Generic Workflow Engine

Design Documentation

Prepared by
Kapil Dev(2017csb1085)
Aman Kumar(2017csb1067)
Kartikeya Pise(2017csb1086)
Soumya(2017csb1114)
GROUP:-9

IIT Ropar
29 Jan 2020

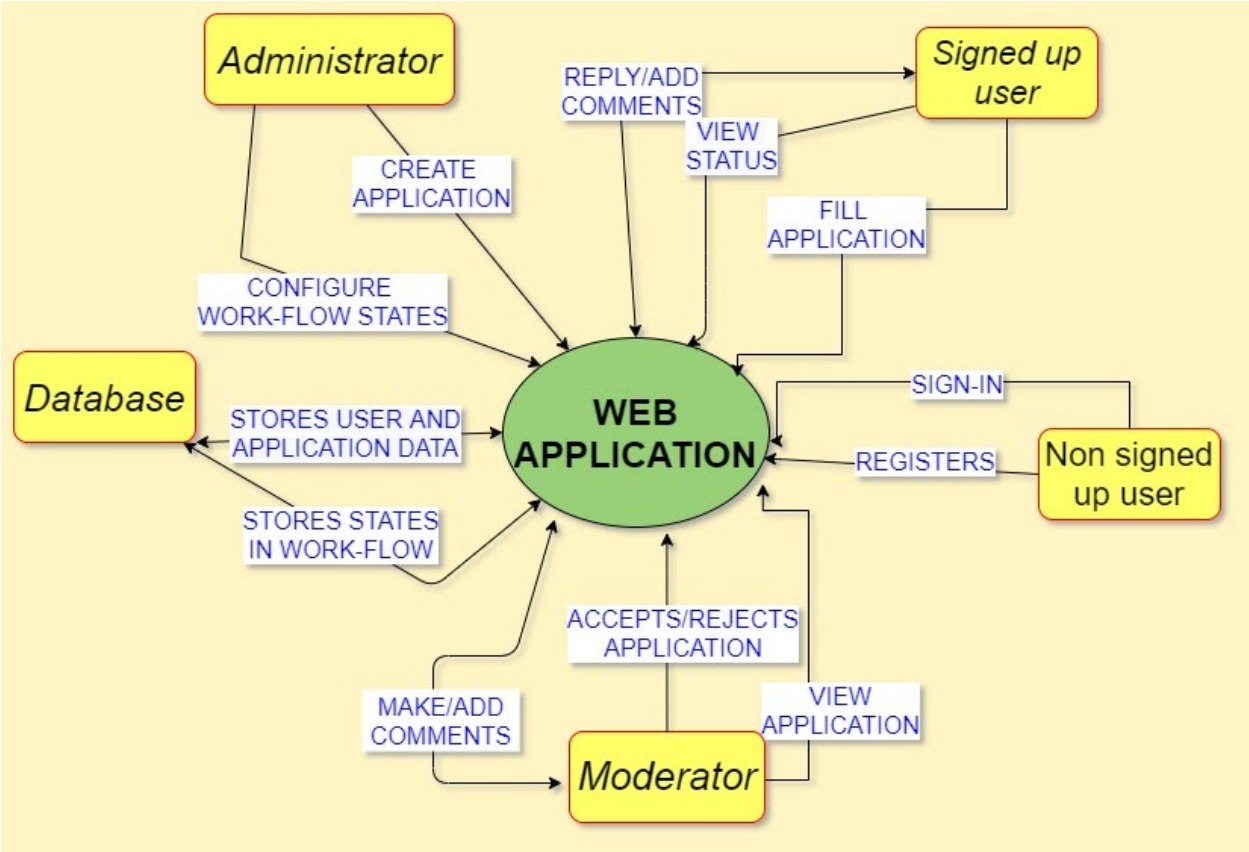
Introduction :

A **software design document**(**SDD**) is a written description of a software product, that a software designer writes in order to give a software development team overall guidance to the architecture of the software project. An SDD usually accompanies an architecture diagram with pointers to detailed feature specifications of smaller pieces of the design. Practically, the description is required to coordinate a large team under a single vision, needs to be a stable reference, and outline all parts of the software and how they will work. This design document provides different design orientations to help to understand the working of the software. It also highlights the design decisions are taken and the way all the entities interact in our systems through multiple diagrams giving a different view each.

Context Diagram :

The Context Diagram shows the system under consideration as a single high-level process and then shows the relationship that the system has with other external entities
Another name for a Context Diagram is a Context-Level Data-Flow Diagram or a Level-0 Data Flow Diagram.

CONTEXT DIAGRAM FOR OUR MODEL



Explanation:

So in this context diagram we have five entities :

1) Moderator: Moderator uses the portal for performing the assigned roles and to use all the functionalities he is having after this web application escalates the form to the required state in the chain.

2) Database: With the help of the web application it stores states and application data and it is queried again and again.
3) Administrator: To create an application form it assigns tasks and entries and gives the data to the web application. Also to configure workflow he sends the data like how many states to add .
4) Signed Up Users : These are logged in users whenever they need to generate a new form or task or to reply or comment to a query they send their data in the form to a web application.
5) Non signed up user : Same as above to register to the portal they need to send their registering details to the portal.

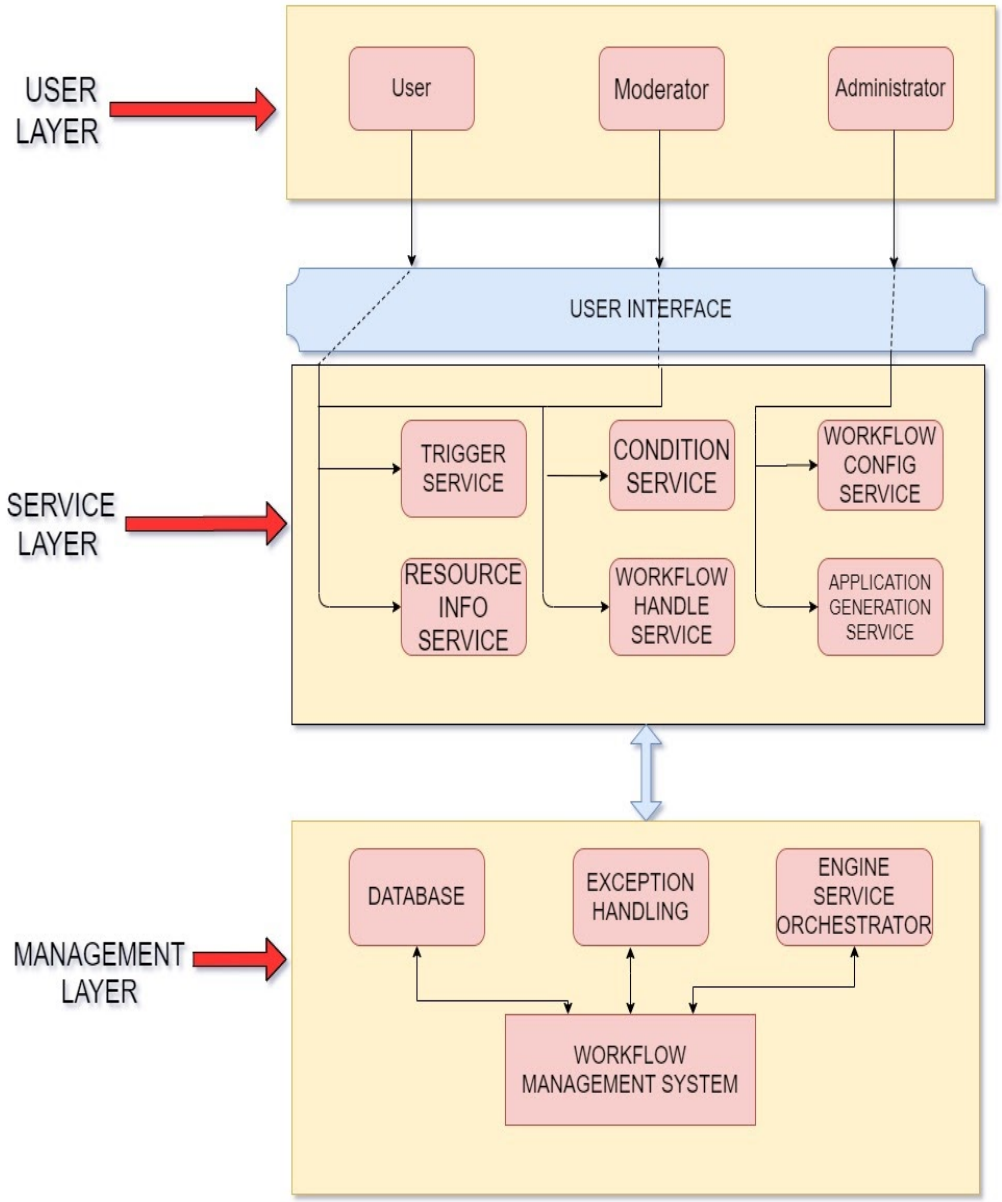
Logical Structure Diagram:

A structure diagram is a conceptual modeling tool used to document the different structures that make up a system such as a database or an application. It shows the hierarchy or structure of the different components or modules of the system and shows how they connect and interact with each other. It is a tool used to guide developers to ensure that all parts of the system work as intended in relation to all the other parts.

A logical structure diagram is a model of how a business operates, and the processes it depicts represent business activities. The logical structure diagram shows points where data is collected regardless of the type of storage used. Logical data flow diagrams are easier to understand for non-technical personnel, so they can help bridge the gap between technical and business requirements. They can also help foster collaboration and communication about process changes.

A system built with a logical structure diagram is thought to be more stable because it's based upon business processes instead of transitory business technologies. The business processes shown will likely continue to exist regardless of the physical means used to accomplish them. For example, businesses will always have a need to monitor payments to vendors, regardless of whether invoices and payments are tracked manually in a ledger book or through a software program.

LOGICAL STRUCTURE DIAGRAM FOR OUR MODEL



LOGICAL STRUCTURE DIAGRAM

Explanation:

The above diagram describes the entire workflow system into 3 major layers where each layer describes certain functionalities and models. Each layer is interconnected and individually each layer comprises certain qualities which distinguishes them from the other and modularized for better management of the system.

The three layers are:

1. User Layer: It shows the different users in the system and comprises of their data and all the login and other functionalities done through user interface.
2. Service Layer: This layer describes all the services that the system provides in a modular fashion. These services are provided to the users using the user interface and gets configured through the workflow management layer.
3. Workflow management Layer: This layer is the backbone of the entire system where all the configuration of the system is done. Here all the database management is done as well as new functionalities are added if required for specific tasks.

Data Flow Diagram:

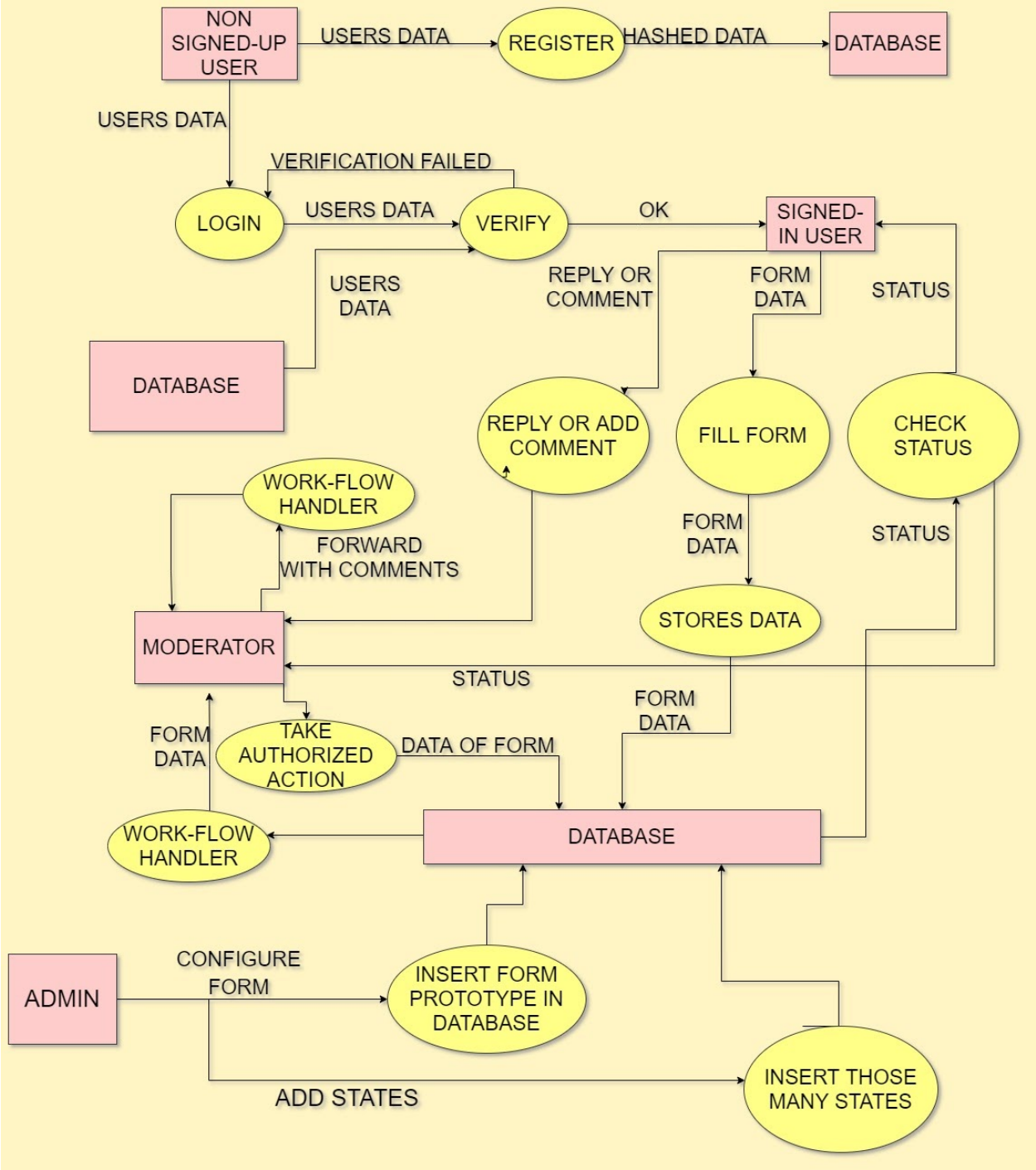
A Data-Flow Diagram is a graphical visualization of the movement of data through an information system. DFDs are one of the three essential components of the

structured-systems analysis and design method (SSADM). A DFD is process-centric and depicts 4 main components.

- Processes (circle)
- External Entities (rectangle)
- Data Stores (two horizontal, parallel lines or sometimes and ellipse)
- Data Flows (curved or straight line with arrowhead indicating flow direction)

Each DFD may show a number of processes with data flowing into and out of each process. If there is a need to show more detail within a particular process, the process is decomposed into a number of smaller processes in a lower level DFD. Data-flow diagrams can be regarded as inverted Petri nets because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. The structure of DFD allows starting from a broad overview and expands it to a hierarchy of detailed diagrams.

DATA-FLOW DIAGRAM FOR OUR MODEL



In this dfd we describe the interaction of various entities with different processes like none signed up user sends its data to the process register process .
So in this dfd we describe different processes like :

1) Workflow Handler : This process manages to keep track of who should be the next state to receive the form in chain.
2) Reply/add comment : This process allows users to respond to the query generated by the clients at higher states in the chain.
3) Take Authorised action: This process allows specified states to add comments and reply to the request they have got it includes reject, forward, ask for more details.

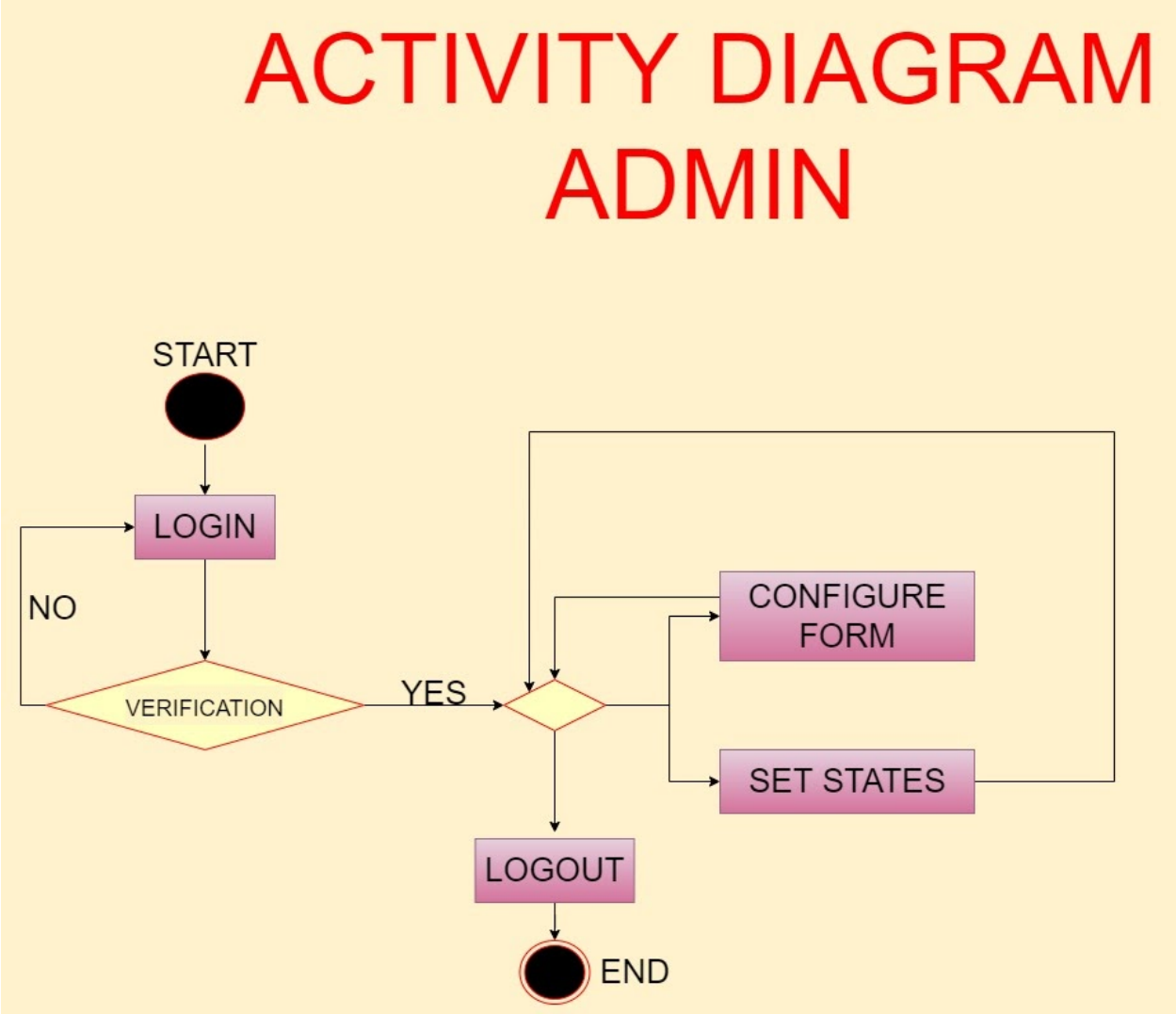
Activity Diagram:

Activity diagram is another important behavioral diagram in the UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of the flow chart that modeling the flow from one activity to another activity.Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single-use case relate to one another, in particular, use cases where activities may overlap and require

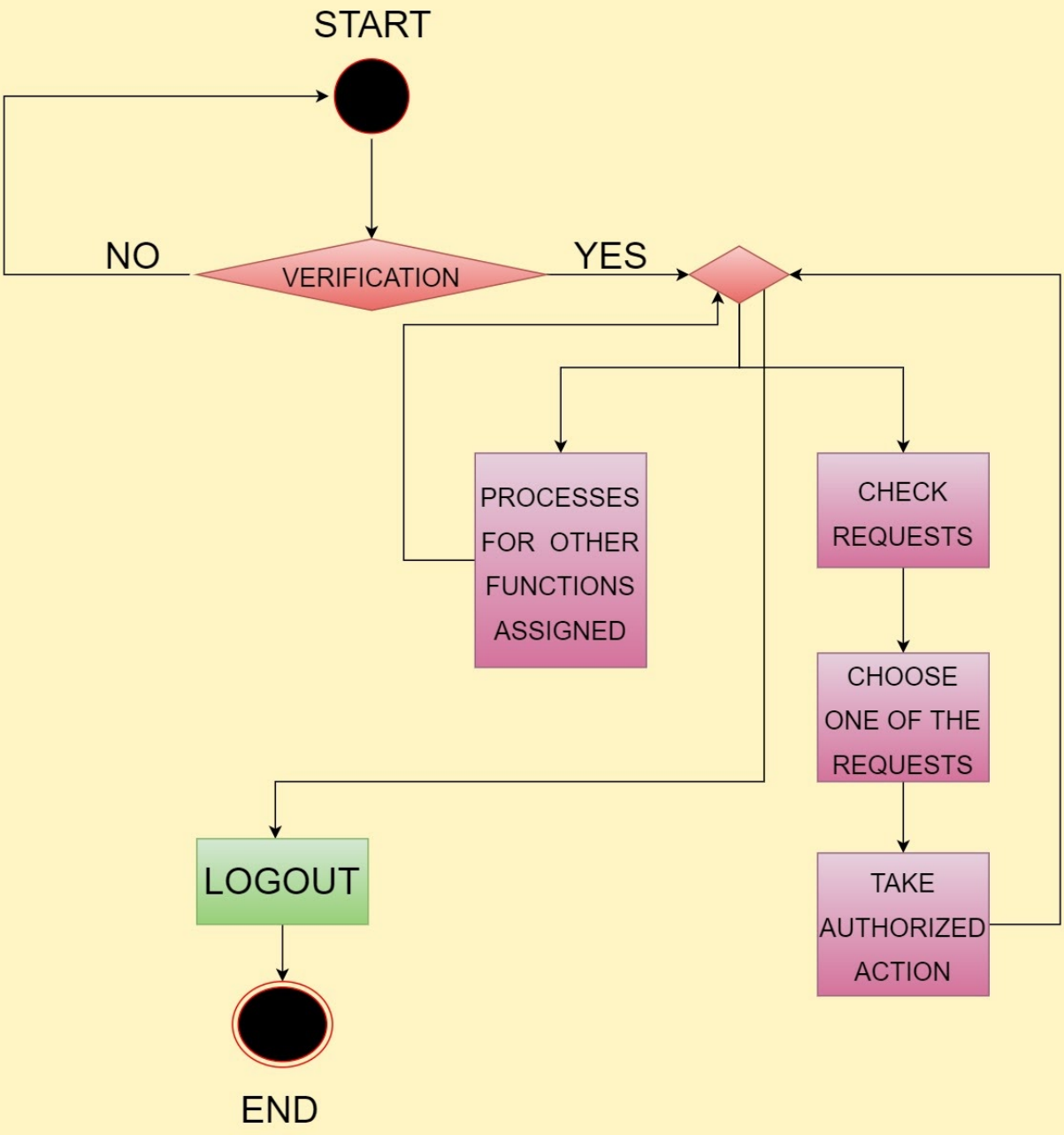
coordination. It is also suitable for modeling how a collection of use cases coordinate to represent business workflows

ACTIVITY DIAGRAMS FOR OUR MODEL

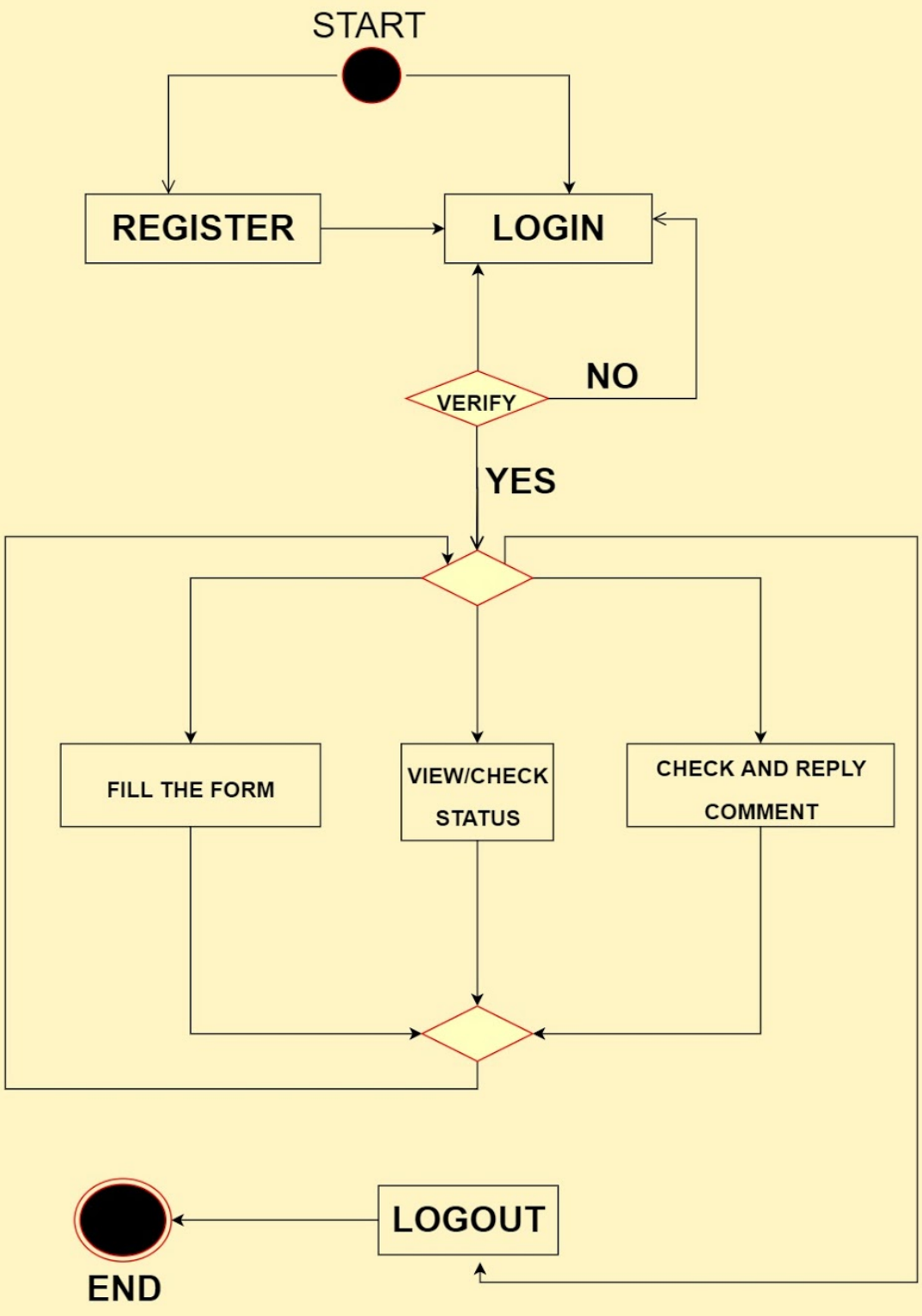
(We have made 3 activity diagrams)



ACTIVITY DIAGRAM MODERATOR



ACTIVITY DIAGRAM
USER



Explanation:

For above activity diagrams:

1) In this activity diagram admin starts with a login if he got his verification done then he can configure the workflow system and after that he log out.
2) In the second activity diagram moderator first logged in then he uses various options available to him uses different processes and checks the request and takes authorised action , after that he logged out and activity ends.
3) In the third activity diagram we show the activity of a simple user where he first registers then logs in and uses different functionalities available to him and logs out to end the activity.

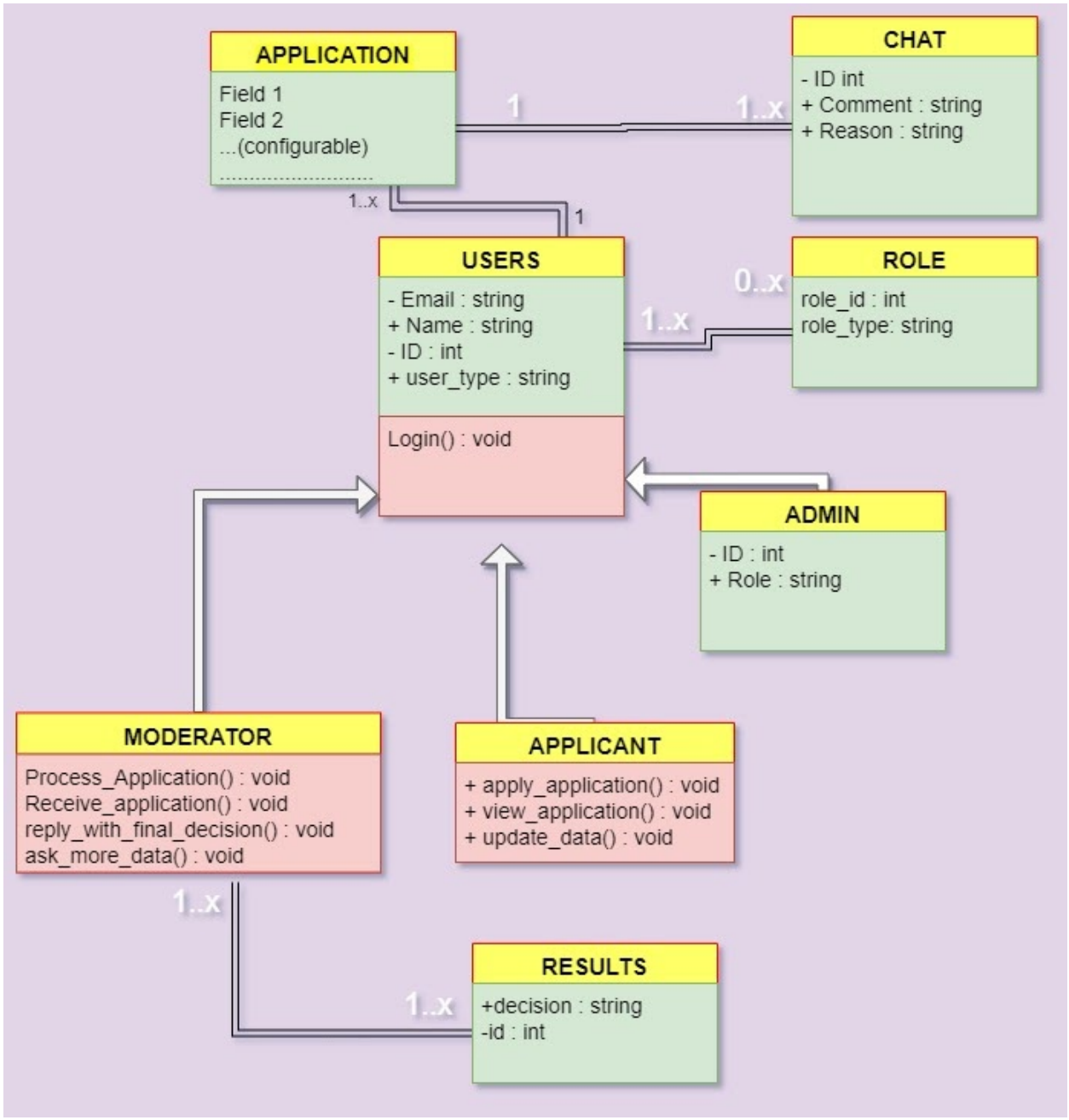
Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is **a type of static structure diagram** that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagrams:

1. Shows the static structure of classifiers in a system

2. Diagram provides a basic notation for other structure diagrams prescribed by UML
3. Helpful for developers and other team members too
4. Business Analysts can use class diagrams to model systems from a business perspective.



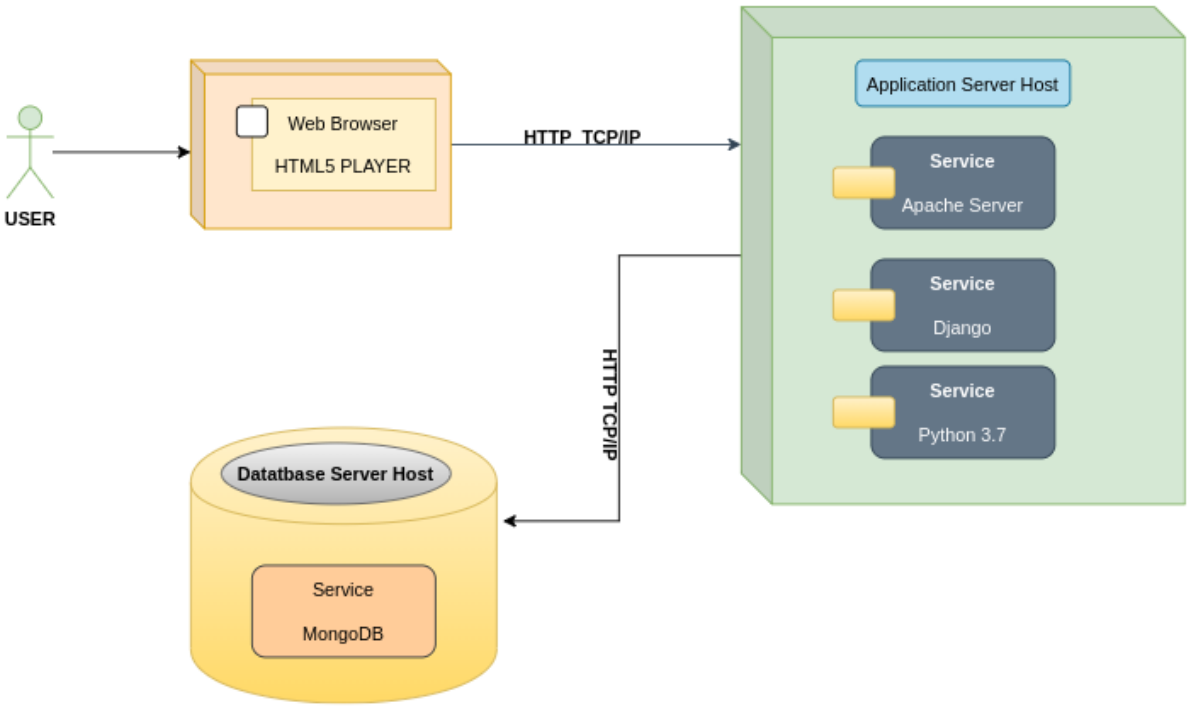
Explanation:

The above figure describes the major classes that will be created in the system and the types of data associated with it. The diagram also displays the type of relation present between all the classes and the necessary data required to a class. The data variables mentioned in the classes can complete most of the generalized tasks except some hidden data variables used inside the system.

Deployment Diagram:

A UML deployment diagram is a diagram that shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams are a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often used to model the static deployment view of a system (topology of the hardware). The deployment diagram is mainly used to describe the physical components where the software components are deployed. It also describes how these components interact with each other through protocols. Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering. Purpose of Deployment Diagrams

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.
- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes



Design Decisions :

TECHNOLOGY USED/ DESIGN CONCEPTS USED	TECHNOLOGIES CONSIDERED/ ALTERNATE DESIGN CONCEPTS	REASONS.WHY?
MongoDB DATABASE ● Pros: ○ As it is a NoSQL database it is flexible. ○ It supports horizontal Scalability. ○ It gives high speed as it is a document-oriented database. ● Cons: ○ It has high memory use. ○ The document size is limited to 16MB.	RELATIONAL DATABASE ● Pros: ○ The redundancy of data can be reduced much. ○ I can execute custom queries. ○ Supports all types of join operations. ● Cons: ○ Fixed schema. ○ Null values occupy extra space then needed and cause errors in the execution of queries.	● We are going to have dynamic data corresponding to different users which may result in null values for many columns if used with a fixed schema. ● We are familiar with the MongoDB database among all the NoSQL databases.
DJANGO FRAMEWORK ● Pros: ○ It is implemented in Python and easy to use. ○ It is secure and provides better CDN connectivity and content	FLASK ● Pros: ○ It is highly flexible. ○ It is a micro framework so performance is better.	● Django is based on python which is familiar to us. ● Some of our team members are acquainted with Django.

management. ● Cons: ○ Django is monolithic in nature. ○ Uses Regular expressions for URLs.	○ It supports scalability and modularity ● Cons: ○ It doesn't have more tools and libraries to support developers. ○ It is not standardized and hard to debug and understand others' code. .NET ● Pros: ○ It is simple and easy to maintain. ○ It is Object-oriented and consistent. ● Cons: ○ Limited Object Relational Support. ○ Slower than native code.	● Security is a must for Institutions using business workflows which is better implemented through Django.
SERVER SIDE LANGUAGE:- PYTHON ● Pros: ○ It has many inbuilt libraries. ○ Easy to use / clear syntax. ○ Supports various programming paradigms. ● Cons: ○ Slow to run ○ It might not be very	JAVA ● Pros: ○ It is Reliable ○ Ridiculously Robust ○ It is platform Independent. ● Cons: ○ It lacks the most modern features. ○ Slow and heavy ○ Confusing	● It is easy to use and our group members are Familiar with it. ● As it is a widespread language, it is easy to use/improve as developers can easily be found to further work on

future proof. ○ The process of shipping/distributing software is really complicated.	mis-features. PHP ● Pros: ○ Lots of good packages. ○ It is a pretty fast language compared to Python. ● Cons: ○ Poorly designed language ○ Inconsistent function names.	the software.
Web Server:- Apache HTTP Server ● Pros: ○ Benefits from great documentation and integrated support. ○ It is regularly maintained and updated. ○ It is flexible ● Cons: ○ It requires a strict updating policy that needs to be done regularly. ○ Personalised protocols can make new bugs. So debuggers are required.	NODE JS ● Pros: ○ Easy to learn. ○ It is easily Scalable and highly Extensible. ○ Advantage of caching. ● Cons: ○ API is not stable ○ It does not have a strong library support system.	● Apache HTTP server services more than 50% of the Websites, so it is easy to find its documentation and developers. ● Source code of Apache is Free to use and no license is required.
Design decision:- Modularity in Workflow Management	Monolithic Code ● Pros:	● As the workflow engine is generic

layer(Logical Structure diagram) ● Pros: ○ Easy to debug. ○ Easy to add different functionalities as per the need of the Client. ○ Easy to update. • Cons: ○ Decrease in performance due to data exchange through different modules. ○ Excessive and/or poor modularization can lead to code that is difficult to read.	○ Speed is high as everything is in one code. ○ Straightforward code management. ● Cons: ○ Hard to debug. ○ Hard to update or change some functionalities.	in nature modularity helps in adding specific functionalities as per requirements of the client. ● Developers can easily implement specific logic in different modules without concerning themselves to know the entire code.