team visualize the functionalities requirements of a system and to identify and understand relationship between actors and use cases.

Components:

1. Actors - They represent anyone or anything that interacts with the system. They can perform the following:
   (i) input information to the system.
   (ii) only retrieve information from system.
   (iii) both input and retrieve information to and from the system.

   In UML, actors are represented by stickman symbols.

Actor symbol

2. Use cases - They eventually map to the menu option. They represent the functionalities by the system. Each individual functionality provided by system is captured by use case. It is a sequence of transactions performed by a system, that yields measurable result of values for a particular actor.

In UML, use case is represented as an oval.

Use case symbol.

Use case can be related:

(i) Extends relationship - to represent seldom invoked use case or exceptional functionality.

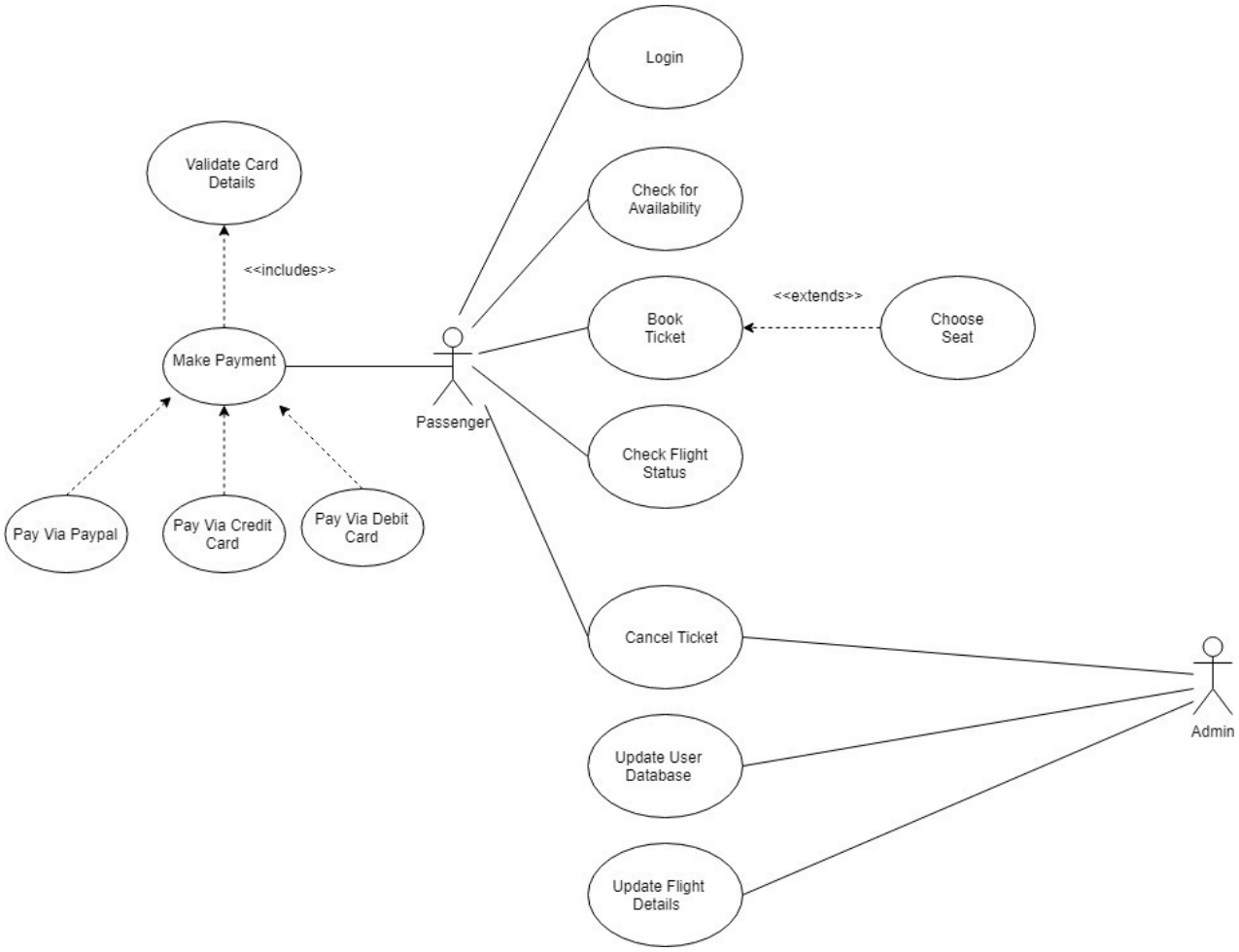
(ii) Includes relationship - to represent functional behaviour common to more that than one use case

(iii) Generalization relationship - To represent when two or more use cases have commonalities in behaviour, structure and purpose.

Use case diagram is an interaction view of some or all actors, use cases and their interactions identified for a system.

Examples are: diagrams showing all use cases for selected actor, all use cases for implementing an interaction and all ~~it~~ its relationships.

Output:



Use-Case diagram for Airport Management System

Tushar Nankani
1902112

Software Engineering

EXPERIMENT - 6

AIM: To design activity and state diagram for the project.

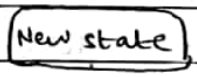
THEORY:

→ Activity diagram: It provides a way to model the workflow of a business process. It can also be used to model code-specific information such as class operation. They are very similar to flowcharts.

- It is a special case of state machine in which part of the state are activities and most of the transitions are implicitly triggered by completion of activities in source activity.
- Activity diagrams are activity centres while state charts are state centres.
- Activity diagrams can model various workflows such as approval of orders, number of transactions and so on.
- Each activity represents performance of group of action in a workflow. Once the activity is complete, flow of control moves to next activity or state through transition.
- If an outgoing transition is not clearly defined, then it is triggered by completion of contained actions.
- Activity diagram can be placed in the logical view but cannot reside within the component view.
- A unique activity diagram feature is a swim lane that defines who or what is responsible for carrying out the activity or state.

→ State chart diagram: It models the dynamic behaviour of individual classes or any kind of object. They show sequence of states that an object goes through, the events that causes a transition from one state to another and actions that result from state changes. Each state represents a named condition during the life of an object during which it satisfies some condition or wait for some event. A state chart diagram contains 1 start state and multiple end states. Transitions connect various states on the diagram decisions, synchronization and activities may also appear on state chart diagrams, similar to activity diagrams.

1. State: This icon appears as a rectangle with rounded corners and a name. It also contains a component for actions.



Name of states should be unique to its enclosing class or if nested within the state. All state icons with same name in a given diagram represent the same state.

2. Action: Actions on state occurs at one of 4 times: entry, exit, do or event. An 'on-event' action is similar to state transition labels with syntax as event (args) [condition]: action.

3. Start state: It shows the beginning of workflow on an activity diagram or beginning of execution of state machine on start chart diagram there can be only 1 - start chart.
When modelling nested states or nested activities, one new start state can be created in each context.
Normally only one outgoing transition can be placed from start state. However, multiple transitions may be placed on a start state if atleast one of them is labelled with a condition. Incoming transitions are not allowed.
Start state icon is small, filled circle that may contain a name.

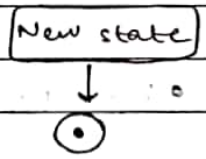
• Begin process

4. End state: It represents final or terminal state on an activity diagram or state chart diagram. It is placed to show end of a workflow an activity diagram or end of state chart diagram.
- Transitions can occur only into end state.
- There can be any number of end state.
- can label end state as desired.
- End state icon is filled circle inside slightly larger unfilled circle that may contain name.

⊙ end process

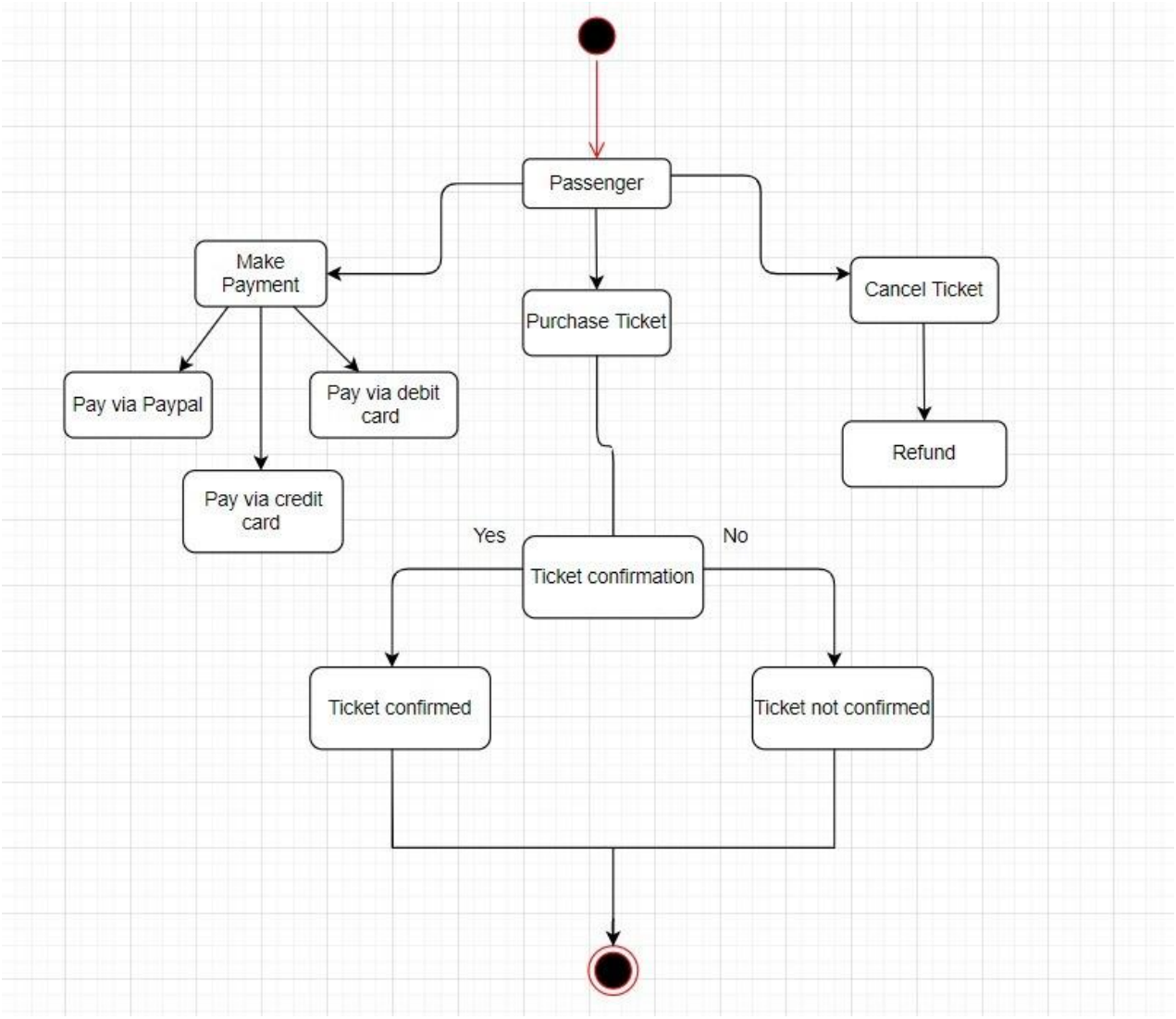
5. State transitions: It indicates that object in source state will perform certain actions and enter destinations state when specified event occurs or when certain conditions are satisfied. It is a relationship between two states, activities or between an activity and state. Transitions originating from state cannot have same event unless conditions are present on the event.

The icon for state transition is a line with an arrowhead pointing towards destination state or activity.

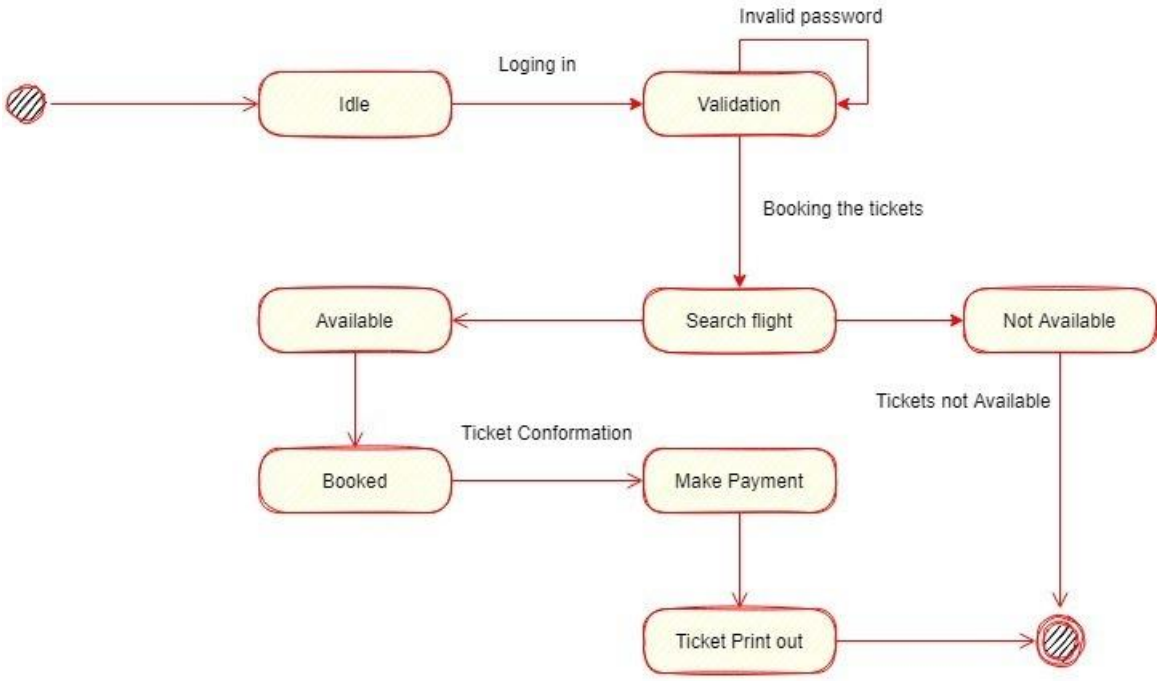


6. Nested states: States may be nested to any depth level. Enclosing states are refered to as super states and everything that lies within the bounds of super state is refered to as its contents. Nested states are called sub states.

Output:



Activity Diagram for Airport Management System



State Chart Diagram for Airline Ticket Booking

Tushar Nankani
1902112

Software Engineering

EXPERIMENT - 7

AIM : To develop data flow diagram for the project for level 0 and level 1 using proper tools.

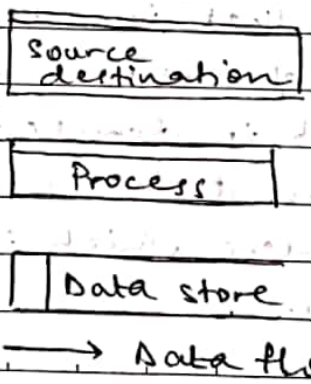
THEORY :

- A data flow diagram (DFD) is a graphical representation of flow of data through a computer system.
- It concerns things like where data will come from and go to as well as where it will be stored and it has nothing to do with processing timing.
- It is not a flowchart and it should not include control elements.
- They are classified as either logical or physical.

1) A logical DFD focuses on the business and how the business operates. It describes the business event that takes place and the data required and produced by each event.

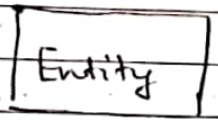
2) A physical DFD shows how the system will be implemented.

- It is composed of four basic symbols:

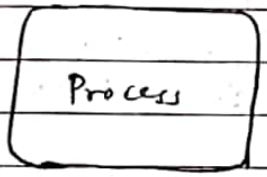


→ Elements of a DFD:

1) External entity:
- It represents sources of data to the system or destination of data from the system.
- They are people, things, organizations etc.
- They are indicated by sharp cornered rectangles or boxes in a DFD.



2) Process:
- The tasks performed on the data are called processes.
- They are represented by rounded cornered rectangles.



3) Data store:
- It represents data that is not moving.
- It is a repository of data and data can be written into it which is depicted by an incoming arrow.
- Two data stores cannot be connected by a data flow. Read operation can be done which is represented by an outgoing arrow.

- External entity cannot read or write into a data store.
- It is represented by an open sided rectangle.

File Name

4) Data flow:
- It represents the movement of data.
- It is indicated by an arrow symbol.

Data →

→ Levels of DFD:

1) Level 0 DFD:
- Also known as context level DFD, it is the simplest DFD.
- It is concerned with how the system interacts with the outside world.
- It basically represents the input and output of the entire system.

Steps:
- Identify the main system.
- Identify the external people who interact with the system.
- Decide what data entities will enter into the system.
- Determine the output.

2) Level 1 DFD:

- The basic module of the system is represented in this phase and how data moves through the modules is shown.
- It produces a high level view of the system that identifies the major processes and data stores.

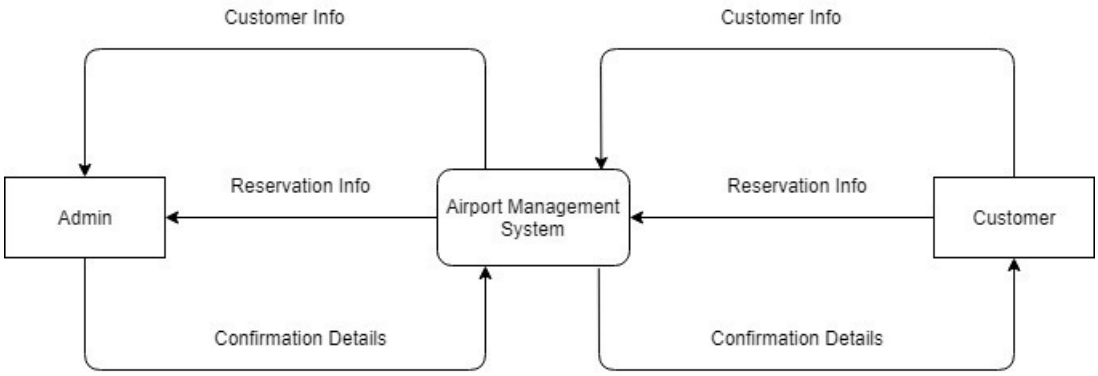
Steps:

- Focus on the process and break it into two or more sub-processes.
- Identify what data flows between these processes and between the entities.
- Note that no new ~~tot~~ entity can be introduced.

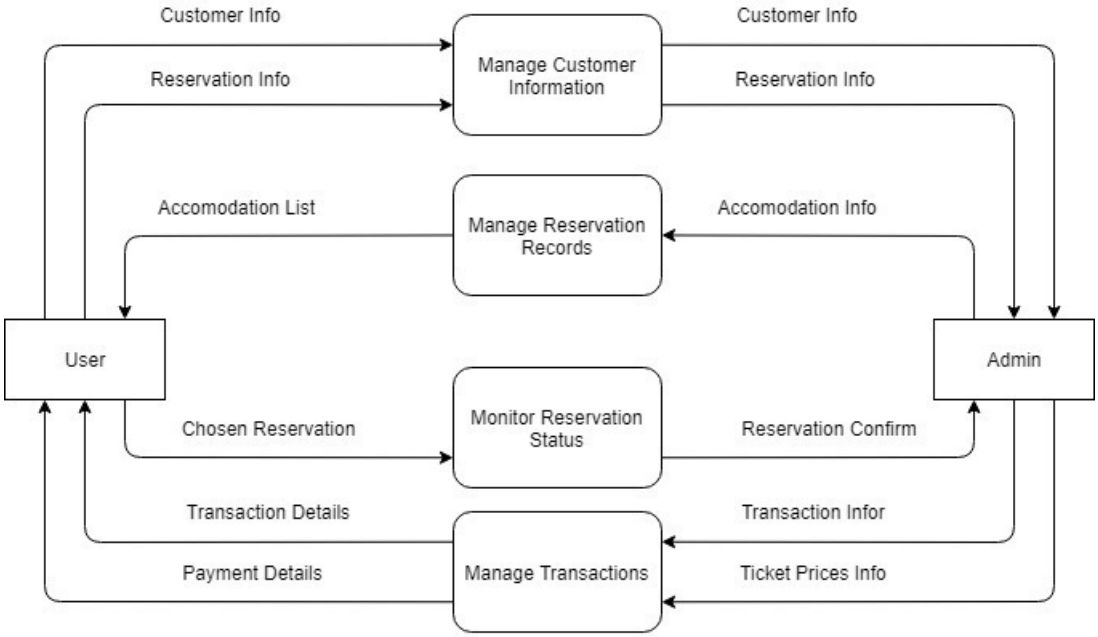
3) Level 2 DFD:

- It can be used to plan or record the specific/necessary detail about the systems functioning.

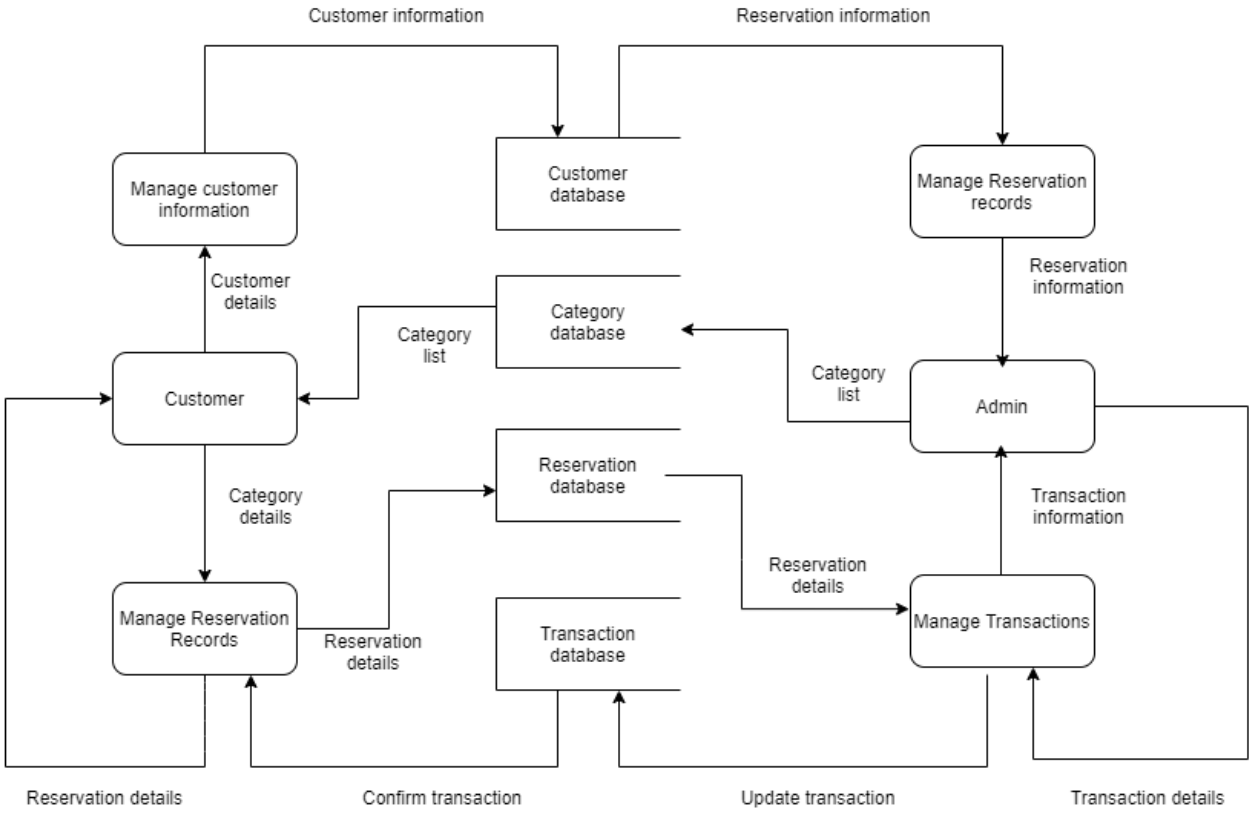
Output:



DATA FLOW DIAGRAM LEVEL 0



DATA FLOW DIAGRAM LEVEL 1



DATA FLOW DIAGRAM LEVEL 2

Tushar Nankani
1902112

Software Engineering

EXPERIMENT-8

AIM: Create a project schedule using Gantt chart

THEORY:

Work Breakdown Structure (WBS):

- Logical decomposition of work to be performed.
- Focuses on how the product, service or result is naturally subdivided
- Outline of what work is to be performed.
- WBS should be deliverable oriented.
- Ensure WBS allows for delivery of all project deliverables as defined in project scope.
- Developing WBS should involve people who will be doing the work.

Example:

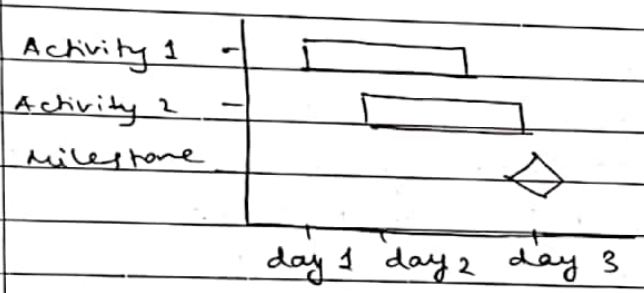
6.2 Testing

6.2.1	Unit testing	2 days
6.2.2	Integration testing	3 days
6.2.3	validation testing	2 days
6.2.4	System testing	4 days
6.2.5	Milestone testing done	0 days

→ Gantt Chart

- Visual scheduling tool
- Graphical representation of information in WBS.
- shows dependancies between task, personnel, other resource allocations

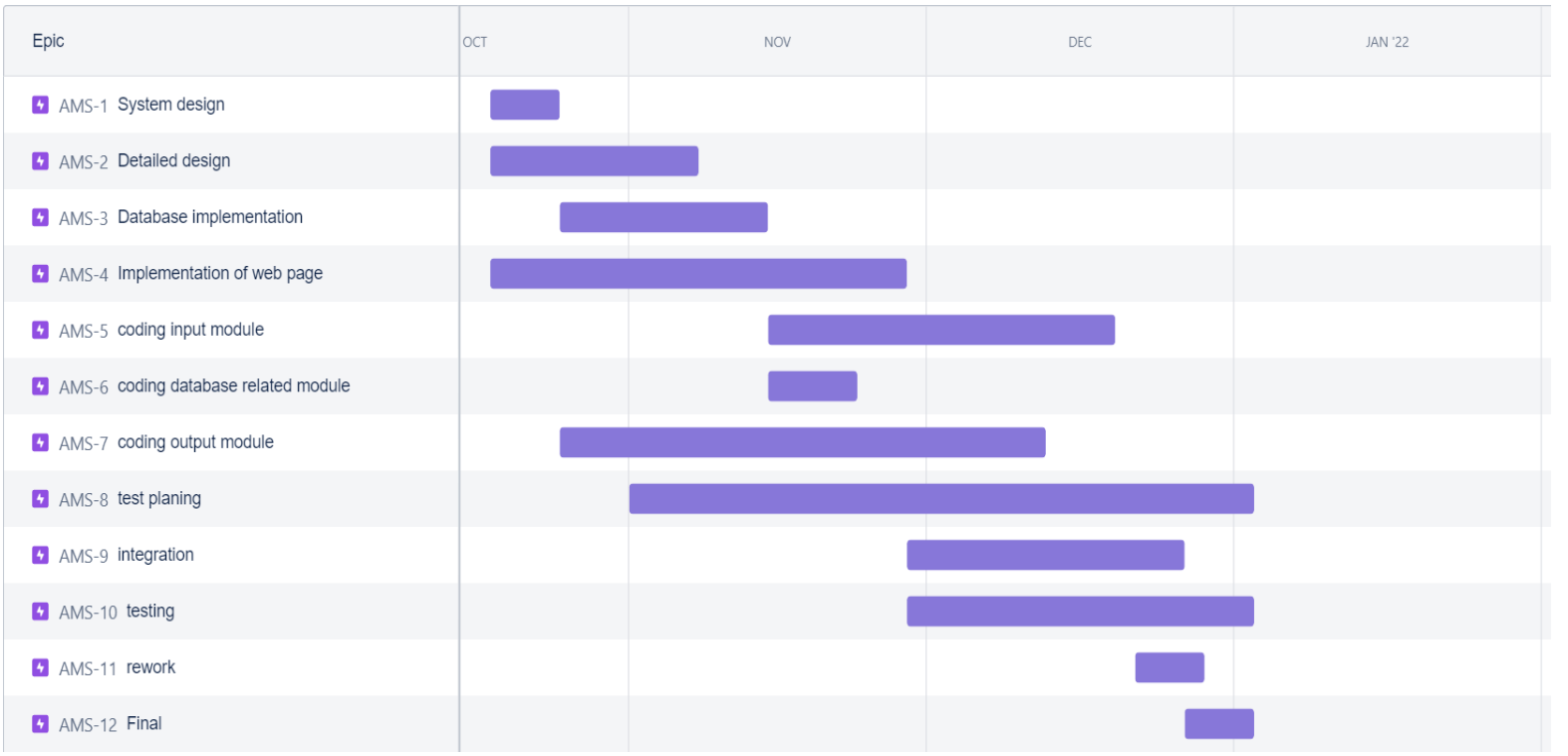
- Track progress towards completion
- List of all task, milestones from WBS along vertical axis.
- List time frame along horizontal axis.
- Activities : create box as length of each activity time duration
- Milestone : create diamond on day the milestone is scheduled to be completed.



→ Timeline chart:

- All project tasks listed in far left column.
- Next few columns may list following for each task : projected start date, stop date, duration, actual start/stop date, duration, task interdependancies
- To the far right are columns representing dates on calender
- Length of horizontal bar on calendar indicates duration of task.
- When multiple bars occur at the same time interval on the calendar, it implies concurrency.

Output:


Epic
OCT
NOV
DEC
JAN '22
AMS-1 System design
AMS-2 Detailed design
AMS-3 Database implementation
AMS-4 Implementation of web page
AMS-5 coding input module
AMS-6 coding database related module
AMS-7 coding output module
AMS-8 test planing
AMS-9 integration
AMS-10 testing
AMS-11 rework
AMS-12 Final

Software Engineering

EXPERIMENT - 9

AIM: To conduct function point analysis for the project.

THEORY:

Cost estimation simply means a technique that is used to find out the cost estimates. The cost estimate is the financial spend that is done on the efforts to develop and test software.

Cost estimation models are some mathematical algorithms or parametric equations that are used to estimate the cost of a product or a project.

1. Empirical Estimation technique: It is usually based on some guesses, based on previously collected data, prior experience with the development of similar types of projects and assumptions.

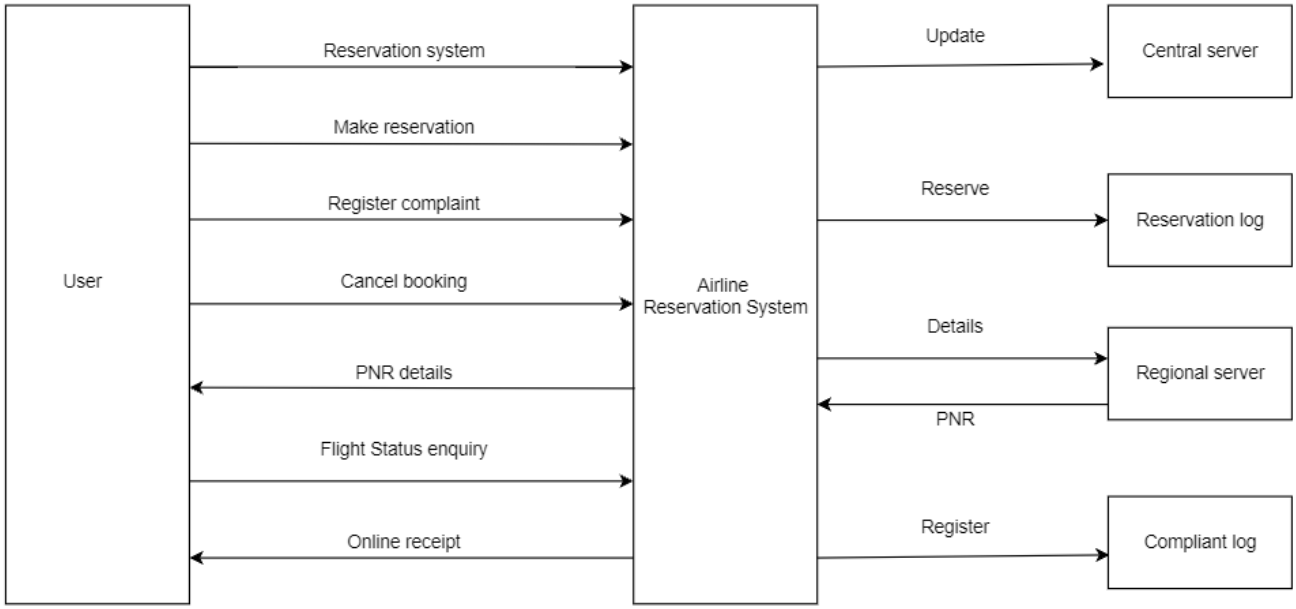
2. Heuristic Technique: The relationship among different project parameters is expressed using mathematical equations in this technique. COCOMO is such a heuristic technique.

3. Analytical Estimation Technique: It is used to measure work. The results are derived by making certain basic assumptions about the project.

Output:

Step-1

Consider an Airline Reservation System:



Here,

- No. of user inputs 3
- No. of user outputs 2
- No. of user inquiries 2
- No. of files 2
- No. of external files 2

- For an Airline Reservation System, the Weighting Factor is assumed to be **Average**
-

Information domain Value	**Count**		**Weighting Factor**				**Total**
			Simple	**Average**	**Complex**		
No. of user inputs	**3**	*	3	**4**	6	=	**12**
No. of user outputs	**2**	*	4	**5**	7	=	**10**
No. of user inquiries	**2**	*	3	**4**	6	=	**8**
No. of files	**2**	*	7	**10**	15	=	**20**
No. of external files	**2**	*	5	**7**	10	=	**14**
					Count total -		**64**

Step -2

Assign values to the 14 questions on a scale of 0 to 5.

- 1. Backup & Recovery =5
- 2. Data communications=4
- 3. Distributed Processing=4
- 4. Performance Critical=4
- 5. Existing Operating Environment=3
- 6. Online Data entry=3
- 7. Input transaction over multiple screens=4
- 8. Master Files updated Online=3
- 9. Information domain values Complex=2
- 10. Internal Processing Complex=3
- 11. Code designed for reuse=4
- 12. Conversion/Installation in design=5
- 13. Multiple Installations=3
- 14. Application designed for change=3

Total value adjustment factor = Σ Fi = 50

Step-3

FP=count total * [0.65 + 0.01 * Σfi]
=64 * [0.65 + 0.01 * 50]
=64 * [0.65 + 0.5]
=64 * 1.15
=73.6
=74

EXPERIMENT - 10

Aim:

Application of COCOMO model for cost estimation of the project

Tushar Nankani
1902112

Software Engineering

EXPERIMENT - 10

AIM: Application of COCOMO model for cost estimation of the project.

THEORY:

COCOMO (Constructive Cost Model) is a regression model based on LOC, i.e. number of lines of code. It is a procedural cost estimate model, often used as a process of reliably predicting the various parameters associated while making a project such as size, effort, cost, time and quality.

It evolved into a more comprehensive estimation model called COCOMO II which had 3 different sizing options:

i) Function Point
ii) Lines of source code.
iii) Object Points

COCOMO II composition model uses object points.

Estimation model using FP, KLOC are also available as part of COCOMO II.

Object point is indirect software measure computed using counts of number of

i) screen
ii) reports
iii) components required to build application.

Each object instance is classified into one of 3 components level, i.e; Simple, medium or difficult.

→ Types of models:

cocomo consists of a hierarchy of three increasingly detailed and accurate forms.

1. Basic cocomo model: It can be used for quick and slightly rough calculations of software costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

$E = a(KLOC)^b$

$time = c(Effort)^d$

Person required = Effort / time

2. Intermediate model:

Cost Drivers are taken into account which makes the model more accurate.

(i) Product attributes:
- Required software reliability extent
- Size of application database
- Complexity of the product

(ii) Hardware attributes:
- Run-time performance constraints
- Memory constraints
- Required turnabout time

(iii) Project attributes:
- Use of software tools
- Application of methods

(iv) Personel attributes:
- Analyst / software engineering capability
- Virtual machine / application experience

$E = (a(KLOC)^b) * EAF$

3. Detailed COCOMO : It incorporates all characteristics of the intermediate version with an assessment of the cost drivers impact on each step of the process. The whole software is divided into different modules and then we apply COCOMO individually and then the effort is summed up.

The six phases are :

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost constructive model.

Problem Statement: Airport Management System

Object Count and Classifying Complexity Level:

Number of Screens	1. Login Screen 2. Create Account Screen 3. Transaction Screen 4. OTP verification Screen	Simple Medium Medium Medium
Number of Reports	1. Balance Report 2. Transaction Report	Medium Medium
Number of 3GL Components	1. JavaScript 2. Python 3. Java	Difficult Difficult Difficult

Complexity Weights and Object Point:

Object Type	Complexity Weights			Given Values			Total
	Simple	Medium	Difficult	Simple	Medium	Difficult	
Screens	1	2	3	1	3	0	7
Reports	2	5	8	0	2	0	10
3GL Components	-	-	10	0	0	3	30
Object Point							47

Object Points = 47

New Object Point (NOP):

Reusability = 20% (assumed)

NOP = Object Point * [(100 – %reuse)/100]

= 47 * [(100 – 20)/100]

= 47 * (80)/100

= 37.6

New Object Point = 37.6

Productivity Rate (PROD):

Developer experience – low

Environment Maturity – low

PROD = (low + low)/2

= (7 + 7)/2

= 7

Productivity Rate = 7

Efforts:

Efforts = NOP / PROD

= 37.6 / 7

= 5.37 person months

Efforts for project development is 5.37 person months.

Size Estimation:

Total FP = 55

3GL components 1FP = 40 LOC

Estimated size = 55 * 40

= 2080 LOC

= 2.08 KLOC

Estimated project size is 2.08 KLOC

EXPERIMENT - 11

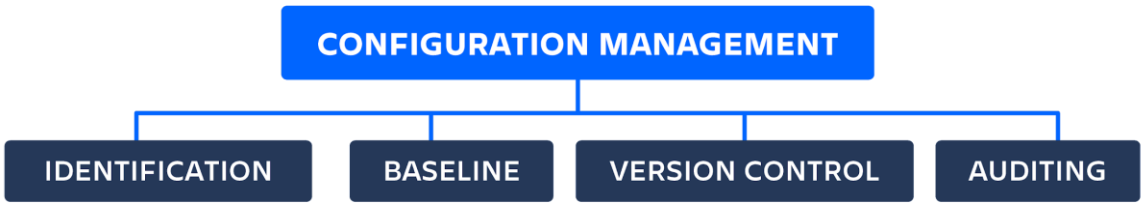
Aim: To do a case study on GitHub for version control.

Theory:

Configuration Management:

Configuration management is a systems engineering process for establishing consistency of a product's attributes throughout its life. In the technology world, configuration management is an IT management process that tracks individual configuration items of an IT system. IT systems are composed of IT assets that vary in granularity. An IT asset may represent a piece of software,or a server, or a cluster of servers. The following focuses on configuration management as it directly applies to IT software assets and software asset CI/CD.

Software configuration management is a systems engineering process that tracks and monitors changes to a software systems configuration metadata. In software development, configuration management is commonly used alongside version control and CI/CD infrastructure.



Configuration management helps engineering teams build robust and stable systems through the use of tools that automatically manage and monitor updates to configuration data. Complex software systems are composed of components that differ in granularity of size and complexity. For a more concrete example consider a microservice architecture. Each service in a microservice architecture uses configuration metadata to register itself and initialize.

Some examples of software configuration metadata are:

- Specifications of computational hardware resource allocations for CPU, RAM, etc.
- Endpoints that specify external connections to other services, databases, or domains
- Secrets like passwords and encryption keys

It's easy for these configuration values to become an afterthought, leading to the configuration to become disorganized and scattered. Imagine numerous post-it notes with passwords and URLs blowing around an office. Configuration management solves this challenge by creating a "sourceof truth" with a central location for configuration.

Git is a fantastic platform for managing configuration data. Moving configuration data into a Git repository enables version control and the repository to act as a source of truth. Version control also solves another configuration problem: unexpected breaking changes. Managing unexpected changes through the use of code review and version control helps to minimize downtime.

Configuration values will often be added, removed, or modified. Without version control this can cause problems. One team member may tweak a hardware allocation value so that the software runs more efficiently on their personal laptop. When the software is later deployed to a production environment, this new configuration may have a suboptimal effect or may break.

Version control and configuration management solve this problem by adding visibility to configuration modifications. When a change is made to configuration data, the version controlsystem tracks it, which allows team members to review an audit trail of modifications.

Configuration version control enables rollback or "undo" functionality to configuration, which helps avoid unexpected breakage. Version control applied to the configuration can be rapidly reverted to a last known stable state.

Version control:

Version control, also known as source control, is the practice of tracking and managing changesto software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter. They are especially useful for DevOps teams since they help them to reduce development time and increase successful deployments.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versionsof the code to help fix the mistake while minimizing disruption to all team members.

For almost all software projects, the source code is like the crown jewels - a precious asset whose value must be protected. For most software teams, the source code is a repository of the invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort. Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences. Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized ina folder structure or "file tree". One developer on the team may be working on a new feature while another developer fixes an unrelated bug by changing code, each developer may maketheir changes in several parts of the file tree.
Version control helps teams solve these kinds of problems, tracking every individual change by each contributor and helping prevent concurrent work from conflicting. Changes made in one part of the software can be incompatible with those made by another developer working at the same time. This problem should be discovered and solved in an orderly manner without blocking the work of the rest of the team. Further, in all software development, any change can introduce new bugs on its own and new software can't be trusted until it's tested. So testing and development proceed together until a new version is ready.

Good version control software supports a developer's preferred workflow without imposing one particular way of working. Ideally it also works on any platform, rather than dictate what operating system or tool chain developers must use. Great version control systems facilitate a smooth and continuous flow of changes to the code rather than the frustrating and clumsy mechanism of file locking - giving the green light to one developer at the expense of blockingthe progress of others.

Software teams that do not use any form of version control often run into problems like not knowing which changes that have been made are available to users or the creation of incompatible changes between two unrelated pieces of work that must then be painstakingly untangled and reworked. If you're a developer who has never used version control you may haveadded versions to your files, perhaps with suffixes like "final" or "latest" and then had to later deal with a new final version. Perhaps you've commented out code blocks because you want to disable certain functionality without deleting the code, fearing that there may be a use for it later.Version control is a way out of these problems.

Version control software is an essential part of the every-day of the modern software team's professional practices. Individual software developers who are accustomed to working with a capable version control system in their teams typically recognize the incredible value version control also gives them even on small solo projects. Once accustomed to the powerful benefits

of version control systems, many developers wouldn't consider working without it even fornon-software projects.

Benefits of version control systems:

Using version control software is a best practice for high performing software and DevOps teams. Version control also helps developers move faster and allows software teams to preserve efficiency and agility as the team scales to include more developers.
Version Control Systems (VCS) have seen great improvements over the past few decades and some are better than others. VCS are sometimes known as SCM (Source Code Management) tools or RCS (Revision Control System). One of the most popular VCS tools in use today is called Git. Git is a Distributed VCS, a category known as DVCS, more on that later. Like manyof the most popular VCS systems available today, Git is free and open source. Regardless of what they are called, or which system is used, the primary benefits you should expect fromversion control are as follows.

1. A complete long-term change history of every file. This means every change made bymany individuals over the years. Changes include the creation and deletion of files aswell as edits to their contents. Different VCS tools differ on how well they handle renaming and moving of files. This history should also include the author, date and
   written notes on the purpose of each change. Having the complete history enables goingback to previous versions to help in root cause analysis for bugs and it is crucial when needing to fix problems in older versions of software. If the software is being actively worked on, almost everything can be considered an "older version" of the software.

2. Branching and merging. Having team members work concurrently is a no-brainer, buteven individuals working on their own can benefit from the ability to work on independent streams of changes. Creating a "branch" in VCS tools keeps multiple streams of work independent from each other while also providing the facility to merge that work back together, enabling developers to verify that the changes on each branch donot conflict. Many software teams adopt a practice of branching for each feature or perhaps branching for each release, or both. There are many different workflows that teams can choose from when they decide how to make use of branching and merging facilities in VCS.

3. Traceability. Being able to trace each change made to the software and connect it to project management and bug tracking software such as Jira, and being able to annotate
   each change with a message describing the purpose and intent of the change can help notonly with root cause analysis and other forensics. Having the annotated history of the
   code at your fingertips when you are reading the code, trying to understand what it is

doing and why it is so designed can enable developers to make correct and harmonious changes that are in accord with the intended long-term design of the system. This can beespecially important for working effectively with legacy code and is crucial in enabling developers to estimate future work with any accuracy.

While it is possible to develop software without using any version control, doing so subjects the project to a huge risk that no professional team would be advised to accept. So the question is not whether to use version control but which version control system to use.

Github commands:

Getting & Creating Projects

Command	Description
git init	Initialize a local Git repository
git clone ssh://git@github.com/[username]/[repository-name].git	Create a local copy of a remote repository

Basic Snapshotting

Command	Description
git status	Check status
git add [file-name.txt]	Add a file to the staging area
git add -A	Add all new and changed files to the staging area
git commit -m "[commit message]"	Commit changes
git rm -r [file-name.txt]	Remove a file (or folder)

Branching & Merging

Command	Description
git branch	List branches (the asterisk denotes the current branch)
git branch -a	List all branches (local and remote)
git branch [branch name]	Create a new branch

Command	Description
git branch -d [branch name]	Delete a branch
git push origin --delete [branch name]	Delete a remote branch
git checkout -b [branch name]	Create a new branch and switch to it
git checkout -b [branch name] origin/[branch name]	Clone a remote branch and switch to it
git branch -m [old branch name] [new branch name]	Rename a local branch
git checkout [branch name]	Switch to a branch
git checkout -	Switch to the branch last checked out
git checkout -- [file-name.txt]	Discard changes to a file
git merge [branch name]	Merge a branch into the active branch
git merge [source branch] [target branch]	Merge a branch into a target branch
git stash	Stash changes in a dirty working directory
git stash clear	Remove all stashed entries

Sharing & Updating Projects

Command	Description
git push origin [branch name]	Push a branch to your remote repository
git push -u origin [branch name]	Push changes to remote repository (and remember the branch)
git push	Push changes to remote repository (remembered branch)
git push origin --delete [branch name]	Delete a remote branch
git pull	Update local repository to the newest commit
git pull origin [branch name]	Pull changes from remote repository

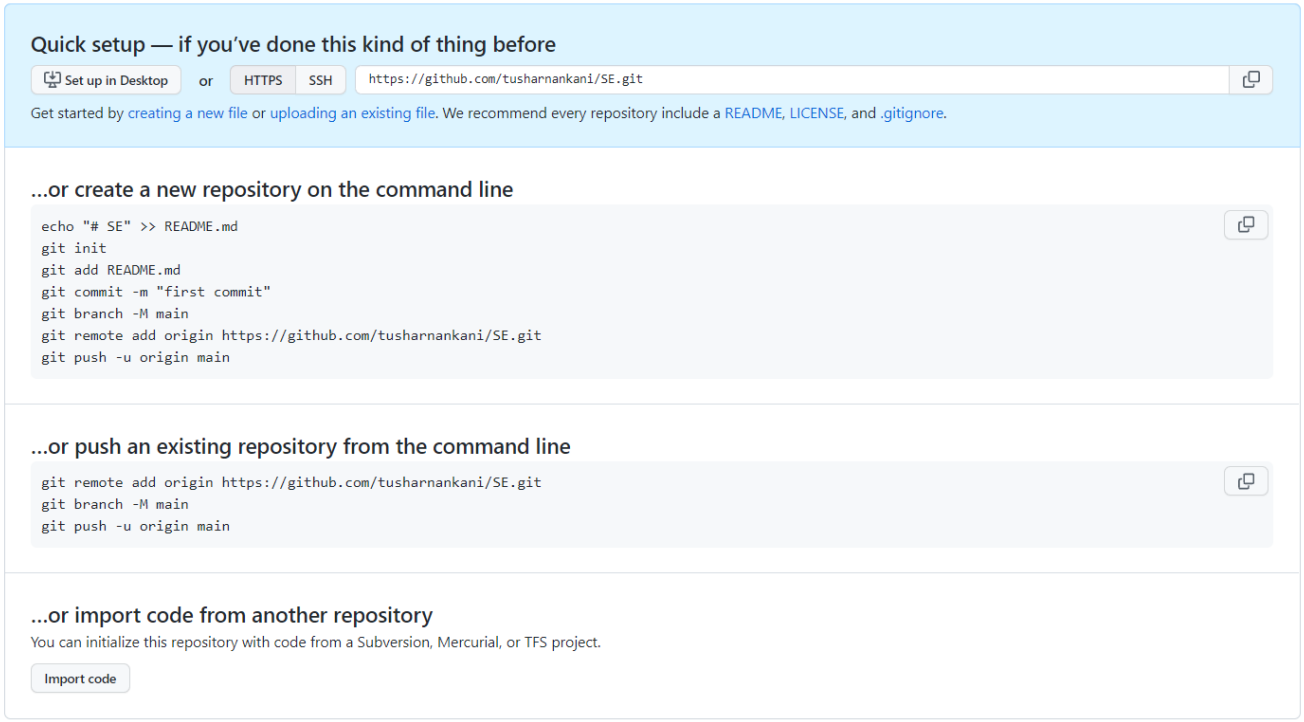
Command	Description
git remote add origin ssh://git@github.com/[username]/[repository-name].git	Add a remote repository
git remote set-url origin ssh://git@github.com/[username]/[repository-name].git	Set a repository's origin branch to SSH

Inspection & Comparison

Command	Description
git log	View changes
git log --summary	View changes (detailed)
git log --oneline	View changes (briefly)
git diff [source branch] [target branch]	Preview changes before merging

OUTPUT:

1. Creating a repository



2. Initial Commit

```
C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE>mkdir SE && cd SE

C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>git init
Initialized empty Git repository in C:/Users/Tushar Nankani/Desktop/COLLEGE/SEM 5/SE/SE/.git/

C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>git add -A

C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>git commit -m "first commit"
[master (root-commit) b11dbfd] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Exp1.docx

C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>git remote add origin https://github.com/tusharnankani/SE.git
```

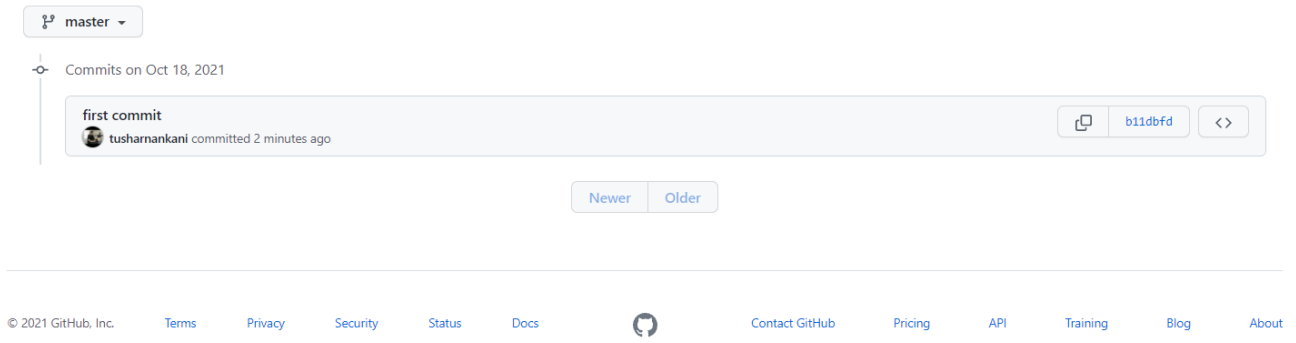
```
C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 10.65 KiB | 10.65 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/tusharnankani/SE.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>
```

3. Commit history

```
C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>git log
commit b11dbfd95e69bf67a119cae1ffd0537d93ff4f2e (HEAD -> master, origin/master)
Author: tusharnankani <tusharnankani3@gmail.com>
Date:   Mon Oct 18 03:01:29 2021 +0530

    first commit
```



4. Git commands

```
C:\Users\Tushar Nankani\Desktop\COLLEGE\SEM 5\SE\SE>git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects
```

Experiment No 12

Aim: Develop test cases for the project using white boxtesting.(Junit)

Theory:

- What is white box testing?

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is testing, structural testing, clear box testing, open box testing and transparent box testing. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its
inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the
White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Using this method, Software Engineer can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once
- Exercise all logical decisions on their true and false sides,
- Execute all loops at their boundaries and within their operational bounds
- Exercise internal data structures to ensure their validity.

- What are test cases?

A test case is exactly what it sounds like: a test scenario
measuring functionality across a set of actions or conditions to verify the expected result. They apply to any software application, can use manual testing or an automated test, and can make use of test case management tools.

A test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

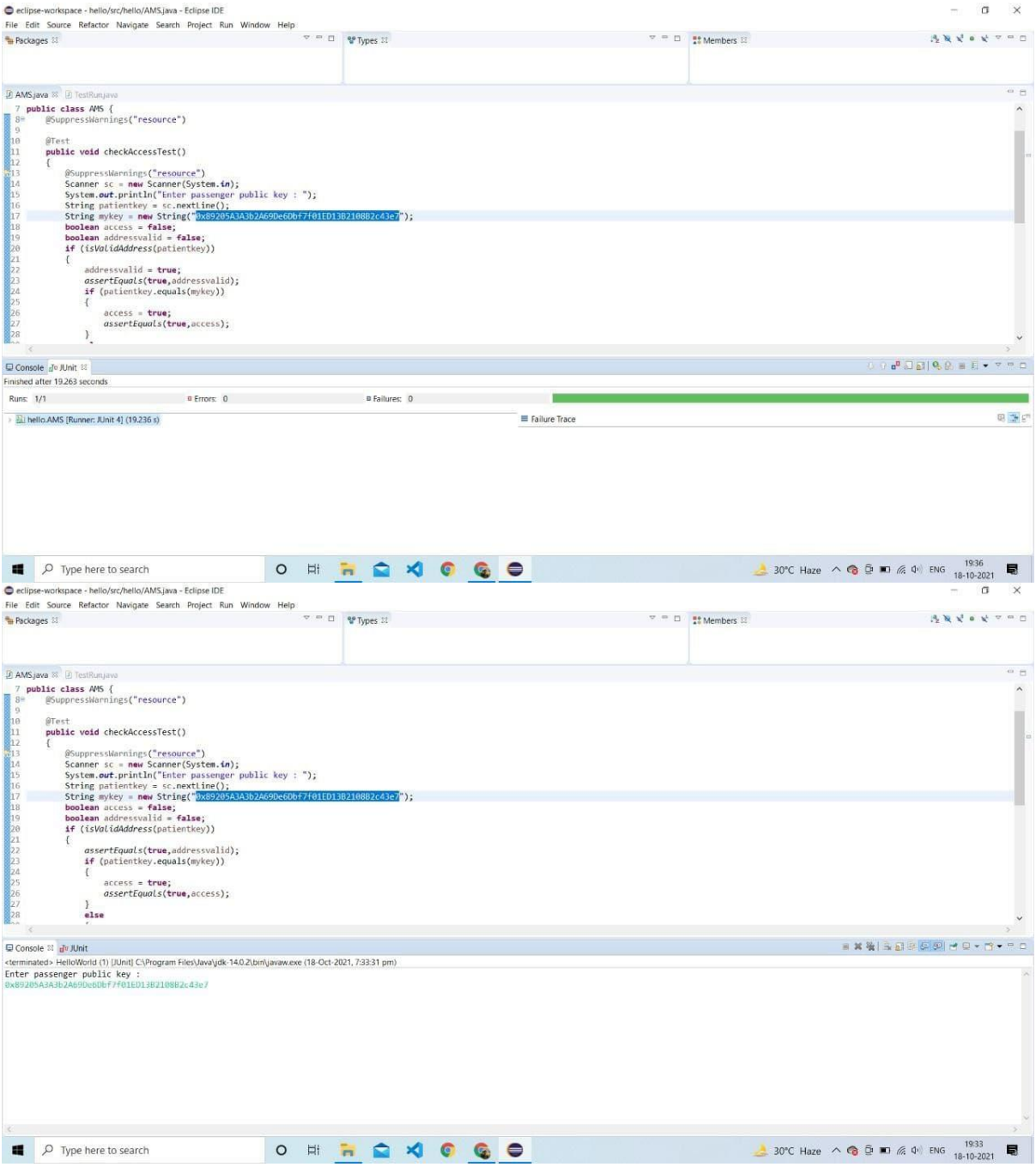
A test case document includes test steps, test data, preconditions and the post conditions that verify requirements.
Test cases define what must be done to test a system, including the steps executed in the system, the input data values that are entered
into the system and the results that are expected throughout test case execution. Using test cases allows developers and testers to discover errors that may have occurred during development or defects that
were missed during ad hoc tests.

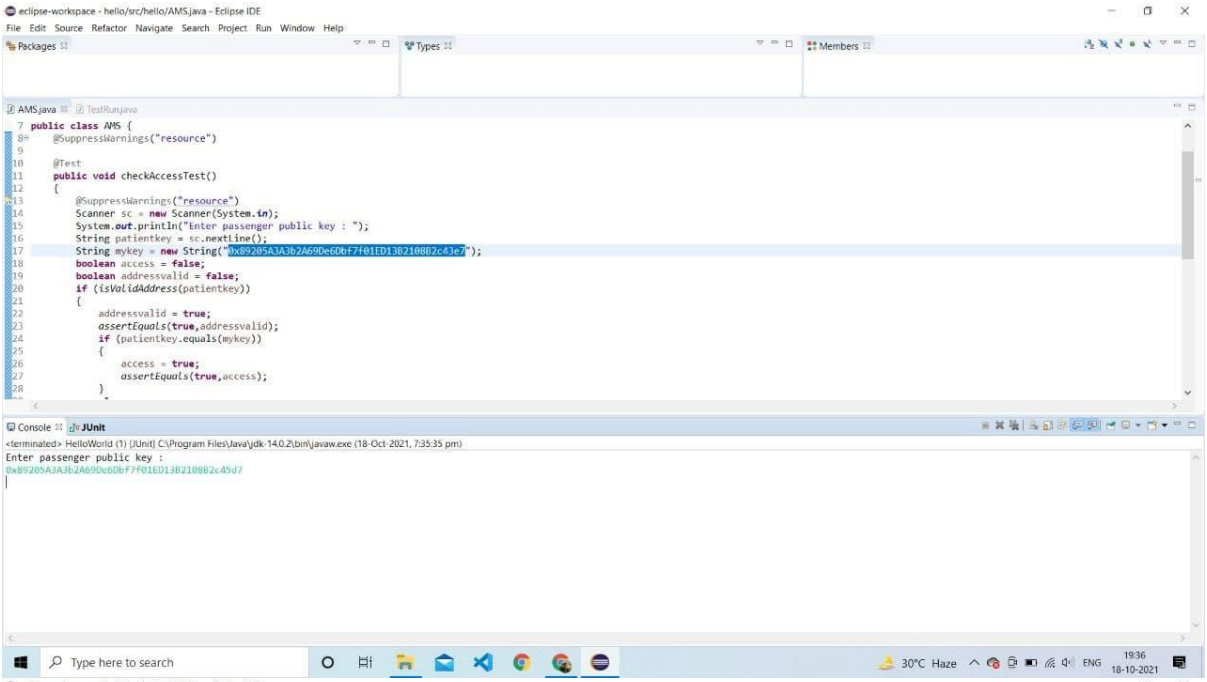
The benefits of an effective test case include:

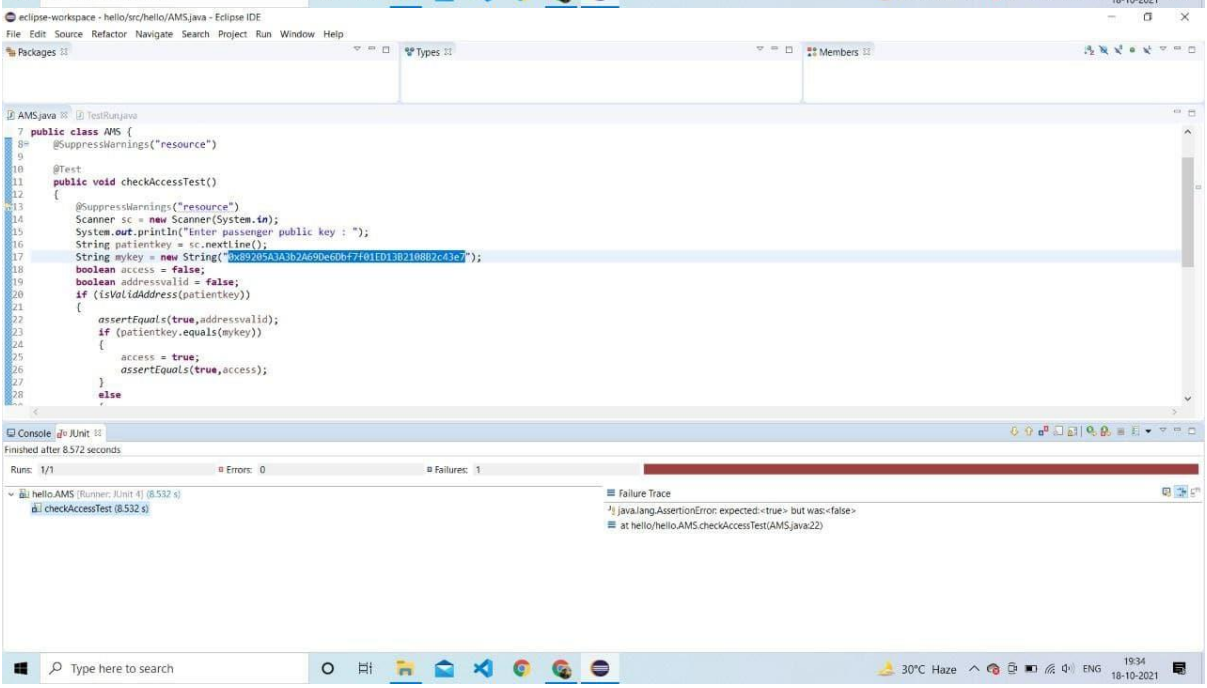
- Guaranteed good test coverage.
- Reduced maintenance and software support costs.
- Reusable test cases.
- Confirmation that the software satisfies end-user requirements.
- Improved quality of software and user experience.
- Higher quality products lead to more satisfied customers.
- More satisfied customers will increase company profits.

Overall, writing and using test cases will lead to business optimization. Clients are more satisfied, customer retention increases, the costs of customer service and fixing products decreases, and more reliable products are produced, which improves the company's reputation and brand image.

Output:




eclipse-workspace - hello/src/hello/AMS.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Packages
Types
Members
AMS.java
TestRun.java
public class AMS {
@SuppressWarnings("resource")
@Test
public void checkAccessTest()
{
@SuppressWarnings("resource")
Scanner sc = new Scanner(System.in);
System.out.println("Enter passenger public key : ");
String patientkey = sc.nextline();
String mykey = new String("0x89205A3A3b2A69De6DbF7f01ED13B2108B2c43e7");
boolean access = false;
boolean addressvalid = false;
if (isValidAddress(patientkey))
{
addressvalid = true;
assertEquals(true,addressvalid);
if (patientkey.equals(mykey))
{
access = true;
assertEquals(true,access);
}
Console
JUnit
<terminated> HelloWorld (1) [JUnit] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (18-Oct-2021, 7:35:35 pm)
Enter passenger public key :
0x89205A3A3b2A69De6DbF7f01ED13B2108B2c45d7
Type here to search
30°C Haze
ENG
19:36
18-10-2021


eclipse-workspace - hello/src/hello/AMS.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Packages
Types
Members
AMS.java
TestRun.java
public class AMS {
@SuppressWarnings("resource")
@Test
public void checkAccessTest()
{
@SuppressWarnings("resource")
Scanner sc = new Scanner(System.in);
System.out.println("Enter passenger public key : ");
String patientkey = sc.nextline();
String mykey = new String("0x89205A3A3b2A69De6DbF7f01ED13B2108B2c43e7");
boolean access = false;
boolean addressvalid = false;
if (isValidAddress(patientkey))
{
assertEquals(true,addressvalid);
if (patientkey.equals(mykey))
{
access = true;
assertEquals(true,access);
}
else
Console
JUnit
Finished after 8.572 seconds
Runs: 1/1
Errors: 0
Failures: 1
hello.AMS (Runner: JUnit 4) (8.532 s)
checkAccessTest (8.532 s)
Failure Trace
java.lang.AssertionError: expected:<true> but was:<false>
at hello/hello.AMS.checkAccessTest(AMS.java:22)
Type here to search
30°C Haze
ENG
19:34
18-10-2021

Tushar Nankani
1902112

SOFTWARE ENGINEERING

ASSIGNMENT-1

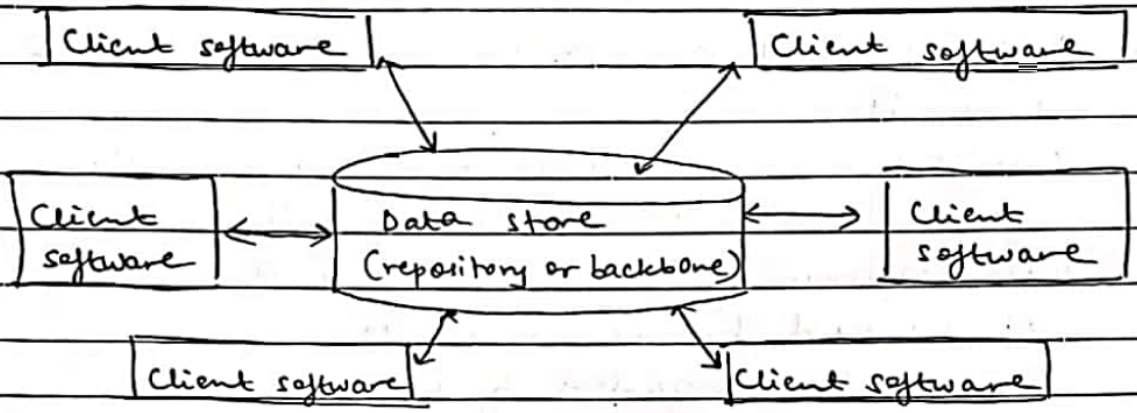
Q. Architectural design - Explain in detail each type with an example.

Ans. Architecture is the structure or organization of program component (modules), the manner in which these components interact and the structure of data that are used by the components.
They can be represented by using one or more of a number of different models:
- structural model
- framework model
- dynamic model
- process models
- functional models

Style describes a system category that encompasses
- A set of components that perform a function required by a system.
- A set of connectors that enable communication, coordinations and cooperations among components.
- Constraints that define how components can be integrated to form the system.
- Semantic models that enables designer to understand the overall properties of a system.

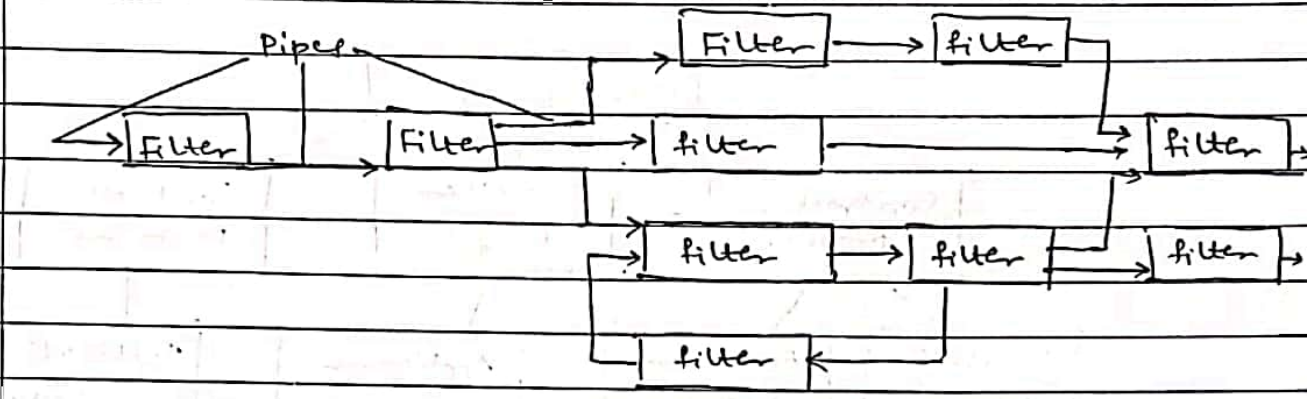
It can be represented by:
- Data centred architecture
- Data flow architecture
- Call and return architecture
- Object oriented architecture
- Layered architecture

1) Data - centered architecture

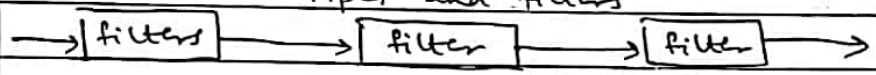


- A data store resides at the center of this architecture and is frequently accessed by other components. to update, add, delete or otherwise modify data within store.
- Client software accesses a central repository which is in a passive state.
- Client software accesses data independent of any changes to the data or the action of other client software.
- A blackboard sends notification to subscribers when data of interest changes and is thus active.
- Data centered architecture promotes integrability.
- Existing components can be changed and new client components can be added to architecture without concern about other clients.
- Data can be passed among clients using the blackboard mechanism. So client components independantly execute processes.

2) Data flow architecture



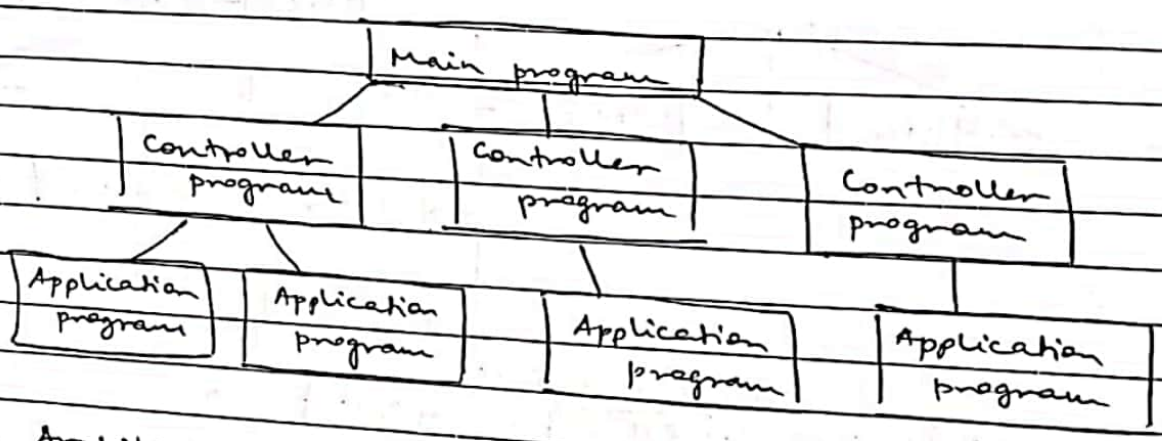
Pipes and filters



Batch sequential

- The architecture is applied when input data are to be transformed though a series of computational or manipulative components into output data.
- A pipe and filter system has a set of components called filters connected by pipes that transmit data from one component to next.
- Each filter works independantly and is designed to expect data input of a certain form and produces data output of a certain form.
- The filter does not require knowledge of the working of its neighbouring filters.
- If the data flow in a degenerate into a single line of transforms, it is termed batch sequential.

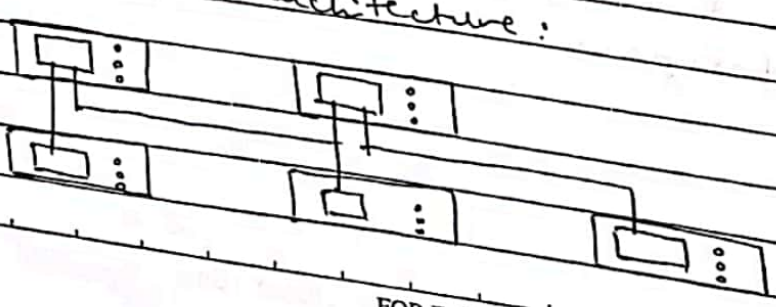
3) Call and return architecture



- Architecture style enables a software engineer to achieve a program style that is easy to modify and scale.
- Two substyles exist within the category :

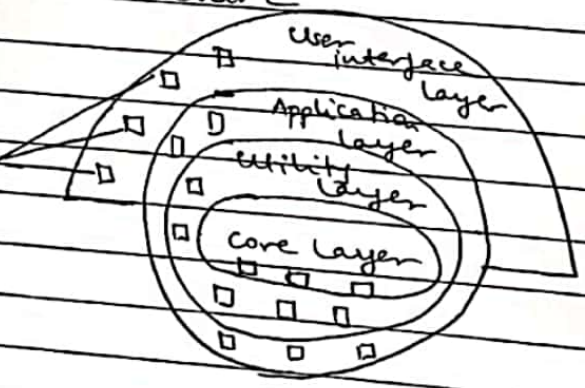
1. main/sub program architecture - Program structure decomposes function into a control heirarchy where a main program invokes a number of program components which in turn may invoke other components.
2. Remote procedure call architecture - The components of a main/sub program architecture are distributed across multiple computers on a network.

4) Object oriented architecture :



- The object oriented paradigm, like the abstract data type paradigm from which it evolved, emphasizes the bundling of data and methods to manipulate and access that data (public interface)
- Components of a system summarize data and the operations that must be applied to manipulate data.
- Communication and coordination between components is accomplished via message passing.

5) Layered architecture



- A number of different layers are defined, each accomplishing operations that progressively become closer to the machine instruction set.
- At the outer layer, components examine user interface operations.
- At the inner layer, they examine operating system interface operations.
- Intermediate layers provide utility services and application software functions.

Tushar Nankani
1902112

SOFTWARE ENGINEERING

ASSIGNMENT - 2

Q. Write a note on Kanban Process Model

Ans. Kanban is a popular framework used by software teams practicing agile software development. It offers advantages to task planning and throughput for teams of all sizes. The benefits are:

A kanban team is only focused on the work thats actively in progress. Once the team completes a work item, they pluck the next item off the top of the backlog. The product owner is free to reprioritize the work in the backlog, without disrupting the team. As long as the product owner keeps the most important work items on top of the backlog, the development team is assured they are delivering maximum value back to the business so there is no need of fixed-length iterations you find in scrum.

Shortened cycle time: Cycle time is the amount of time it takes for a unit of work to travel through the teams workflow from the moment time work starts till it ships. Overlapping skill sets lead to smaller cycle times. When only one person holds a skill set, it bottlenecks the workflow. So shared skills in team members is important and it is the responsibility of the entire team to ensure work is moving smoothly through the process.

- Visual metrics: One of the core values is a strong focus on continually improving team efficiency and effectiveness with every iteration of work. Charts provide visual mechanism for teams. When teams can see data, its easier to spot bottlenecks in the process and can remove them. Control charts and cumulative flow diagrams are used for this.
- Continuous delivery: It is the practice of releasing work to customers frequently - even daily or hourly. It complements kanban because both focus on just-in-time (one-at-a-time) delivery of value.

→ Kanban boards:

The work of all kanban teams revolves around a kanban board; a tool used to visualize work and optimize the flow of work in the team.
A basic kanban board has a three step workflows: To Do, In Progress and Done; however, depending on a team's size, structure and objectives, the workflow can be mapped to meet the unique process of any particular team. The kanban methodology relies upon full transparency of work and real-time communication of capacity therefore the kanban board should be seen as the single source of truth for the teams work.

→ Kanban cards:

For kanban teams, every work item is represented as a separate card on the board. The main purpose of doing this is to allow team members to track the progress of work through its workflow in a highly visual manner. The cards feature critical information about that particular work item, giving the entire team full visibility into who is responsible for that item, a brief description of that job, how long that work is estimated to go on, and so on. They often also feature screenshots and other technical details that is valuable to the assignee.

→ Diagram: