SOFTWARE ENGINEERING LABORATORY

ASSIGNMENT - 1

GROUP - 1

FORMAL DOCUMENTATION OF SOFTWARE DEVELOPMENT LIFE CYCLE

TEAM MEMBERS

2021CSB001
2021CSB004
2021CSB006
2021CSB007

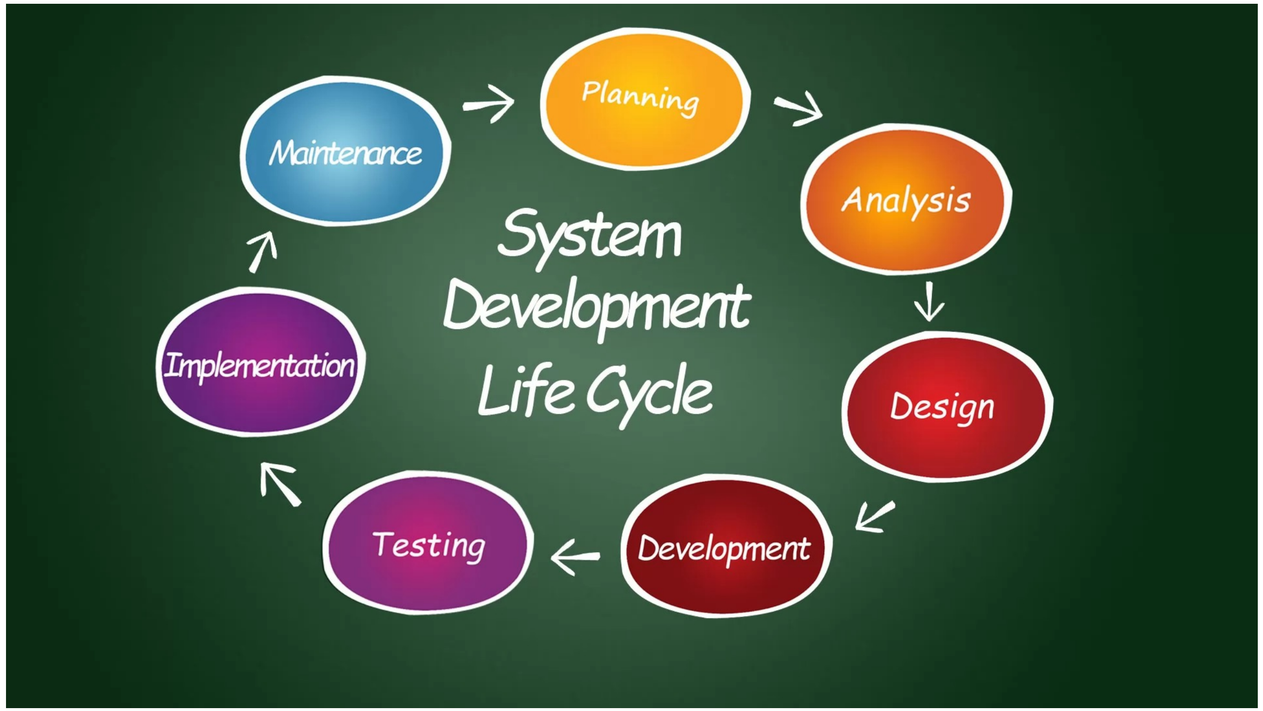
IIEST SHIBPUR

INTRODUCTION

- SDLC stands for **Software Development Life Cycle**
- Main goal is to develop high-quality software at minimum cost and shortest time duration as per requirement.
- Used to structure development process of software product in organised way.

REASONS FOR SDLC

- 1. Ensures quality
- 2. Efficient use of resource
- 3. Increases accountability
- 4. Facilitates communication
- 5. Supports project management



PHASES OF SDLC

- 1. Feasibility Study
- 2. Requirement Analysis and specification
- 3. Design Documents
- 4. Coding or Development
- 5. Testing
- 6. Maintenance

3 CASE STUDIES & OTHER EXAMPLES

- Whilst explaining the following pointers, we shall take the case study of a Home Security System and many other examples..
- After the completion of this segment, there are two more standalone case studies of Social Media Site and Health Application to track Sleep Data.

1. FEASIBILITY STUDY

- **Definition:** A study that uses business and technical information and cost data to determine the economic potential and practicality (i.e., feasibility) of a project.
- **Purpose:**
- Main purpose feasibility study to find these following questions:
  - How difficult will it be to build?
  - How much time available to build?
  - Are there sufficient human resource available for the project?
  - What are the contingencies and mitigation plans if the project takes too long?

1. FEASIBILITY STUDY

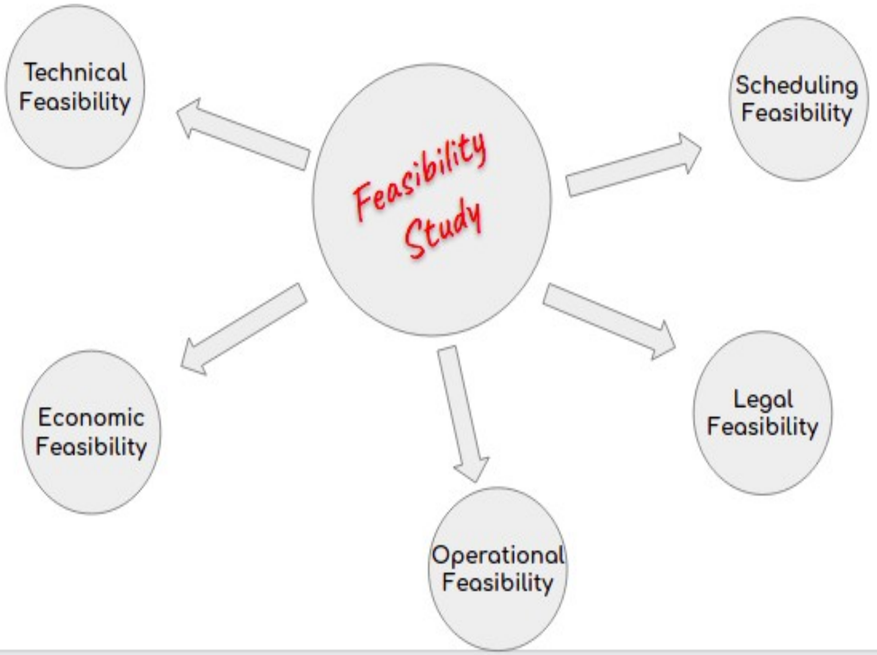
TOOLS USED: Feasibility Report

It includes the following:

- **Topics:**
  - Introduction, Purpose, Objectives, and Scope
  - System Overview and Background
  - System, Plant or Operational Details
  - Current Systems and Processes
  - Current Operations
  - Physical Environment
  - User Organisation
  - Deliverables and End Products
  - Analysis, Solutions and Alternatives
  - Approvals

1. FEASIBILITY STUDY

- **Types of Feasibility:**
  - **Economical Feasibility:** Can we develop the software within the current budget?
  - **Schedule Feasibility:** Can we complete the project within deadline?
  - **Operational Feasibility:** Can we create software according to our customers requirements?
  - **Legal Feasibility:** Does our project meet the cyber law and other regulatory frameworks?
  - **Technical Feasibility:** Can current systems support our software?



1. FEASIBILITY STUDY

- **Outcome of the Feasibility Report:**
  - It summarises the findings and provides recommendations for whether or not proceed with the project. It includes all the aforementioned points.
  - If the study finds that the project is not feasible, the project may be abandoned or put on hold, and the team will look for alternative solutions to the problem.
  - If the study finds that the project is viable, the team can proceed with the next steps in the SDLC.

1. FEASIBILITY STUDY

EXAMPLE

EXAMPLE TABLES IN FEASIBILITY REPORT OF HOME CAMERA SYSTEM

COMPARISON OF VULNERABILITY

HVDA Used	Smart Home			Vuln.			Attacks		Solution
		Router	Apps	SFA	PPBC	Hrd.	Sound	Network	
Amazon Echo	Yes	Yes	No	Yes	No	-	Yes	MITM Packet info Replaying Web App. API	Improved SFA during orders
Amazon Echo	Yes	Yes	Yes	-	-	-	-	DoS	-
Amazon Alexa	Yes	-	Yes	-	-	-	Yes	Respiratory data Skill data set	-
Amazon Echo	Yes	-	-	-	-	Yes	eMMC SD card pinout JTAG	-	-
Amazon Alexa	Yes	-	Yes	-	-	-	-	Skill data set	-
Google Voice Assistant	No	-	Yes	Yes	-	-	Yes	VoicEmployer	-

POWER CONSUMPTION ANALYSIS

Device	Mode	Power usage (watts) in 1 hour	Power in 24 hours	Energy (kwh) Per day	Energy (kwh) Per month
Alexa	Idle	2.8	67.2	0.0672	2.016
	Listening	3.6	86.4	0.0864	2.592
	Playing music (low volume)	2.8	67.2	0.0672	2.016
	Playing music (medium volume)	3.0	72	0.072	2.16
	Playing music (high volume)	7.0	168	0.168	5.04
	Basic reaction	4.0	96	0.096	2.88
Ras-pi		4.0	96	0.096	2.88

https://www.researchgate.net/publication/357628553_Security_Analysis_And_Feasibility_of_Smart_Entrance_System_in_Smart_Home_Applications

2. REQUIREMENT ANALYSIS AND SPECIFICATION

TOOLS USED: System Requirement Specifications (SRS) Document

- **Some examples of popularly used tools:**
- ***EasyRM*** *(Cybernetic Intelligence GmbH)*: it builds a project-specific dictionary / glossary that contains detailed requirements, descriptions and attributes.
- ***Rational RequisitePro*** *(Rational Software)*: Allows to build a requirements database, represent relationships among requirements, and organise, prioritise, and trace requirements.
- ***Volere*** *(UK)*: Contains a template for requirement modelling.

2. REQUIREMENT ANALYSIS AND SPECIFICATION

- **Definition:** Requirements analysis is a process (phase) used to determine the needs and expectations of a new product.
- **Purpose:**
  - Define and analyse the exact requirements of the customer.
  - Document these requirements properly.
  - Determine the changes in the system which will come due to these requirements. This is called “gap analysis”.

2. REQUIREMENT ANALYSIS AND SPECIFICATION

- **Types:**
  - **Requirement gathering and analysis:** activity to collect all relevent information of the software products from the customers.
  - **Requirement Documentation:** Represent requirements in consistent format.
- **Tools:**
  - **Data Flow Diagram:** A data flow diagram / bubble chart is a graphical representation of the flow of data through a system .
  - **Control Flow Diagram:** A graphical representation of how control flow during the execution of program.
  - **ER Diagram**: It is a non technical design method. It is used to establish relations between entities .

COMPONENTS OF AN SRS DOCUMENT

REQUIREMENT ANALYSIS

XML Legal Document Utility	Version: <1.0>
Software Design Document	Date: 2007-04-20
SDD-XLDU	

Revision History

Date	Version	Description	Author
04/18/07	<1.0>	Initial Version of Document	Rex McElrath

Table of Contents

Software Design Document

1 Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behavior models, and other supporting requirement information.

1.1 Purpose

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built.

1.2 Scope

This Software Design Document is for a base level system which will work as a proof of concept for

1.3 Definitions, Acronyms, and Abbreviations

- **Data Objects –** Data objects are Java objects with predefined structures capable of holding data in a structure that is quickly and easily accessible by other parts of the software system. They provide also can help provide a convenient abstraction of the data in a database so that it can be retrieved into a format, such as a denormalized format, that makes access and manipulation of the data easier than if the database had to be called directly. http://java.sun.com/products/jdo/
- **Denormalized** - Normalization of a database is the activity of restructuring the database to avoid data anomalies and inconsistencies by focusing on functional dependencies to help structure the data. A web address to reference about normalization is: http://en.wikipedia.org/wiki/Database_normalization . Denormalization is the act of undoing some of the structural changes made during normalization to help with performance. http://en.wikipedia.org/wiki/Denormalization
- **Digital Signature** – A digital signature is a unique object which is strongly tied to a single entity and the document which signature is intended for. In the same way that a ink on paper signature has characteristics that are unique to a person due to variations in writing a digital signature has characteristics that uniquely tie it to a single person and signing instance. **http://en.wikipedia.org/wiki/Digital_signature**

1.4 References

- XML Legal Documents Utility Software Development Plan
  - Version 1.0, Last Updated on 2007-01-31

1.5 Overview

The Software Design Document is divided into 11 sections with various subsections. The sections of the Software Design Document are:

1 Introduction
2 Glossary
3 Use Cases
4 Design Overview
5 System Object Model
6 Object Descriptions
7 Object Collaborations
8 Data Design
9 Dynamic Model
10 Non-functional Requirements
11 Supplementary Documentation

2 Glossary

2.1 Glossary is unused in current document due to Section 1.3 Definitions, Acronyms, and Abbreviation providing terms and definitions for internal use of the document.

11 Supplementary Documentation

11.1 Tools Used to Create Diagrams

11.1.1 UML Modeling Tools
11.1.1.1 ArgoUML – Version 0.24, http://argouml.tigris.org/
11.1.2 Entity Relationship Diagramming Tools
11.1.2.1 Dia – Version 0.95, http://live.gnome.org/Dia

3 Use Cases

Use-Case Model Survey

3.1 Actors

3.1.1 Document Manager

3.1.1.1 Information: The Document Manager is a user who works with legal documents. This is an abstraction of the specific users as they all perform similar actions, but for different reasons. For example, a court clerk and an attorney both sign documents, bu an attorney does so to state that they created or agree to the documents and the cou clerk does so to state that the document has been received and is now secured with a secure hash to detect modification. The mechanics and the processes used for each are the same to apply their respective digital signatures, but the intent and meaning o each application of a digital signature is different. The specific actors who fall into the broader category of document manager are:

3.1.1.1.1 Judge
3.1.1.1.2 Court Clerk
3.1.1.1.3 Attorney
3.1.1.1.4 Paralegal Professional
3.1.1.1.5 Pro Se Party

10 Non-functional Requirements

10.1 Performance Requirements

- The system should be able to generate previews of documents within 15 seconds of user request.
- The system should be able to be multi-tasking to allow multiple users, up to 40 simultaneous users per interface instance to interact with the system without having to wait on others to fini working with the system.
- The system should be able to hold and search through large amounts of documents. The da structures used for the system will be fairly simple consisting of a few fields to hold document types and their related codes, XML instances with an id, and an audit log table, however the size of the simple data structures could potentially be quite large.
  - Expected capacity for large volume courts – approximately 108, 000 new documents a year with expected retention capacity of 10 years of active documents. After 10 years, documents can be stored in slower to access storage media. Which equates to approximately 1,080,000 documents that will need to be able to be stored and searched.

10.2 Design Constraints

- The software to be built should take advantage of open source libraries and supporting software, such as databases and web containers, unless an adequate open source product is not available or creatable for use.
  - The work will be licensed under an existing open source license, available at, http://license.gaje.us , and donated for use to standards committees that the agency participates in, such as the LegalXML Technical Committee.
- The software should adhere to locally or nationally recognized standards.
  - XML schemas should follow the National Information Exchange Naming and Design Rule http://www.niem.gov/topicIndex.php?topic=file-ndr-0_3 .

When implemented in later versions, document retention schedules should follow the guidelines set forth by the Administrative Office of the Courts in Georgia for records retention http://www.georgiacourts.org/aoc/records.php .

https://www.oasis-open.org/committees/download.php/24846/Example-SoftwareDesignDocument-LegalXMLUtility.pdf

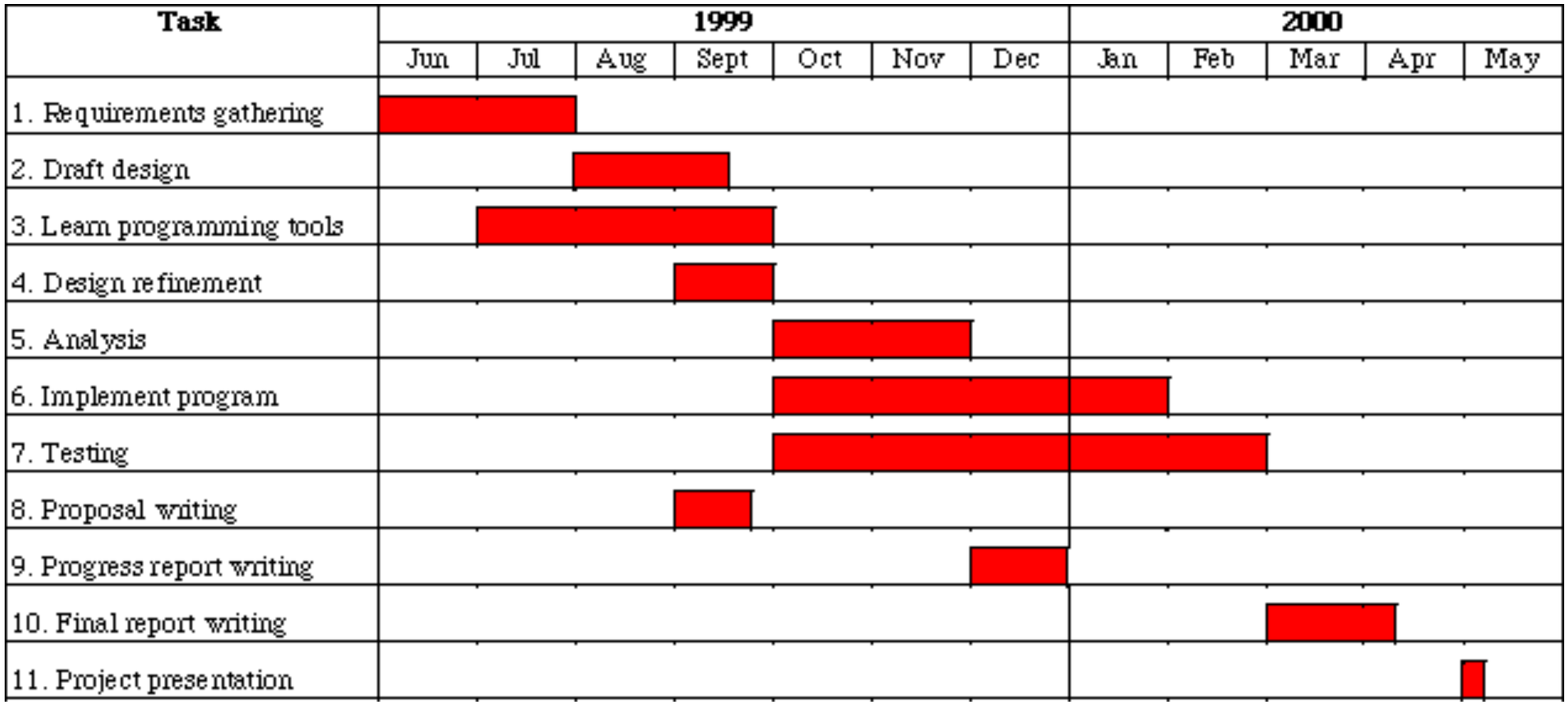
2. REQUIREMENT ANALYSIS AND SPECIFICATION

EXAMPLE

GANTT CHART

- Gantt Charts provide a visual representation of tasks along with their scheduled timelines.

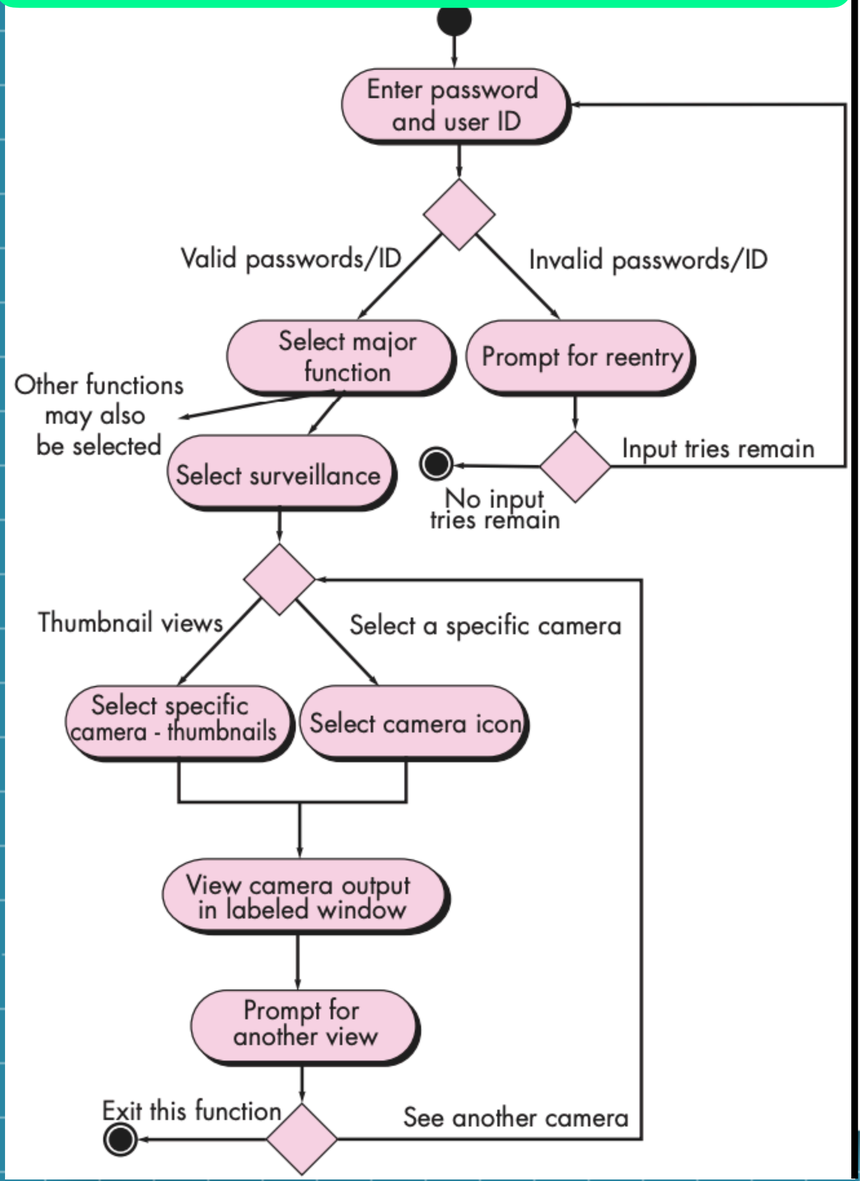
They help business analysts visualise the start and end dates of all the tasks in a project.



2. REQUIREMENT ANALYSIS AND SPECIFICATION

EXAMPLE

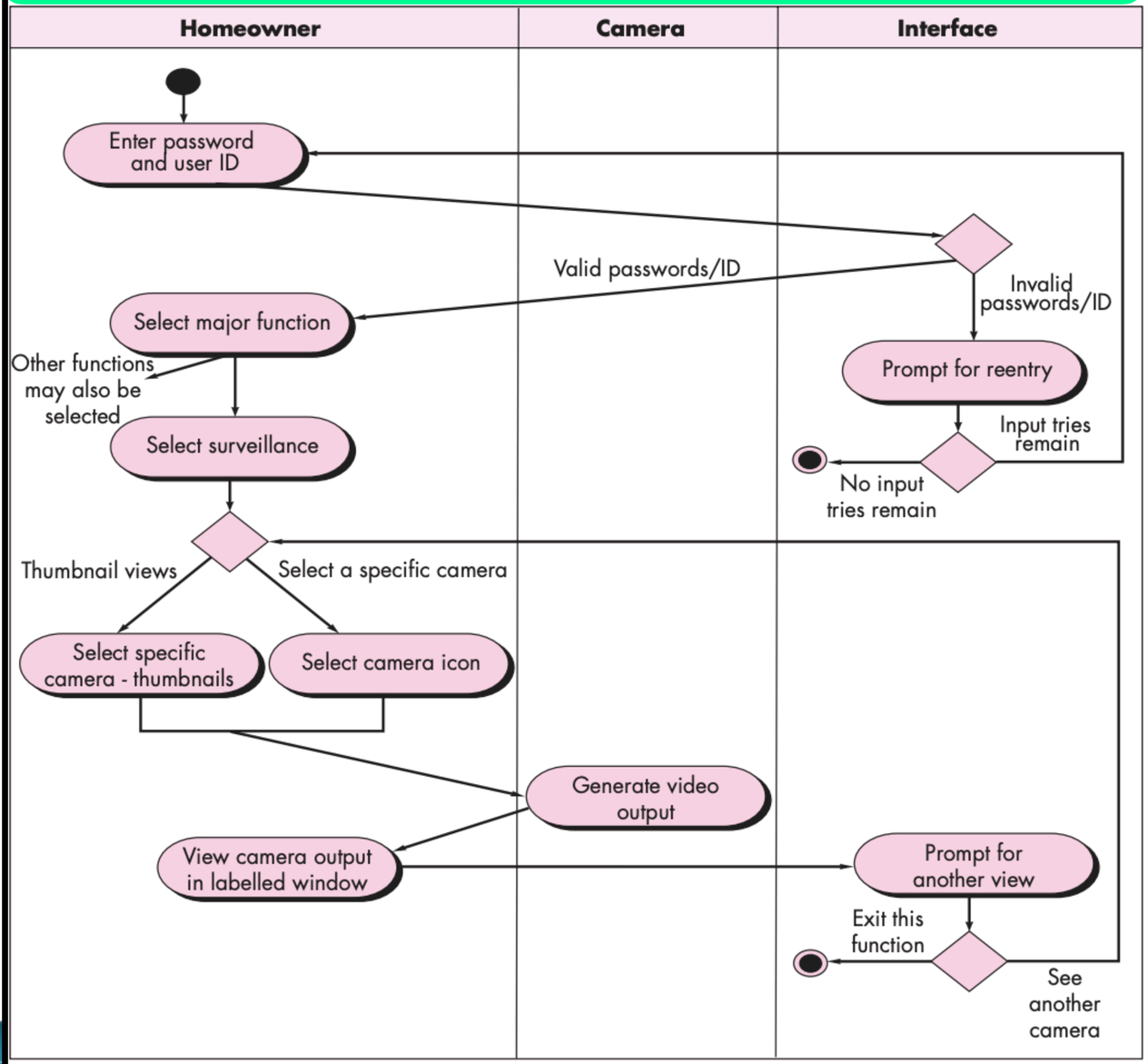
UML MODEL



Both Diagrams:

For Accessing camera surveillance system via the Internet.

SWIMLANE DIAGRAM



3. DESIGN DOCUMENTS

TOOLS USED:

System Design Document

- **Definition:** The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built.
- **Purpose:**
  - To create a blueprint of the project.
  - This blueprint helps the team members for consecutive and systematic work.
  - Create prototype which helps as a bridge between requirements, design and developments.

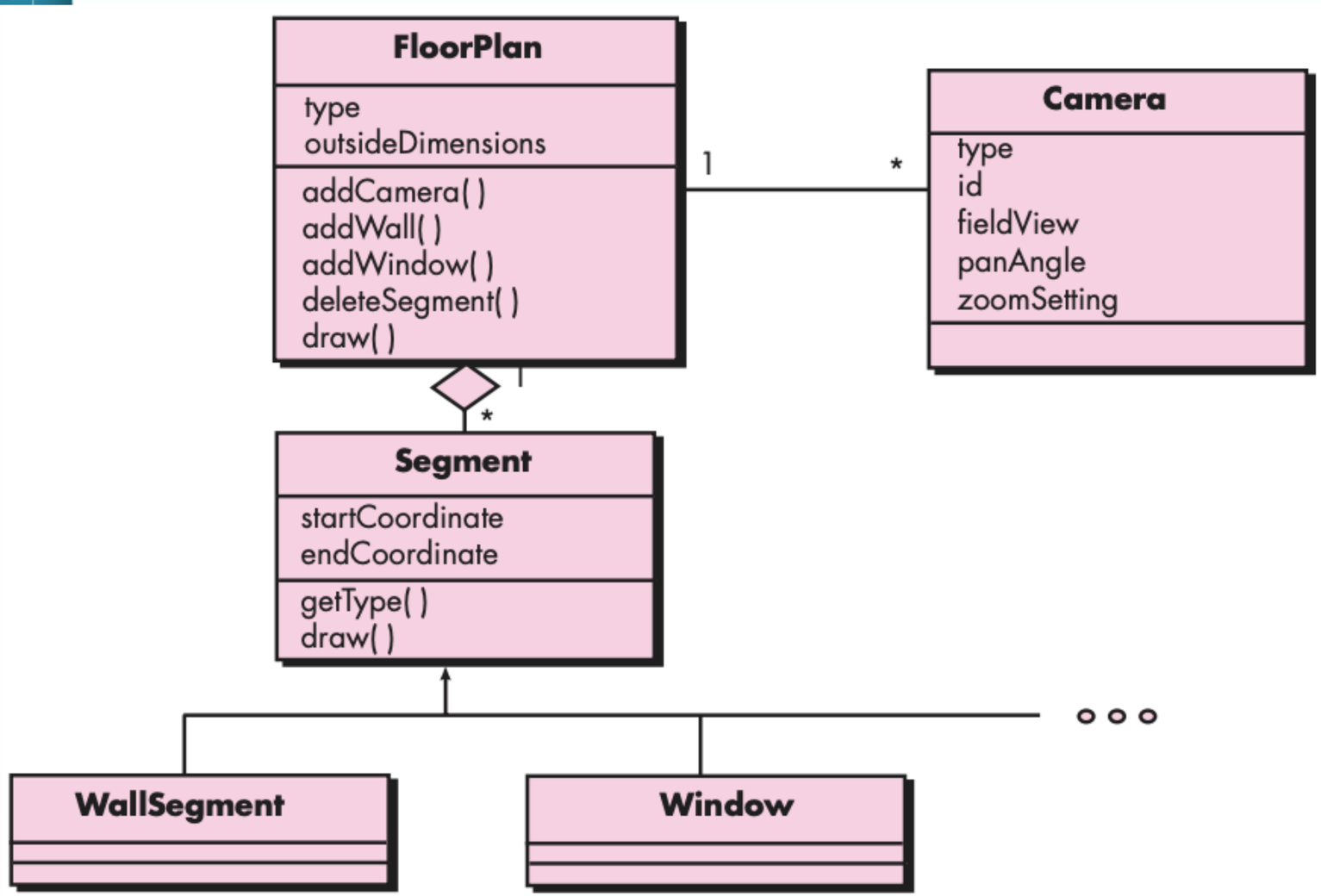
3. DESIGN DOCUMENTS

- **Summary:**
  - **Introduction:** Contains short paragraph on purpose and scope of the project.
  - **Glossary:** Short descriptions on various items used in the document.
  - **Use cases:** Visual representation of the funtionalities to be offred.
  - **Design overview:** Lists all coding related documents in summary.

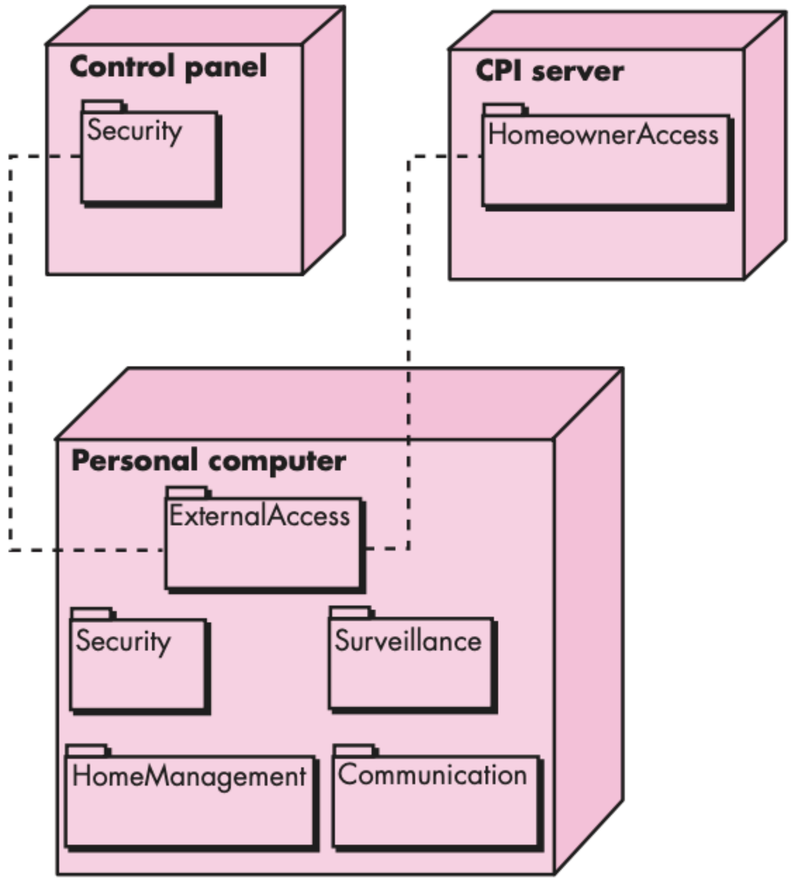
3. DESIGN DOCUMENTS

EXAMPLE

DESIGN CLASS FOR FLOOR PLAN



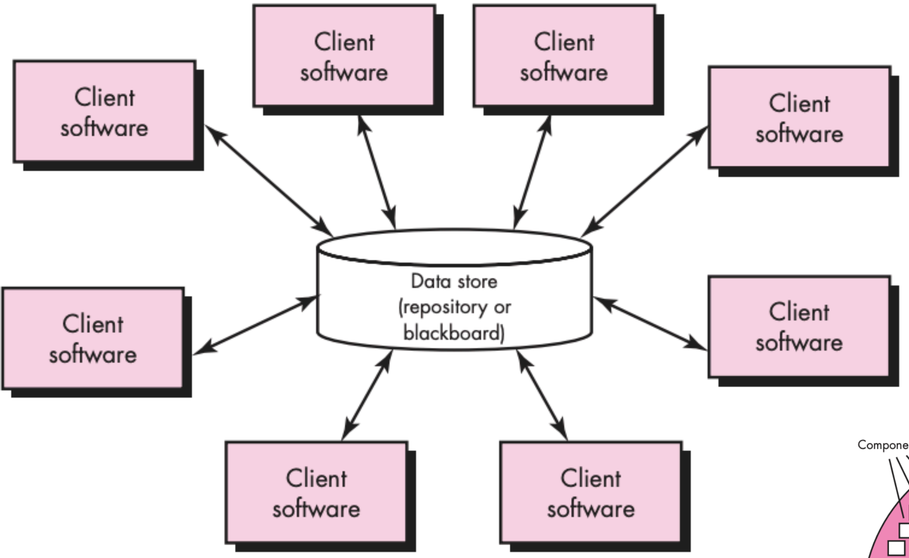
UML MODEL FOR DEPLOYMENT

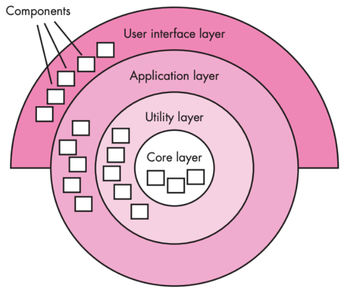


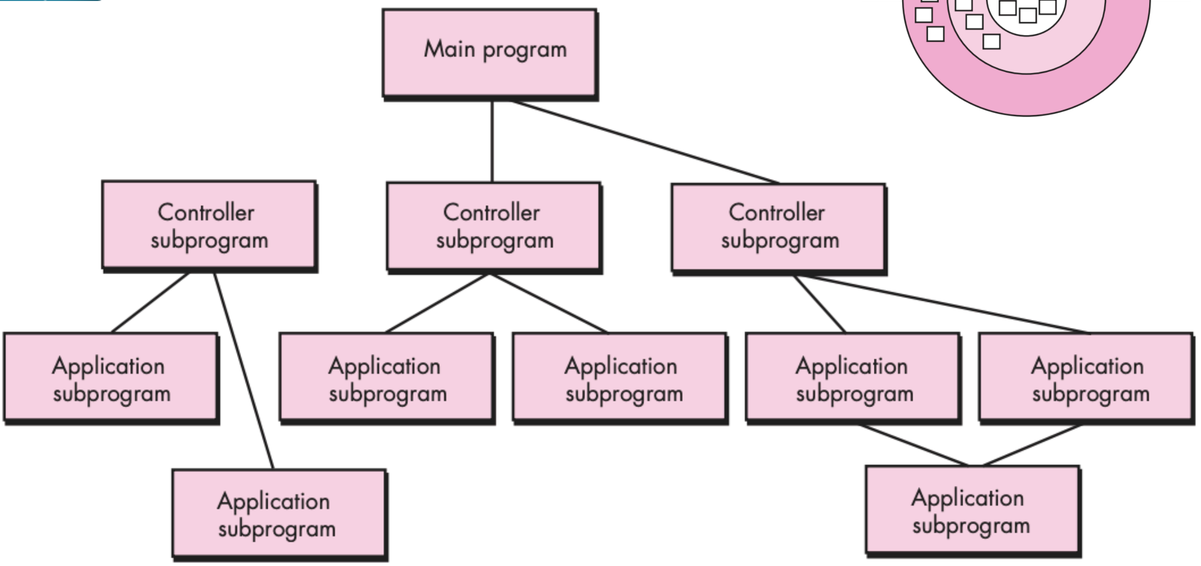
3. DESIGN DOCUMENTS

MORE EXAMPLES

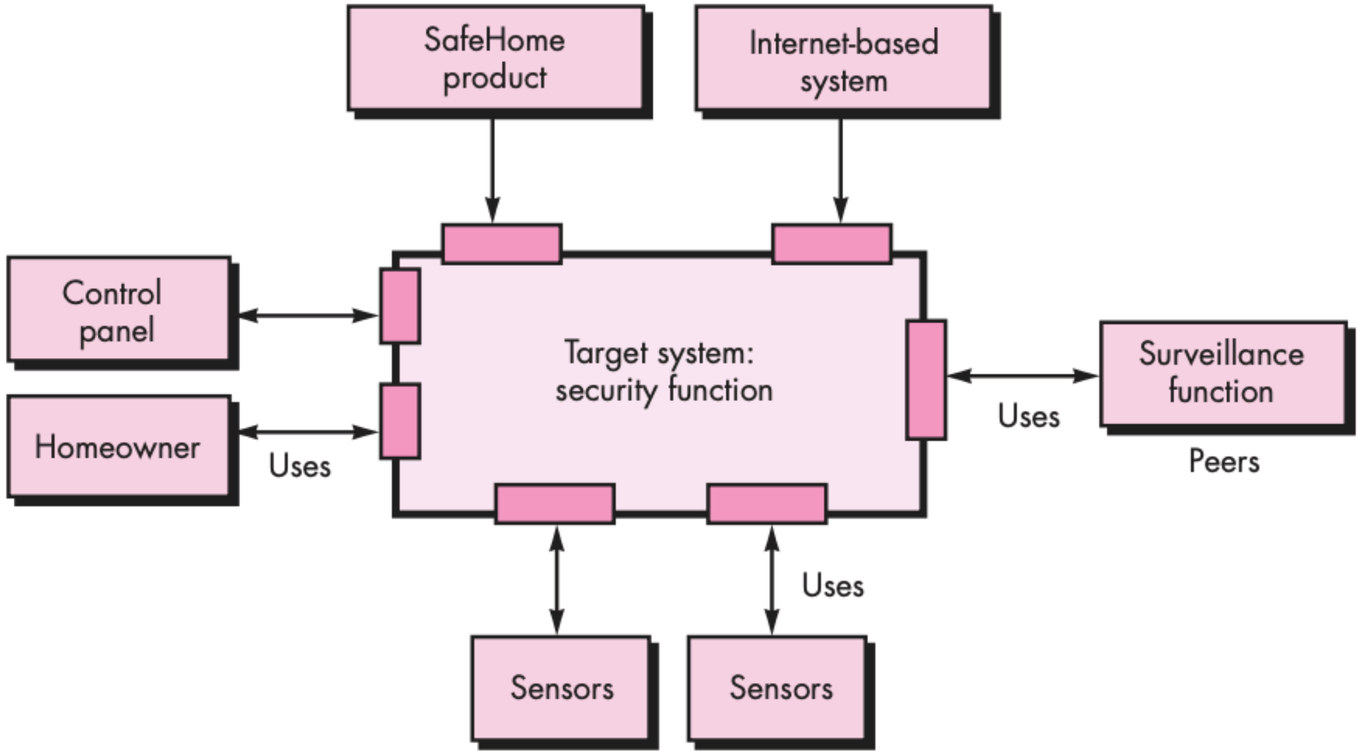
ARCHITECTURAL DESIGNS







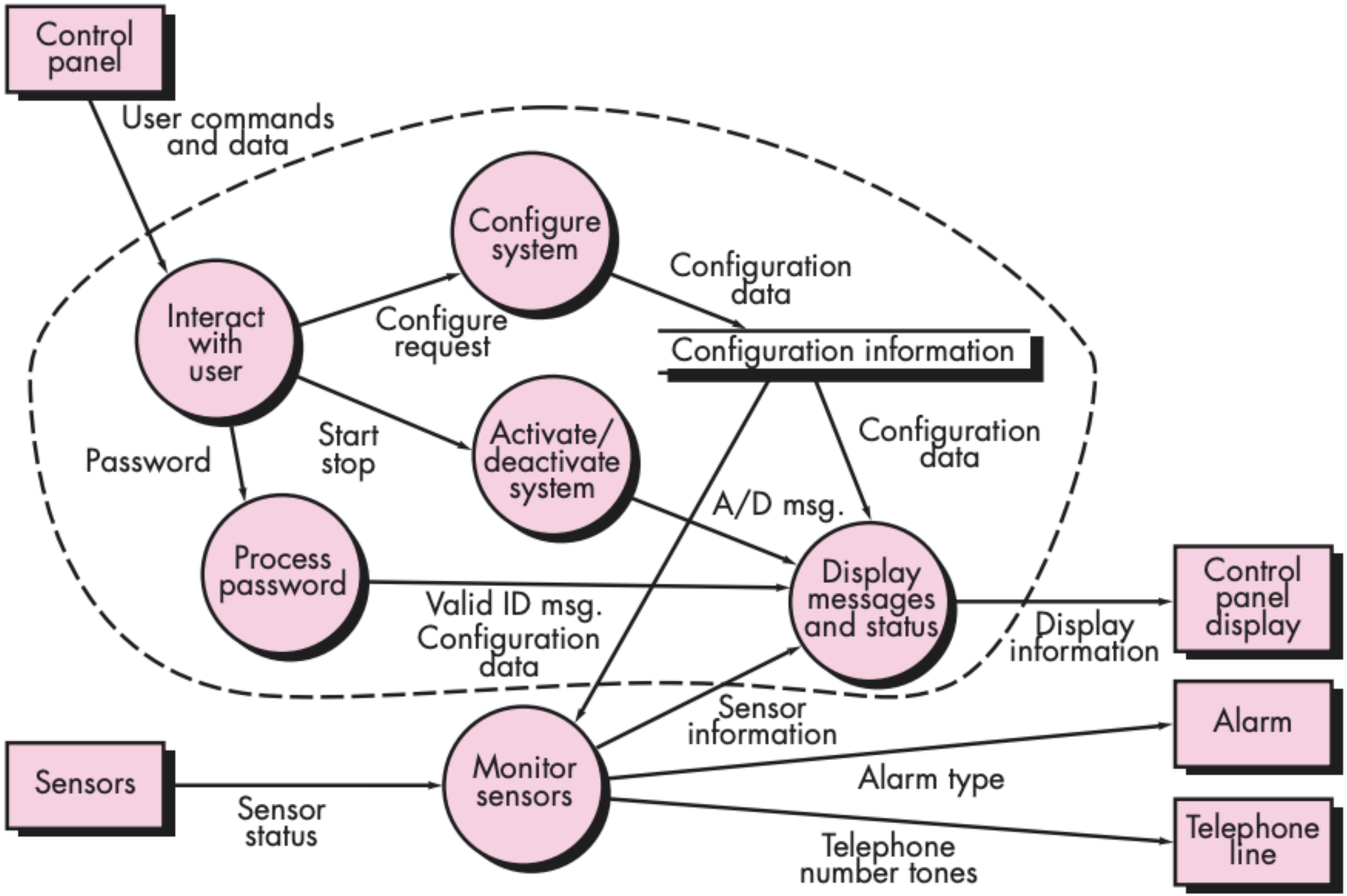
ARCHITECTURAL CONTEXT DIAGRAM



- Architectural design of Home Security System Function

3. DESIGN DOCUMENTS

DATA FLOW DIAGRAM

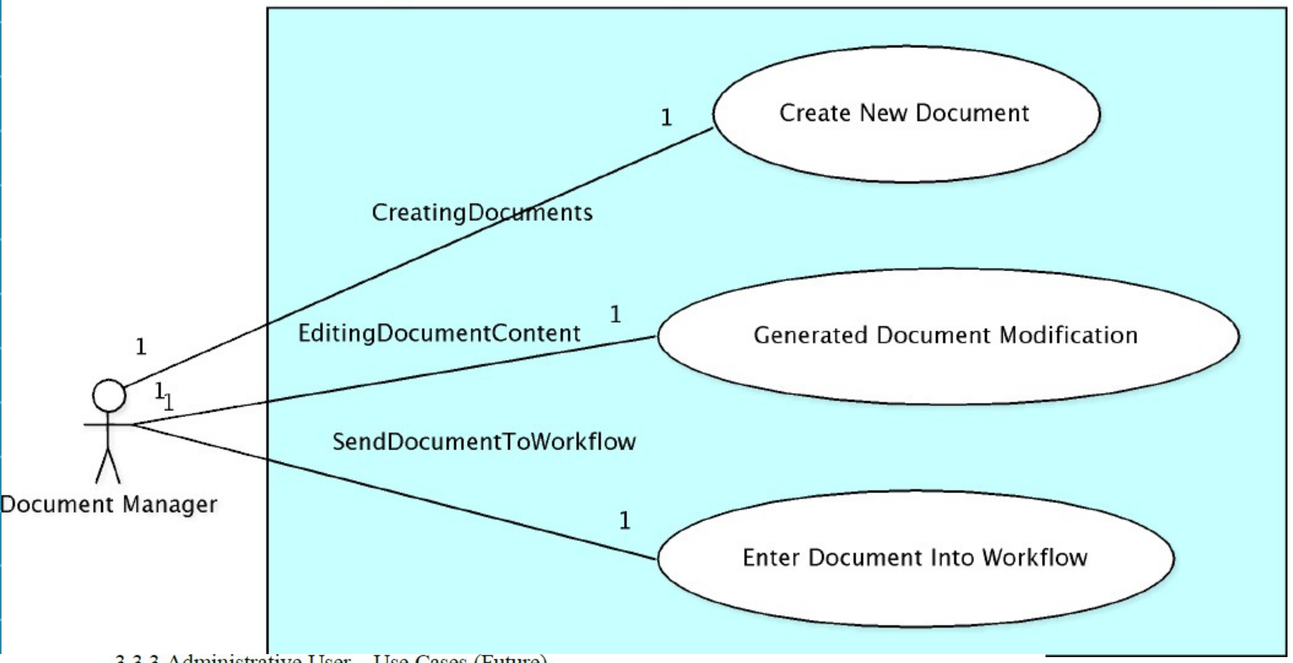


- It contains bubbles for processes and inputs to each, named actors.
- It is a mechanism to represent expected Data Flow.

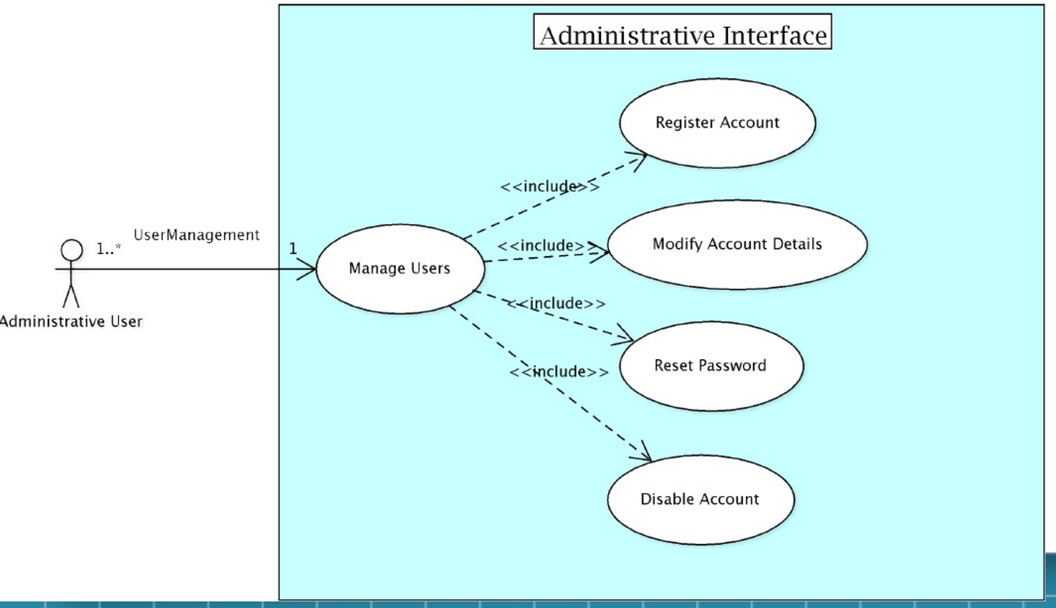
3. DESIGN DOCUMENTS

OTHER EXAMPLE

3.3.1 Document Manager- Essential Use Cases ("Enter Document into Workflow" for future update)



3.3.3 Administrative User – Use Cases (Future)



USE CASES *(left)* WITH DOC MANAGER *(right)*

Use case name: Create New Document		**ID:** *CND*	**Priority:** High
Primary actor: Document Manager	**Source:** Attorneys, Judges	**Use case type:** *Business*	**Level:** Overview

Interested Stakeholders:
Judge, Court Clerk, Attorney, Paralegal Professional

Brief description:
This use case describes the creation of a document which is a key function of the system. In this use case, the actor's goal is to generate a document.

Goal:
- The successful completion of document generation.

Success Measurement:
- The document is generated and reviewed by the user as acceptable for use.

Precondition:
- Document Management User has successfully passed through Authentication and Authorization
- Data sufficient to populate all required fields in a data set for a document has been entered into the system that will be used to draw data from to generate the document's data set.

Trigger:
- Document Management User has reached a point in their workflow in which a document is to be generated.

Relationships:
Include:
Extend:
Depends on:

Typical flow of events:
1. A document or set or related documents are selected to be generated.
2. The data from the case management system is pulled by the System Under Design based on the template and case record chosen to populate the document or documents data sets.
3. The Document Management User is allowed to preview the documents and summary of data set used to populate document
4. Once satisfied with the document and data, the user saves the document and enters it into a work flow such as sending to reviewer, or sending for signature.

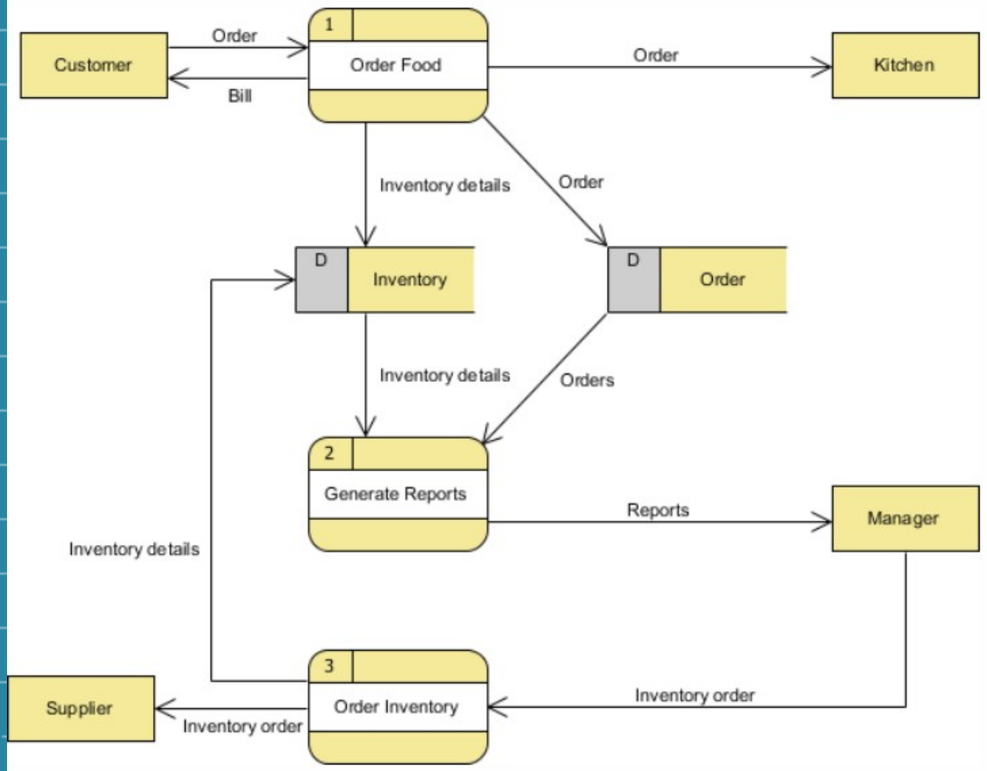
Assumptions
1. It is assumed that workflows will be carried out internally or with close partnered agencies that can be interacted with n a similar manner as with an internal system.
2. It is assumed that the case management system will hold appropriate data for use to generate documents.
3. It is assumed that a standardized template for a document is desired instead of using a free form document.

Implementation Constraints and Specifications:

3. DESIGN DOCUMENTS

OTHER EXAMPLES

DATA FLOW DIAGRAM FOR A FOOD ORDERING APP



ESSENTIAL SECTIONS TEMPLATE

Right:

NON-FUNCTIONAL REQUIREMENTS

10.1 Performance Requirements

- The system should be able to generate previews of documents within 15 seconds of user request.
- The system should be able to be multi-tasking to allow multiple users, up to 40 simultaneous users per interface instance to interact with the system without having to wait on others to finish working with the system.
- The system should be able to hold and search through large amounts of documents. The data structures used for the system will be fairly simple consisting of a few fields to hold document types and their related codes, XML instances with an id, and an audit log table, however the size of the simple data structures could potentially be quite large.
  - Expected capacity for large volume courts – approximately 108, 000 new documents a year with expected retention capacity of 10 years of active documents. After 10 years, documents can be stored in slower to access storage media. Which equates to approximately 1,080,000 documents that will need to be able to be stored and searched.

10.2 Design Constraints

- The software to be built should take advantage of open source libraries and supporting software, such as databases and web containers, unless an adequate open source product is not available or creatable for use.
  - The work will be licensed under an existing open source license, available at, http://license.gaje.us , and donated for use to standards committees that the agency participates in, such as the LegalXML Technical Committee.
- The software should adhere to locally or nationally recognized standards.
  - XML schemas should follow the National Information Exchange Naming and Design Rules, http://www.niem.gov/topicIndex.php?topic=file-ndr-0_3 .

When implemented in later versions, document retention schedules should follow the guidelines set forth by the Administrative Office of the Courts in Georgia for records retention, http://www.georgiacourts.org/aoc/records.php .

4. CODING OR DEVELOPMENT

TOOLS USED: Source Code Documentation

- **Definition:** The coding is the process of transforming the design of a system into a computer language format.
- **Purpose:**
  - To translate the design of system into a computer language format.
  - Deciding languages for the software.
  - Dividing the phase into modules for efficiency.

CHARACTERISTICS OF PROGRAMMING LANGUAGE

4. CODING OR DEVELOPMENT

Portability: High-level languages are machine-independent, facilitating the development of portable software.

Generality: Most high-level languages support a wide range of programs, reducing the need for programmers to become experts in multiple languages.

Brevity: High-level languages should allow the implementation of algorithms with minimal code. Programs in these languages are often shorter compared to low-level equivalents.

Error Checking: High-level languages often include robust error checking at both compile-time and run-time, helping catch and fix bugs early in the development process.

Cost: The overall cost of a programming language is influenced by its various features.

Quick Translation: The language should allow for quick and efficient translation processes.

Efficiency: It should enable the creation of efficient object code for optimal program performance.

Modularity: Programs should be developed as separate modules, compiled independently, with a structure that ensures self-consistency among these modules.

Widely Available: The language should be widely accessible, with translators available for major machines and primary operating systems.

4. CODING OR DEVELOPMENT

SOFTWARE QUALITY ASSURANCE

- **Some important points under Software Quality Assurance:**
- Standards: Imposed by stakeholders
- Reviews and audits: Ensuring financial security
- Testing
- Error/defect collision analysis
- Change of management: Leads to chaos and confusion
- Education: Training programs to employees
- Vendor management: Third party usage
- Risk management: Identifying any potential risks.
- Security management: Adhering to latest firewall; protection against cyber-threats
- Safety: Assessing impact of software failure and preparing backups (if required)

5. TESTING

TOOLS USED:

Test Plan

- **Definition:** Before deploying the software to the end users the software is tested in different test environment.
- **Purpose:**
  - Run our software to check if it is giving correct result or not.
  - Helps to improve quality of the project.
  - Helps to find defects in the system.

Commonly used popular Tools:

QTP

Selenium

LoadRunner

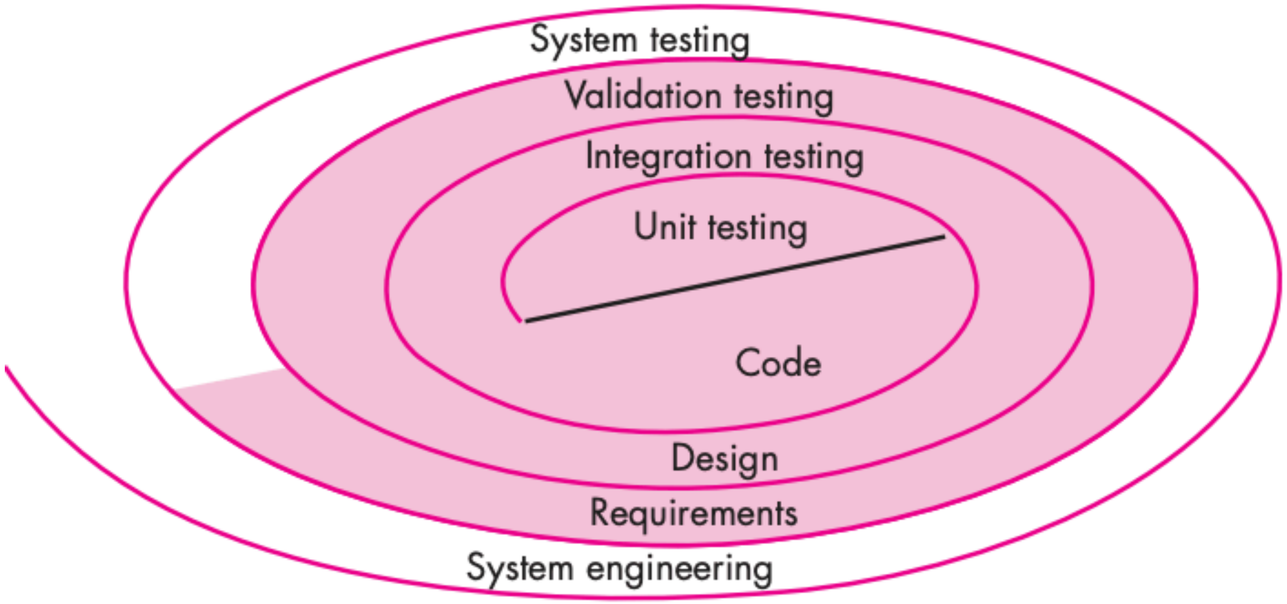
5. TESTING

- **Types:**
  - **Blackbox Testing:** Functionalities of software applications are tested without having knowledge of internal code structure .
  - **Whitebox Testing:** software testing technique in which internal structure of the software must be known like data structure , control statements.
  - **Integration Testing:** Combining all modules and checking for any unprecedented errors.
  - **Verification and Validation** *(V & V)*:

Verification: "Are we building the product right?"

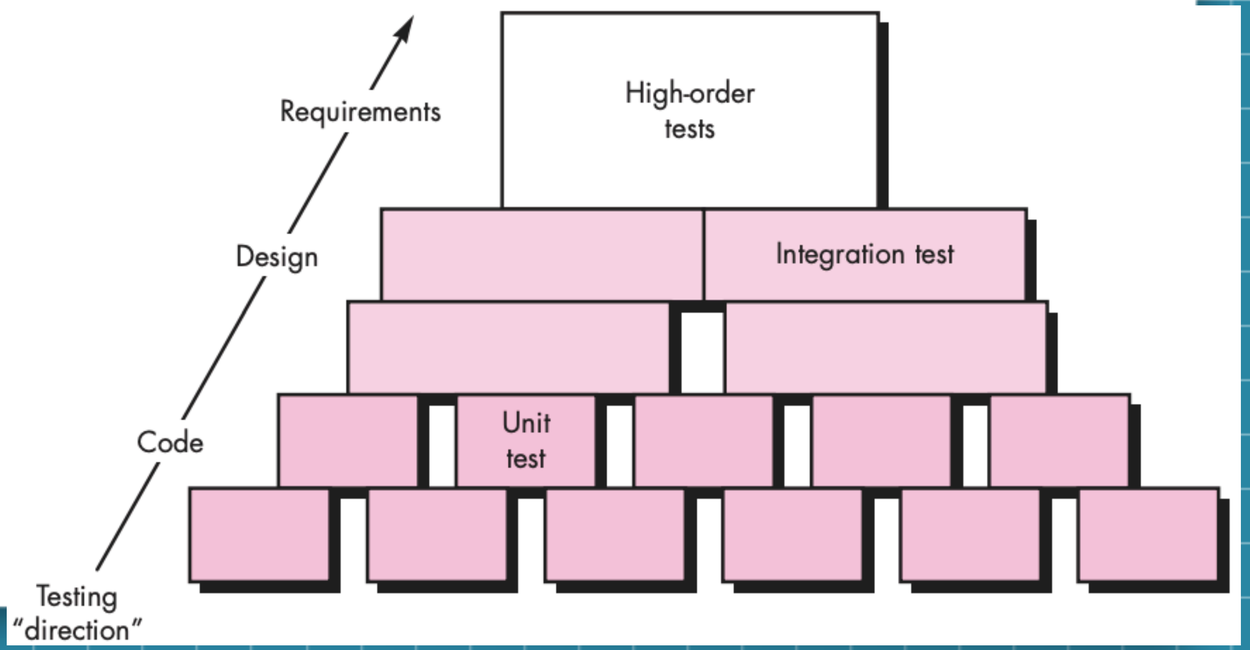
Validation: "Are we building the right product?"

5. TESTING



- *Left:* Strategy for Testing

- *Right*: Software Testing steps



6. MAINTENANCE

TOOLS USED:

Maintenance Plan

- **Definition:** Maintenance is the backbone of software success. When software is developed, then the software needs to be modified from time to time to keep it up to date with the environmental changes and changing the user requirements.
- Purpose:
  - For error correction
  - Change in user requirements with time.
  - To upgrade hardware and software requirements with time.
  - Costly to maintain old technology
  - To enhance system efficiency
  - To optimise the code to run faster.
  - To modify the components.
  - To reduce any unwanted side effects.
  - Undocumented code may lead to confusions

6. MAINTENANCE

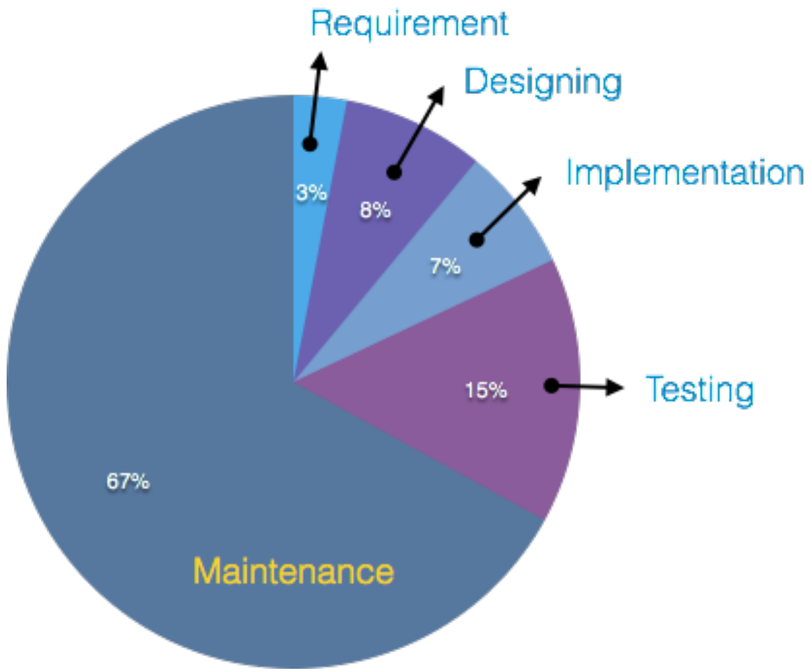
- The people who made the software may no longer be around or associated with that company. Therefore, effective modularity and proper design techniques will enable the best possible transfer of details of the application to the next set of people.

CORRECTIVE

ADAPTIVE

PERFECTIVE

PREVENTIVE



- *Alongside:* A graph depicting the importance of each step to the contribution of the Home Security System.

SOFTWARE DEVELOPMENT LIFE CYCLE: CASE STUDY 2

SOCIAL MEDIA PLATFORM (Facebook)

PHASE - 1

PROJECT INITIATION PHASE

REQUIREMENT GATHERING

The User Interface: Focus on creating a simple but elegant user interface that enhances usability. The platform user should be able to register and use the app without hassles..

Push Notifications: Adding a feature that enables real-time notifications for the features that the user selects can make a huge difference.

User Profile: The ability to make extensive modifications to the profile from the app interface can attract more users and retain existing ones. The pleasant experience of the app users is vital to keeping them on your app platform.

News Feed: The news feed feature allows the user to receive the latest news updates. It should also be considered to have coding private, group, and public chat options.

File Sharing Feature: These days, social media apps integrate image, audio, and video content sharing features. Emojis and stickers are also becoming common, so your Facebook-like app should integrate them as well.

Customisation: Make it easy for your app users to add, remove, or edit friends, connections, and groups.

Messaging Feature: WhatsApp and Facebook Messenger have acquired a massive user base, indicating that people have a deep interest.
in Instant messaging. So, adding instant messaging features to the social media platform will increase its popularity. These days, most IM apps also include file sharing features.

SRS DOCUMENTATION

PREPARATION OF BUSINESS MODEL / PROTOTYPE *(Sample)*

1. FACEBOOK STRUCTURE
1.1. News Feed
1.2. Friends List
1.3. Wall
1.4. Timeline
1.5. Likes and Reactions
1.6. Comments
1.7. Messages and Inbox
1.8. Notifications
1.9. Groups

2. APPLICATIONS
2.1. Events
2.2. Marketplace
2.3. Notes
2.4. Places
2.5. Platform
2.6. Facebook Questions
2.7. Photos
2.8. Videos
2.8.1. Live streaming
2.8.2. Controversial Usage
2.9. Facebook Paper
2.10. Mentions
2.11. Moments
2.12. Gaming
2.13. Podcasts

3. GENERAL FEATURES
3.1. Facebook dynamic text/type
3.2. Credits
3.3. Feature phones
3.4. Graph Search
3.5. IPv6
3.6. Listen with Friends
3.7. Mood faces
3.8. Phone
3.9. Poke and Greetings
3.10. Smartphone Integration
3.11. Fundraising
3.12. Status Updates
3.13. Subscribe
3.14.. Ticker
3.15. URL Shortener
3.16. Verified Accounts
3.17. Hash-tagging support
3.18. Impressum
3.19. Hidden service
3.20. Say "thanks"
3.21. Call-to-action button
3.22. Snooze
3.23. "Did you Know": Questionnaires
3.24. Music Archive
3.25. Off-Facebook Activity
3.26. Memories

4. SECURITY

PHASE - 2

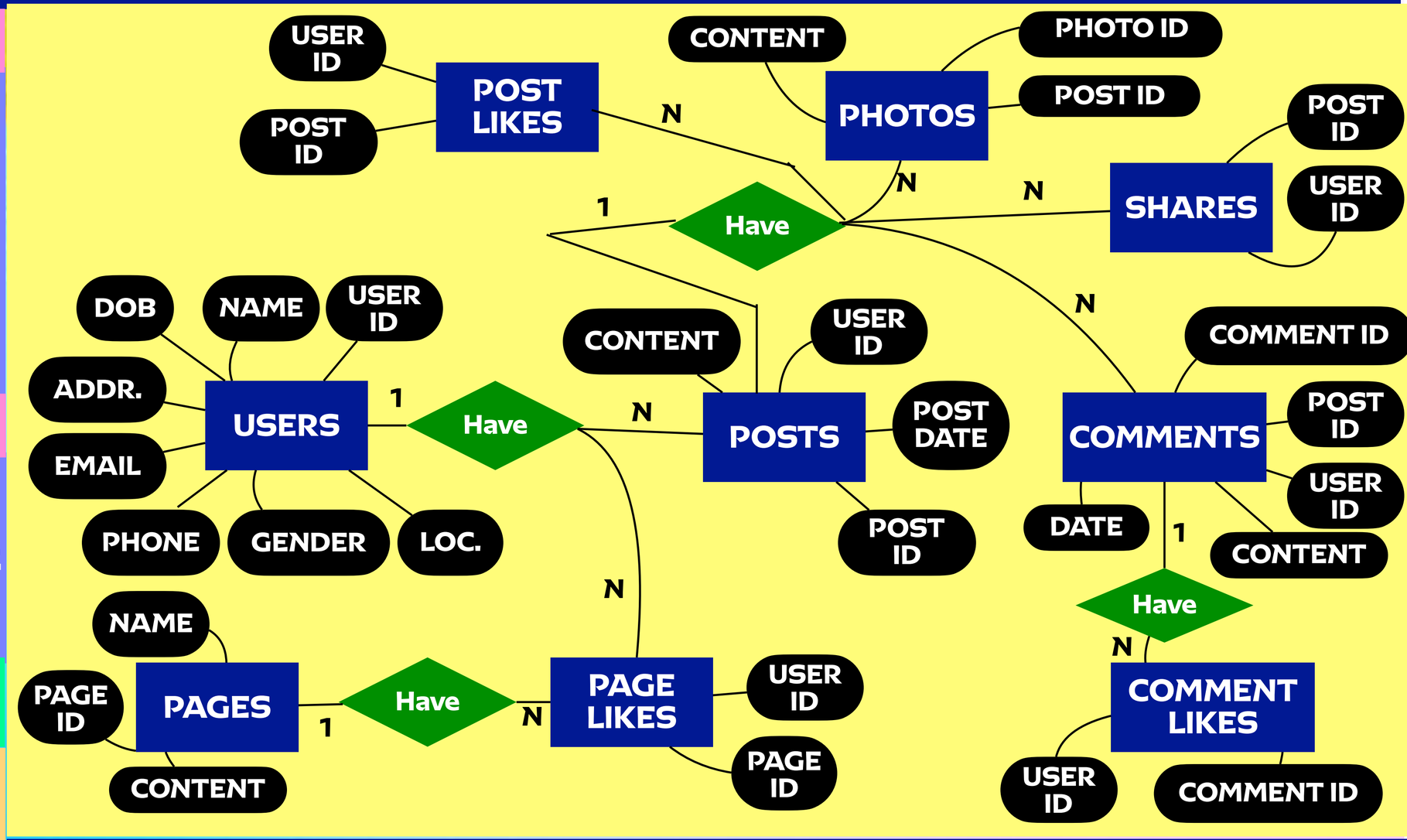
REQUIREMENTS SPECIFICATION PHASE

ENLISTING REQUIRED ATTRIBUTES

USER	PAGE
ID First Name Last Name DateOfBirth hashPassword Email Phone About	iD Name DateOfCreation createdBy Email About totalLikes

POST	CHAT	FRIEND
ID Title DateOfCreation createdBy Likes	User1 User2 Text Time	Person1 Person2 Time

USER-POST RELATIONSHIP	USER-PAGE RELATIONSHIP
UserID PostID Time Flag	UserID PageID Time

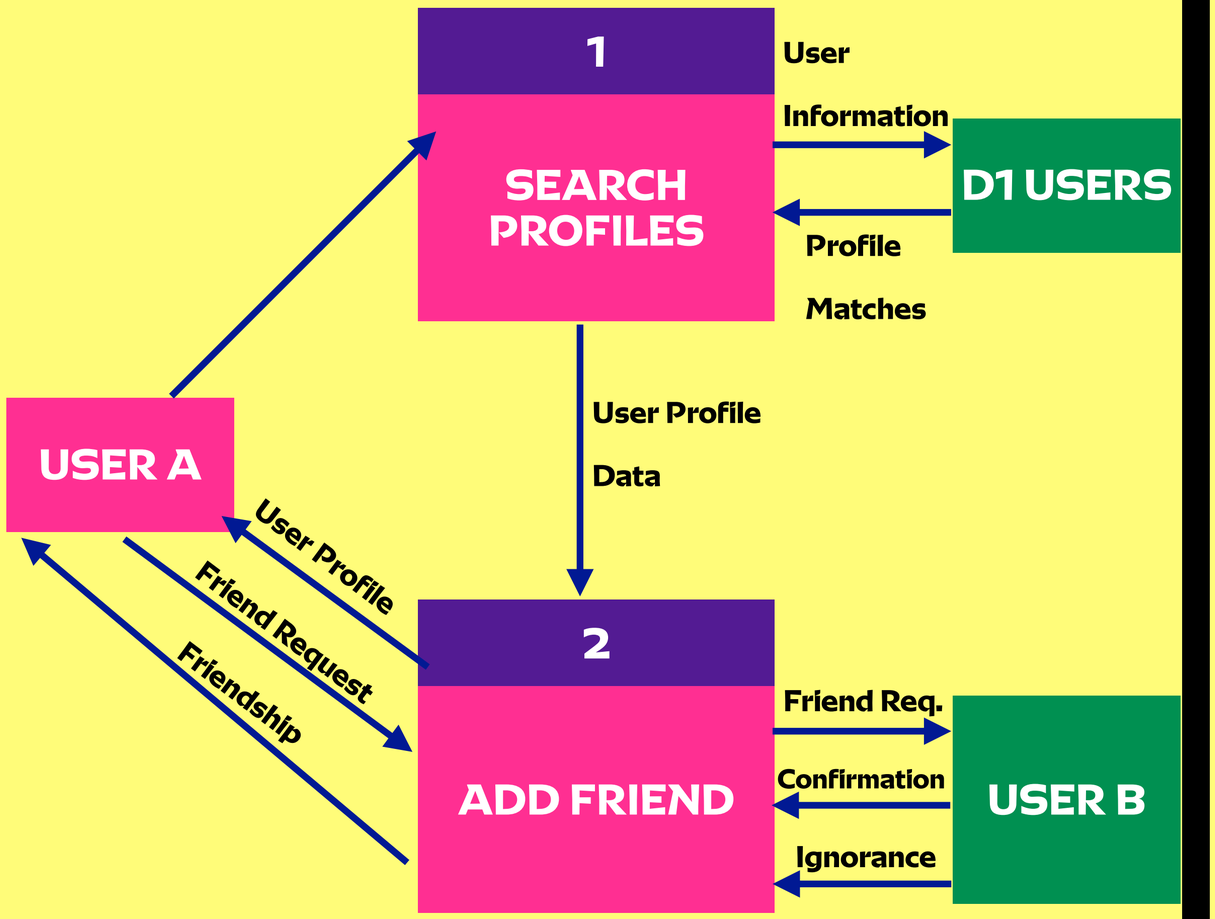


DESIGNING AN ENTITY RELATIONSHIP (ER) DIAGRAM

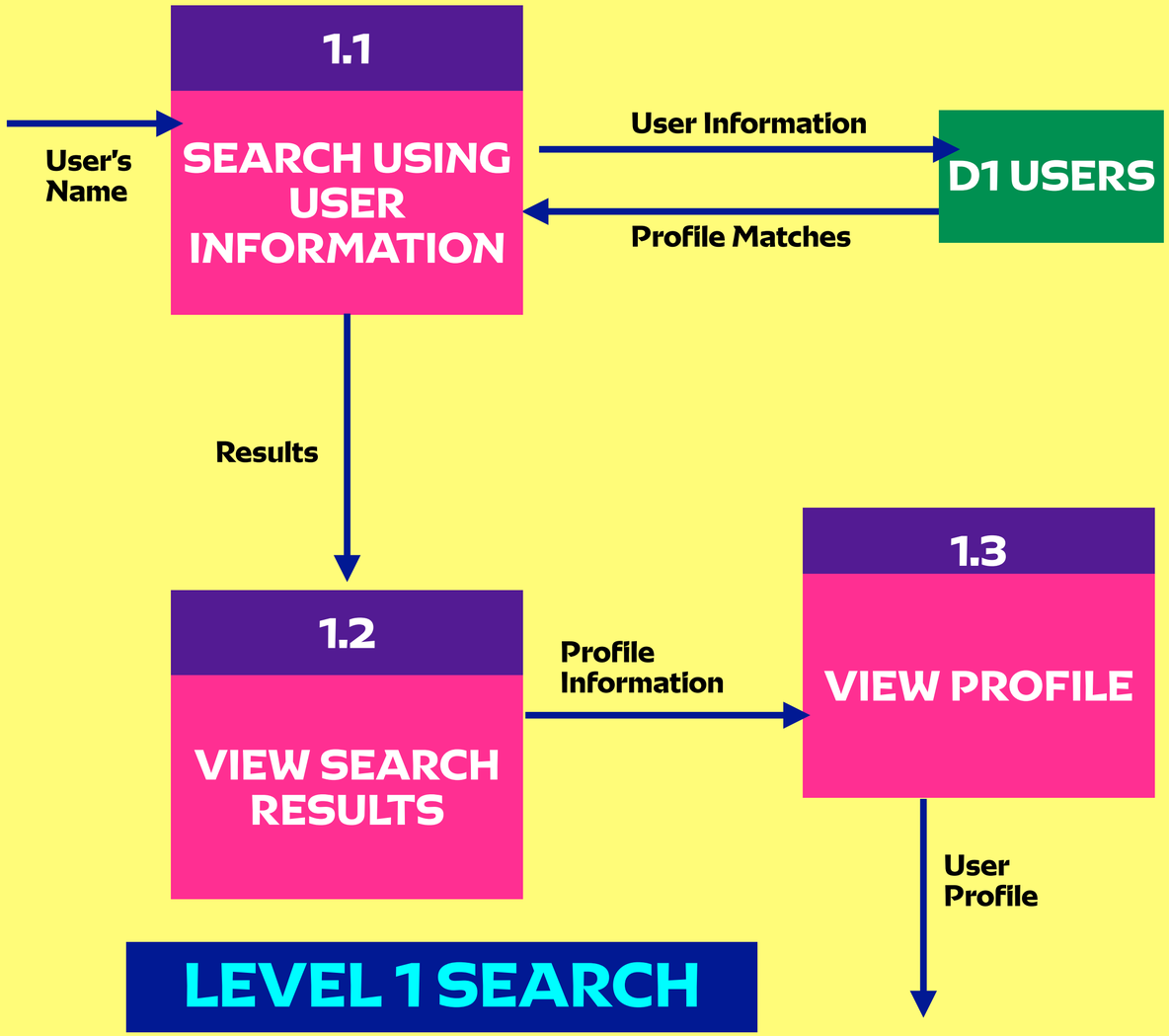
PHASE - 3

DESIGN PHASE

DESIGN : DATA FLOW DIAGRAM



LEVEL 0 SEARCH

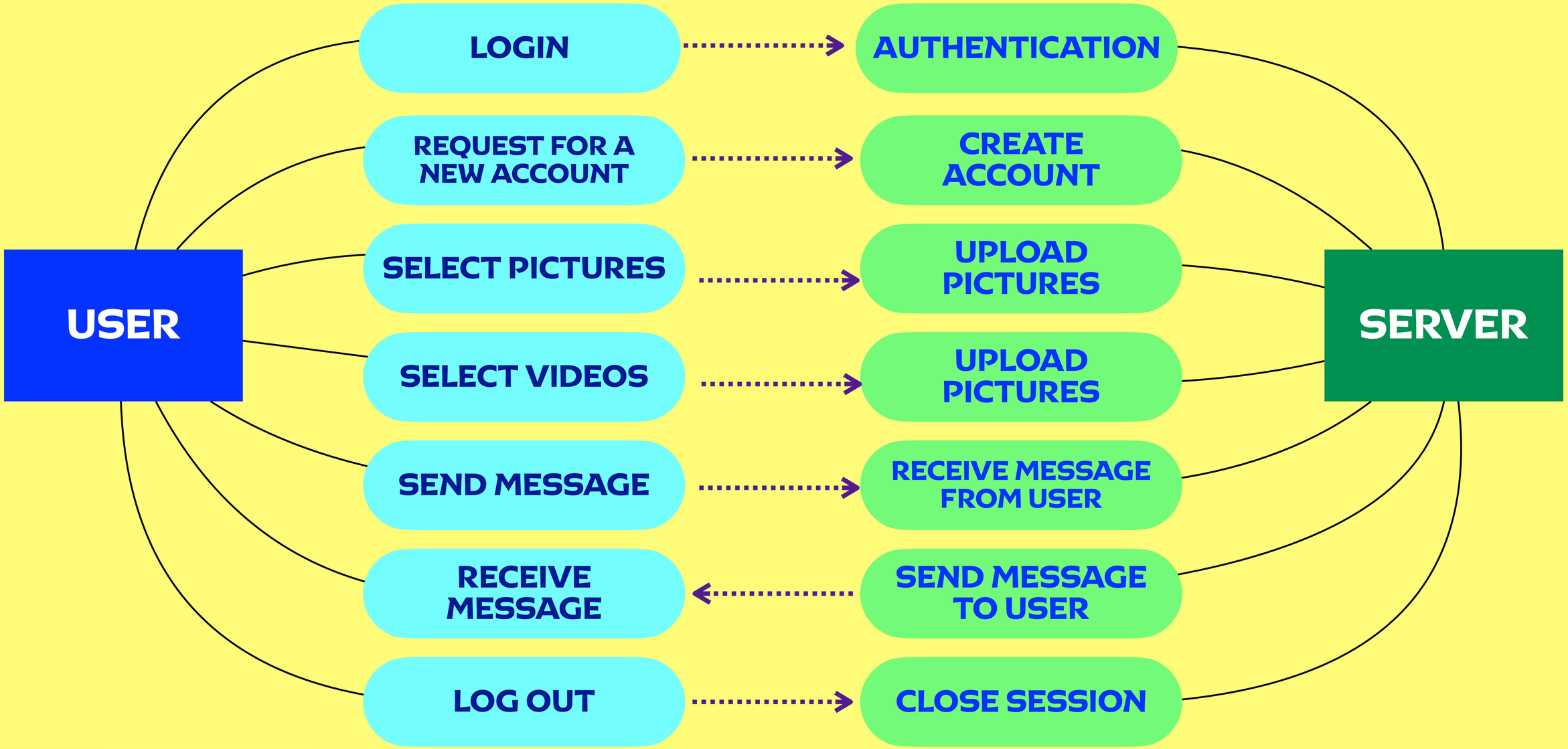


LEVEL 1 SEARCH

PHASE - 3

DESIGN PHASE

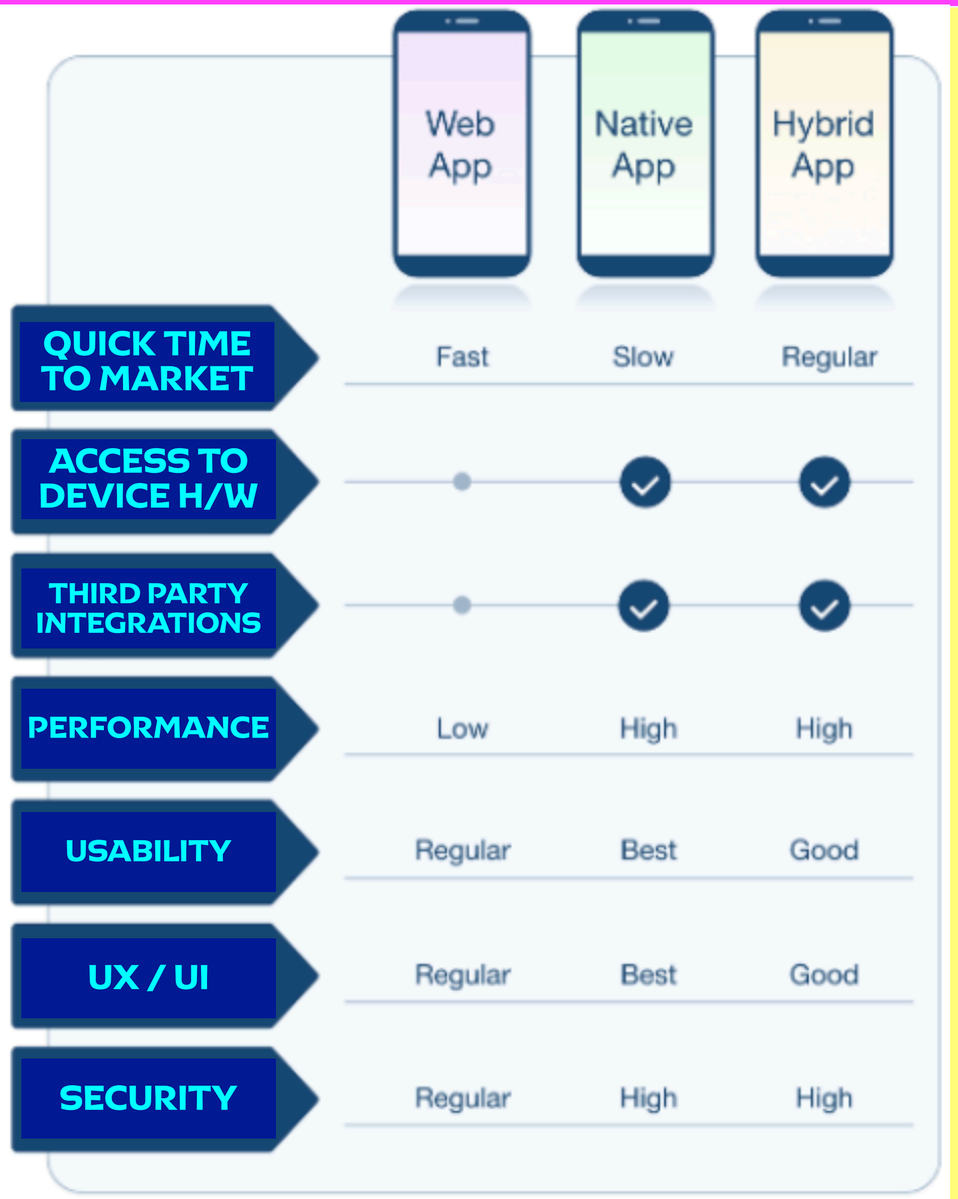
OVERALL DATA FLOW SKETCH



PHASE - 4 : BUILDING CODING AND DEVELOPMENT PHASE

PROGRAMMING LANGUAGES CONSIDERED	
JAVA	BACK-END
HACK	CREATED BY FACEBOOK BACK-END
C++	BACK-END
ERLANG	BACK-END
PYTHON	BACK-END & MACHINE LEARNING
HASKELL	ANTI-SPAM
DLANG	GPU
PHP	SERVER-SIDE WEB DEVELOPMENT
JavaScript	FRONT-END

CODING THE APP FRONTEND



Points to be noted:

- Appearance must be clean
- Three types of apps can be built: Native, Web, Hybrid
- The difference between the three types is depicted alongside.

PHASE - 5

TESTING PHASE

DEBUGGING AND TESTING

MODULAR AND INTEGRATED TESTING

TEST APPS: Cloning parent apps without compromised functionality in Development Mode. Administrators can perform Admin actions on the. Application, while Testers are bound to their domain.

TEST USERS: Simulated accounts (not real people) only visible to Administrators, Developers and Testers. They perform actions like a user but is exempt from Spam and fake account detections.

TEST USER GRAPH API ENDPOINTS: If the App Dashboard is not sufficient for the requirement, or more than 4 testers are required at a time, then a useful tool is Graph API. It has 2 accounts - Application and Test for their respective purposes.

PHASE - 6

MAINTENANCE PHASE

With increasing cyber risks, threats and increase in exploitation of Artificial Intelligence (AI) in the field of hacking, keeping applications up to date with the latest security is very important. Customers rely on the application to keep their personal and private data safe, and violating the same is not ethically or morally correct. With the advancement of technology, the need for newer features arises. For instance, short video reels, communities, editing messages after sending them, etc.

SOFTWARE DEVELOPMENT LIFE CYCLE: CASE STUDY 3

ROLE OF PROJECT MANAGER:

HEALTH APPLICATION TO TRACK USER'S SLEEPING PATTERN

1 PLANNING

ROLE: Understand the market need, define the product's scope, set objectives, and allocate resources.

- **Conducting market research, identifies the target audience, and setting a budget and timeline.**

2 SYSTEMS DESIGN

Ensuring that the design is user-friendly, displaying sleep data graphically. They also ensure the system can handle data from millions of users without crashing.

ROLE: Collaborate with designers to create user interfaces and work with system architects to design the software's architecture.

3 DEVELOPMENT

ROLE: Oversee the coding process, ensuring alignment with the design and timely completion.

Organising development sprints, prioritising tasks, and ensuring the team adheres to the design specifications

4 TESTING

ROLE: Collaborate with Quality Assurance (QA) teams to identify and rectify bugs.

For instance, During testing, the QA team discovers that the app crashes when users try to sync sleep data with other health metrics. The PM ensures this bug is fixed before launch.

5 DEPLOYMENT

- **Role: Supervise the software's rollout to users.**
- **Deciding to release the sleep tracking feature in phases, initially to a select group of users, gathering feedback before a full-scale launch.**

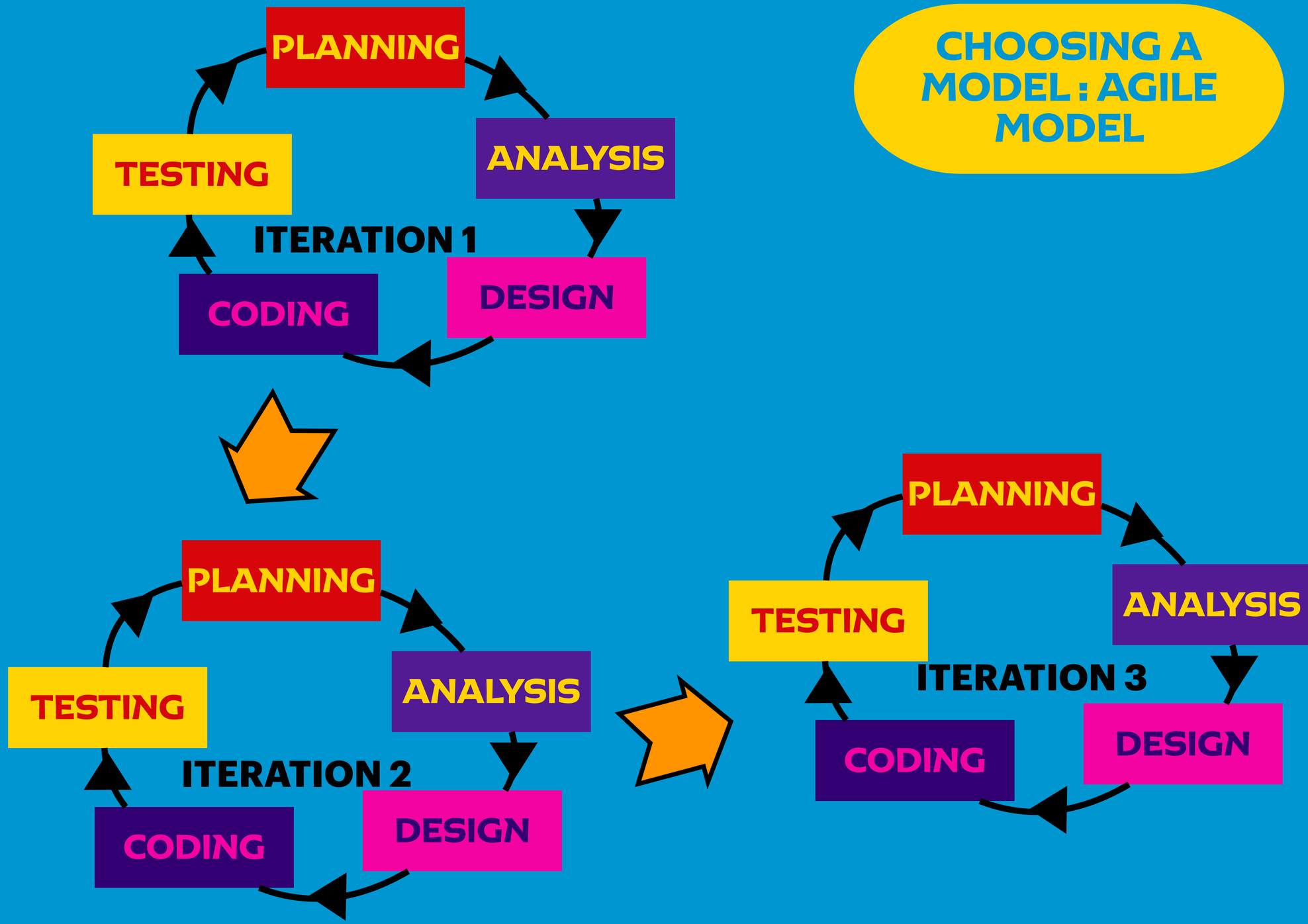
6

MAINTENANCE

ROLE: Monitor the software post-launch, gather user feedback, and plan updates.

For instance, After the feature's launch, users suggest an integration with smart home devices to monitor room conditions for better sleep. The Project Manager will consider this for future updates.

CHOOSING A MODEL : AGILE MODEL



WHY THE AGILE MODEL? : COMPARISON OF MODELS

An Agile model might be the best, allowing for iterative development and incorporating user feedback

Cost and Time are fixed factors at the time of assignment of the Project. While there are not many features, quality is very important for this application. It requires Trial and Incremental process, to be heavily based on user feedback.

DELIVERABLES OF SDLC:

	Waterfall	Iterative	Agile	RAD
Features	Fixed	Fixed	Negotiable	Negotiable
Quality	Fixed	Fixed	Fixed	Negotiable
Cost	Negotiable	Negotiable	Fixed	Fixed
Time	Negotiable	Fixed	Fixed	Fixed

CIRCUMSTANCES BASED GUIDELINE

	Waterfall	Iterative	Agile	RAD
Monetary Project	Suitable	Suitable	Not Suitable	Not Suitable
Incremental Project	Not Suitable	Suitable	Suitable	Suitable
High User Involvement Project	Not Suitable	Suitable	Suitable	Suitable
Trial Project	Not Suitable	Not Suitable	Suitable	Suitable

PHASES OF SDLC:

	Waterfall	Iterative	Agile	RAD
Plan	Important	Important	Less Important	Less Important
Design	Important	Important	Less Important	Less Important
Develop	Less Important	Less Important	Important	Important
Test	Important	Important	Less Important	Less Important
Deploy	Important	Important	Important	Important
Maintain	Important	Important	Less Important	Less Important

BIBLIOGRAPHY / REFERENCES

- SDLC: https://thestudygenius.com/sdlc-software-development-life-cycle/
- Understanding SDLC: https://www.sdlcforms.com/UnderstandingSDLC.html
- Case Study: https://media.neliti.com/media/publications/264195-a-case-study-on-identifying-software-dev-9339d3f6.pdf
- Case Study II: https://www.studocu.com/in/document/rk-university/business-management/prepare-a-sdlc-report-on-facebook-bhargav-1/30352174?shared=u&sid=01705407410
- Document Report: http://home.cse.ust.hk/~rossiter/independent_studies_projects/software_development/software_development_report.pdf
- E-Book: https://www.mlsu.ac.in/econtents/16_EBOOK-7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman_.pdf
- Templates: https://www.volere.org/templates/volere-requirements-specification-template/

OTHERS

- BalticLSC Software User Requirements Specification
- (TECHNICAL SPECIFICATION) DEVELOPMENT OF WEB APPLICATION FORAMIS/ZPIS
- UML Sequence Diagram - Javatpoint
- https://www.javatpoint.com/uml-state-machine-diagram
- Software Design Document: What is it & How to Create it! (Template Included)
- Best Software Design Document Templates - DevTeam.Space
- Data Flow Diagram: Examples - Food Ordering System
- Manual Testing Tutorial: What is, Concepts, Types & Tool
- Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE

ACKNOWLEDGEMENT

- We would like to thank our Professors for providing this opportunity of exploring about SDLC and improving our Presentation skills. We are forever indebted to the Lord, God Almighty, for without His marvellous divine creations nothing of this would be possible. Our heartfelt gratitude goes forward to all our elders, parents, teachers, guides and well-wishers for their support and blessings all throughout.

THANK YOU