VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
WINTER SEMESGTER 2020-21

J COMPONENT REVIEW 2 & 3

SWE-2004: Software Architecture and Design
Course Faculty: Prof. Sree Dharinya S

Slot: C1

Topic: System for Biometric Authorization of Voters

By:
Sabrina Manickam 19MIS0137
Arpan Dubey 19MIS0156

SOFTWARE ARCHITECTURE AND DESIGN

REVIEW 2

ARCHITECTURE STYLES

ABSTARCT

The proposed new system will entail the following features: It will access the 'AADHAR

issued by UIDAI' database for details of citizens. Any changes made to the AADHAR Database such as updation of voters address will be taken into account on Election process also.

Authentication of user will be enabled by biometric scanning of fingerprint and iris. The system will match the biometric information of user with biometric information stored in the AADHAR database.

PIPES AND FILTER

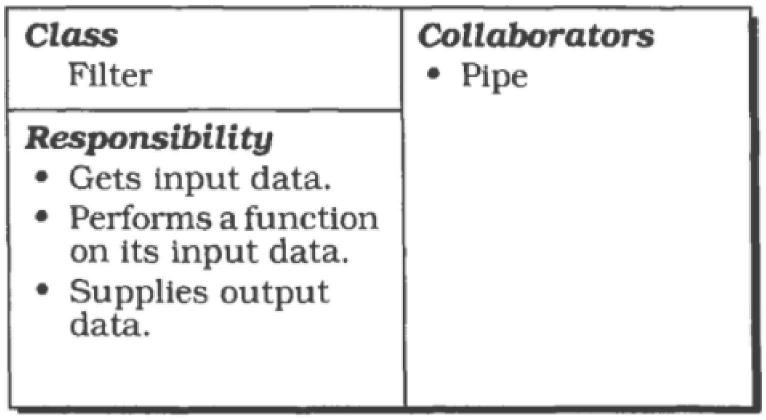
1. COMPONENTS:

FILTERS:

Filter components are the processing units of the pipeline. A filter enriches, refines or transforms its input data. It enriches data by computing and adding information, refines data by concentrating or extracting information, and transforms data by delivering the data in some other representation. A concrete filter implementation may combine all three basic principles. The activity of a filter can be triggered by several events:

- The subsequent pipeline element pulls output data from the filter.

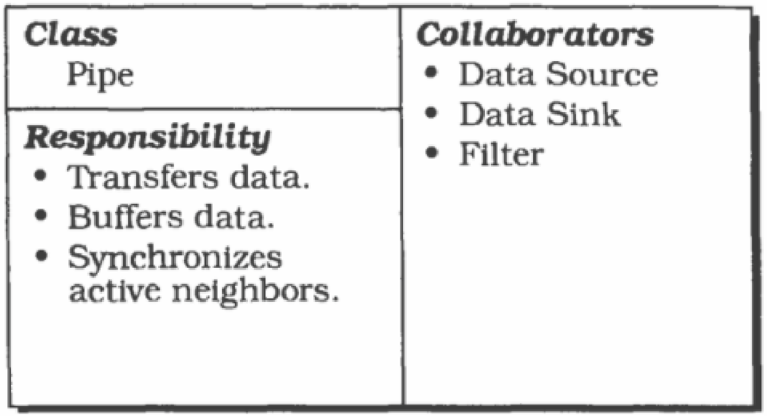
- The previous pipeline element pushes new input data to the filter.
- The filter is active in a loop, pulling its input from and pushing its output down the pipeline.



Class Filter	***Collaborators***
Responsibility • Gets input data. • Performs a function on its input data. • Supplies output data.	• Pipe

PIPES:

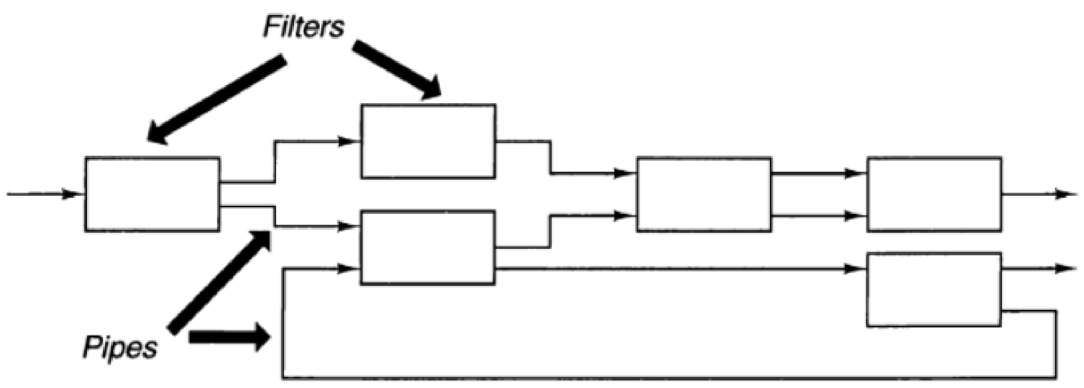
Pipes denote the connections between filters, between the data source and the first filter, and between the last filter and the data sink. If two active components are joined, the pipe synchronises them with a FIFO buffer. If activity is controlled by one of the adjacent filters, the pipe can be implemented by a direct call from the active to the passive component - direct calls make filter recombination harder, however.



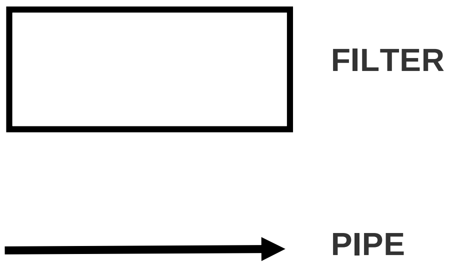
Class Pipe	***Collaborators***
Responsibility • Transfers data. • Buffers data. • Synchronizes active neighbors.	• Data Source • Data Sink • Filter

2. TOPOLOGY

Pipes connect filter output ports to filter input ports. Specializations of the style may restrict the association of components to an acyclic graph or a linear sequence.



3. SEMANTICS

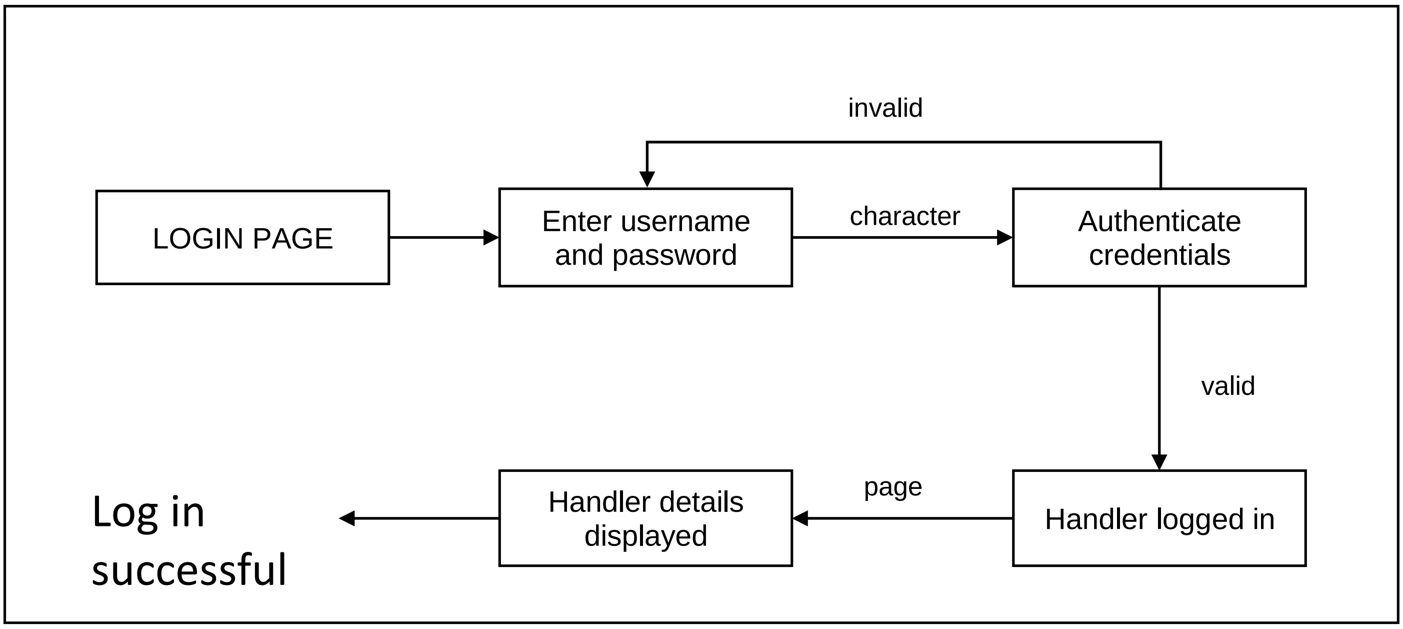


4. DESCRIPTON

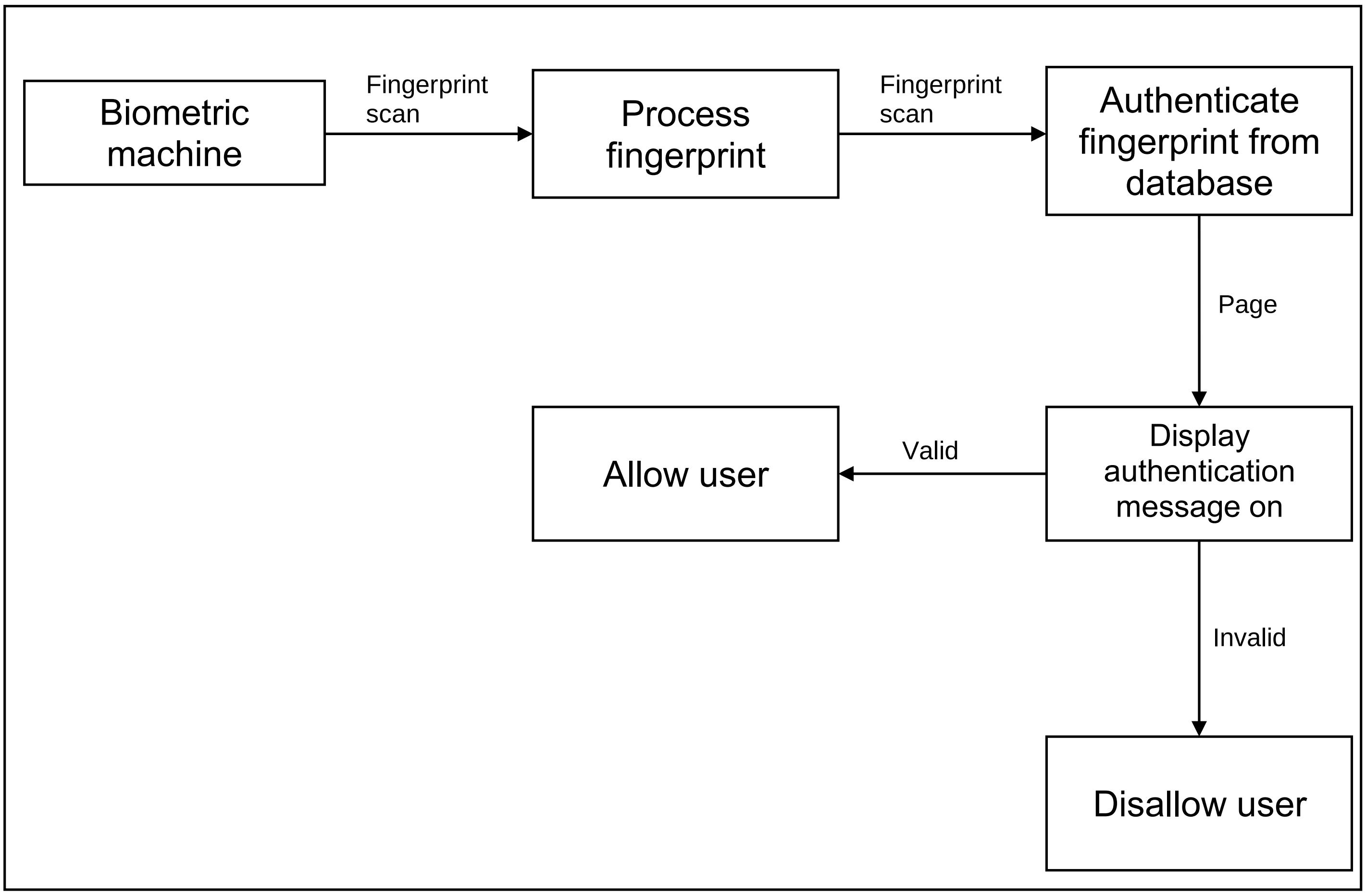
In a pipe and filter style each component has a set of inputs and a set of outputs. A component reads streams of data on its inputs and produces streams of data on its outputs, delivering a complete instance of the result in a standard order. This is usually accomplished by applying a local transformation to the input streams and computing incrementally so output begins before input is consumed. Hence components are termed “filters”. The connectors of this style serve as conduits for the streams, transmitting outputs of one filter to inputs of another. Hence the connectors are termed “pipes”.

5. DIAGRAMMATIC ILLUSTRATION

- **BIOMETRIC HANDLER LOGIN**



- **BIOMETRIC AUTHENTICATION OF VOTER**



6. ADVANTAGES

The Pipes and Filters architectural style comes with a number of benefits:

- No intermediate files necessary (but possible).
- Flexibility by filter exchange.
- Flexibility by recombination. Allows to create new processing pipelines by rearranging filters or by adding new ones.
- Reuse of filter components.
- Rapid prototyping of pipelines from existing filters.
- Efficiency by parallel processing.
- Allow the implementer to understand the overall input/output behaviour of a system as a simple composition of the behaviours of the individual filters.
- Encourage and support re-use. Any two filters can be connected, provided they agree on the data that are being transmitted.
- Easy to maintain and enhanced as new filters can be added to existing systems and old filters can be replaced by improved ones.
- Allow for certain kinds of specialised analysis, such as throughput and deadlock analysis.
- Improves testability due to the ability to test each component in isolation through their clearly defined (often very narrow) interface.
- Overall behaviour is a simple composition of behavior of individual filters.
- Reuse - any two filters can be connected if they agree on the data type of the data transmitted.
- Ease of maintenance - filters can be added or replaced: maintenance focusing on components, not on control
- Prototyping: easy construction of complex applications
- Potential for parallelism - independent filters implemented as separate tasks, consuming and producing data incrementally. (this is particularly the case in Computer graphics pipelines)
- Easy to understand (once the components are understood)
- Filters stand alone and can be treated as black boxes. This isolation of functionality helps to ensure quality attributes such as information hiding, low coupling, modifiability, and reuse.
- Pipes and filters can be hierarchically composed. Higher order filters can be created from any combination of lower order pipes and filters.
- The construction of the pipe and filter sequence can often be delayed until run time (late binding). This permits a controller component to tailor a process based on the current state of the application.

7. DISADVANTAGES

- Sharing state information is expensive, inflexible or even prohibitively dangerous. If the processing stages need to share a large amount of global data, applying the Pipes and Filters pattern is either inefficient or does not provide the full benefits of the pattern.
- Efficiency gain by parallel processing is often an illusion for the following reasons:

  Cost for transferring data between filters may be relatively high compared to the cost of the computation carried out by a single filter. This is especially true for small filter components or pipelines using network connections.

  Some filters consume all their input before producing any output, either because the task, such as sorting, requires it or because the filter is badly coded, for example by not using incremental processing when the application allows it.

  Synchronisation overhead might result in even reduced performance.

  The slowest filter determines the speed of the whole pipeline.
- Data transformation can cause a lot of overhead.
- Error handling is difficult. A common approach is to restart the pipeline in case of an error or exception.
- Not very well suited for handling interactive applications due to their transformational character.
- Depending on the implementation, they may force a lowest common denominator on data transmission, resulting in added work for each filter to parse and unparse its data, which can lead both to loss of performance and increased complexity writing the filters themselves.
- Can degenerate to ‘batch processing’: the filter processes all of its data before passing on (rather than incrementally).
- Sharing global data is expensive or limiting.
- Error handling is Achilles heel, for ex.: a pipeline has consumed three quarters of its input and produced half of its output, and some intermediate filter crashes! Generally restart pipeline.
- Implementation may force lowest common denominator on data transmission. For ex. in Unix pipes: everything is ASCII char.
- Transformational style does not support user interaction.
- Synchronization may be complicated (filters with more than one input).
- If a filter can not produce any output until it has received all of its input, the filter will require a buffer of unlimited size. If fixed size buffers are used, the system could deadlock. A sort filter has this problem.

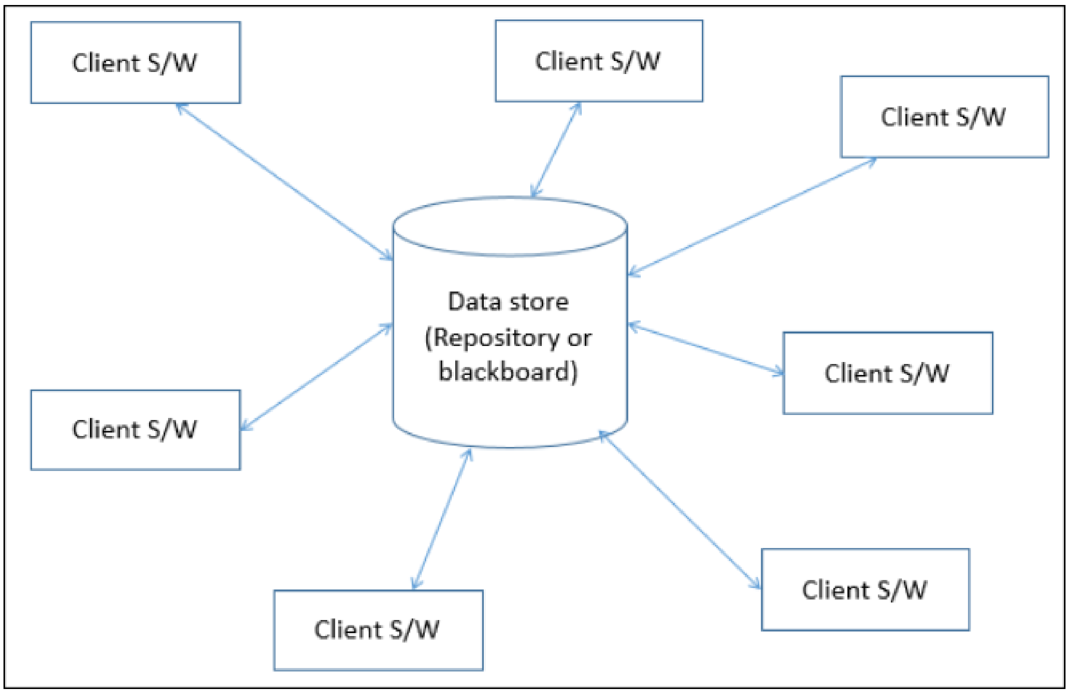
REPOSITORY ARCHITECTURE

1. COMPONENTS

There are two types of components −

- A **central data** structure or data store or data repository, which is responsible for providing permanent data storage. It represents the current state.
- A **data accessor** or a collection of independent components that operate on the central data store, perform computations, and might put back the results.

2. TOPOLOGY



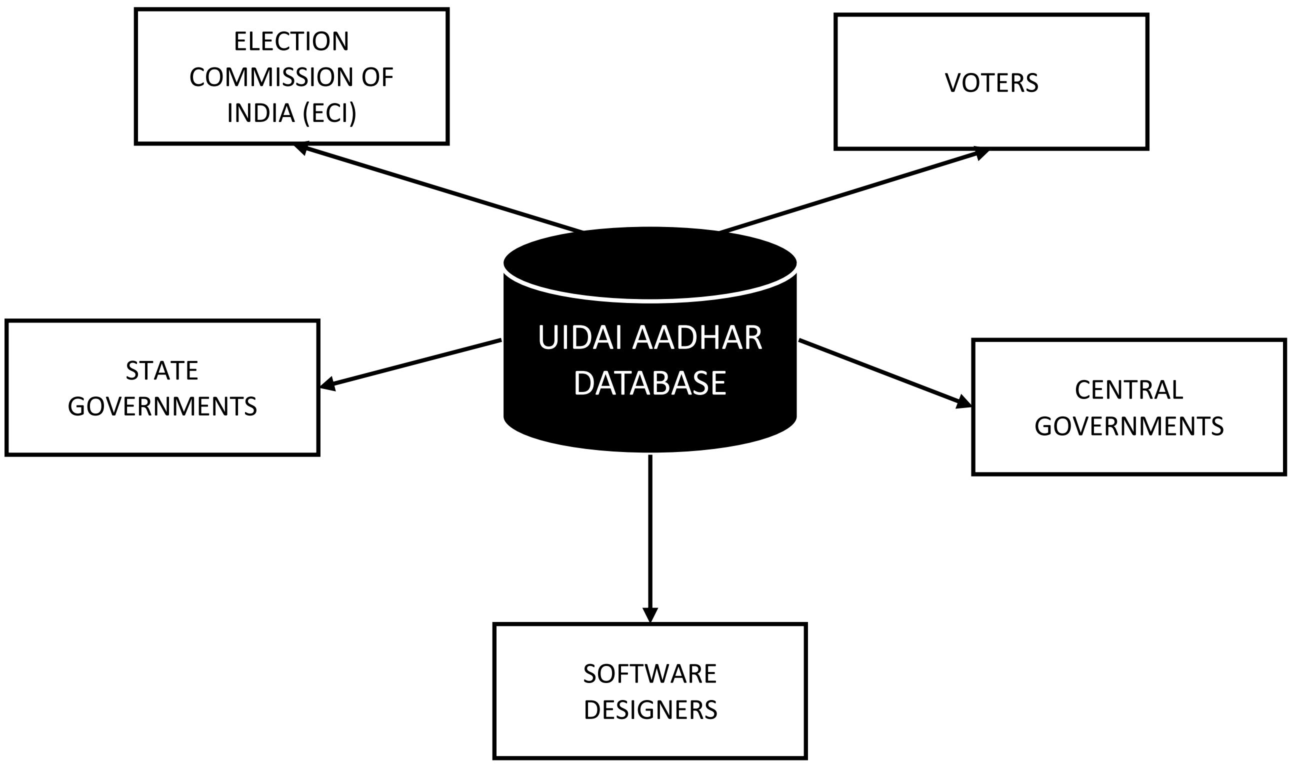
3. SEMANTICS

- **Repository:** a central location in which data is stored and managed.
- **Client:** a desktop computer or workstation that is capable of obtaining information and applications from a server.

4. DESCRIPTION

In Repository Architecture Style, the data store is passive and the clients (software components or agents) of the data store are active, which control the logic flow. The participating components check the data-store for changes.The client sends a request to the system to perform actions (e.g. insert data).The computational processes are independent and triggered by incoming requests.If the types of transactions in an input stream of transactions trigger selection of processes to execute, then it is traditional database or repository architecture, or passive repository.

5. DIAGRAMMATIC ILLUSTRATION



6. ADVANTAGES

- Provides data integrity, backup and restore features.
- Provides scalability and reusability of agents as they do not have direct communication with each other.
- Reduces overhead of transient data between software components.

7. DISADVANTAGES

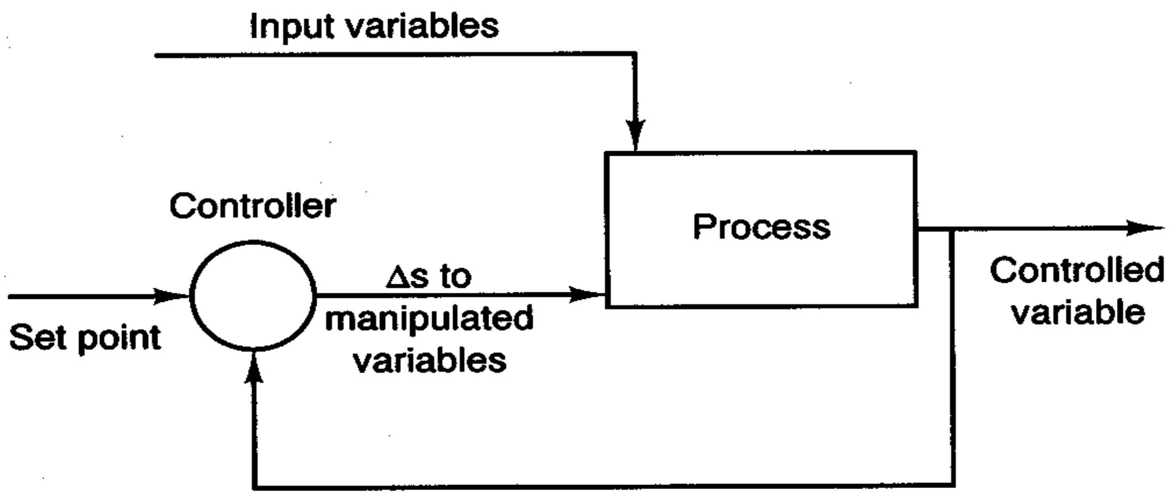
- It is more vulnerable to failure and data replication or duplication is possible.
- High dependency between data structure of data store and its agents.
- Changes in data structure highly affect the clients.
- Evolution of data is difficult and expensive.
- Cost of moving data on network for distributed data.

PROCESS CONTROL STYLE

1. COMPONENTS

- **Process variable:** Properties of the process that can be measured; several specific kinds are often distinguished.
- **Controlled variable:** Process variable whose value the system is intended to control.
- **Input variable:** Process variable that measures an input to the process**.**
- **Manipulated variable:** Process variable whose value can be changed by the controller.
- **Set point:** The desired value for a controlled variable.

2. TOPOLOGY



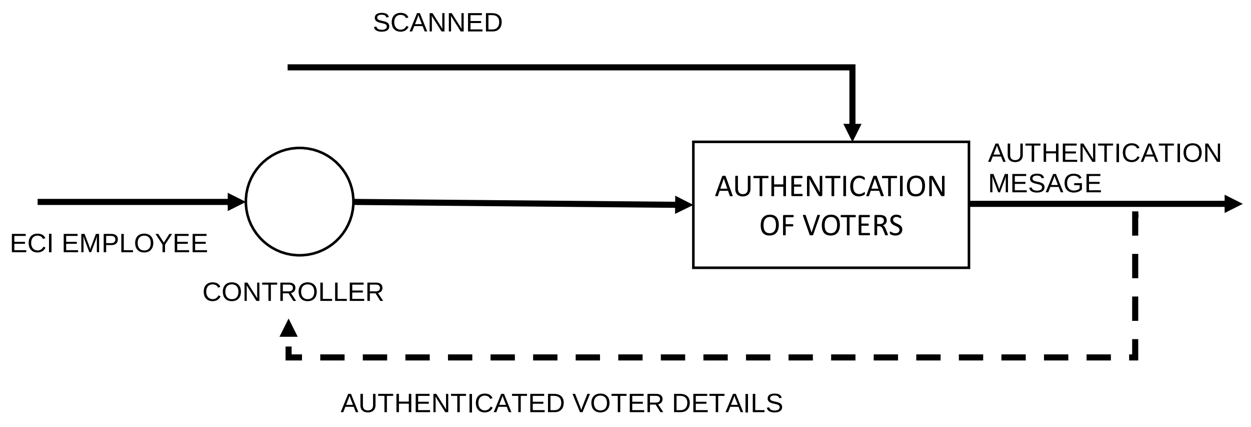
3. SEMANTICS

- **Process:** a series of actions or steps taken in order to achieve a particular end.
- **Controller:** regulates actions on processes

4. DESCRIPTION

It is a type of data flow architecture where data is neither batched sequential nor pipelined stream. The flow of data comes from a set of variables, which controls the execution of process. It decomposes the entire system into subsystems or modules and connects them.

5. DIAGRAMMATIC ILLUSTRATION



6. ADVANTAGES

- Well suited for the control of continuous processes particularly where the control algorithm is subject to change (possibly even at run-time).

7. DISADVANTAGES

- Not easily applicable when multiple interacting processes (and controls) are needed.

CLIENT SERVER ARCHITECTURE

1. COMPONENTS:

- Component types: client, which requests services of another component, and server, which provides services to other components.
- Connector types: request/reply, the asymmetric invocation of server's services by a client.

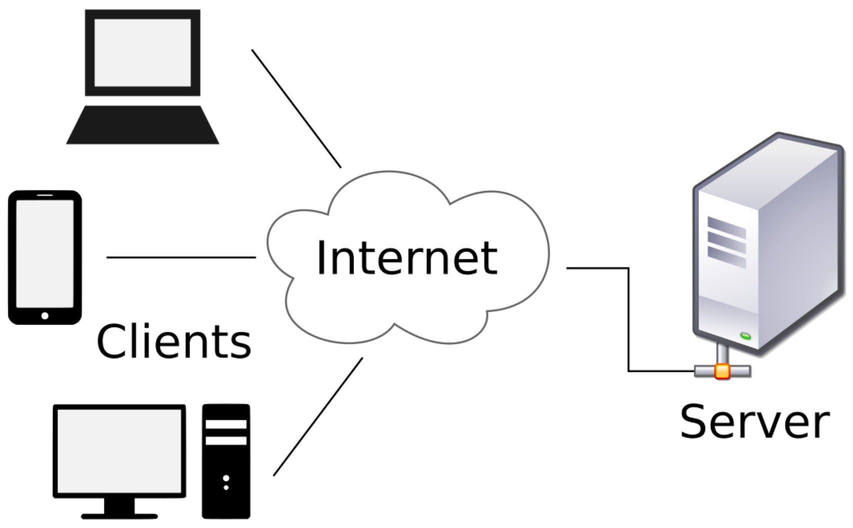
2. CONNECTORS

Attachment relation associate's clients with the request role of the connector and servers with the reply role of the connector and determines which services can be requested by which clients.

3. TOPOLOGY

In general, unrestricted. Specializations may impose restrictions:

- Numbers of attachments to a given port or role
- Allowed relationships among servers
- Tiers



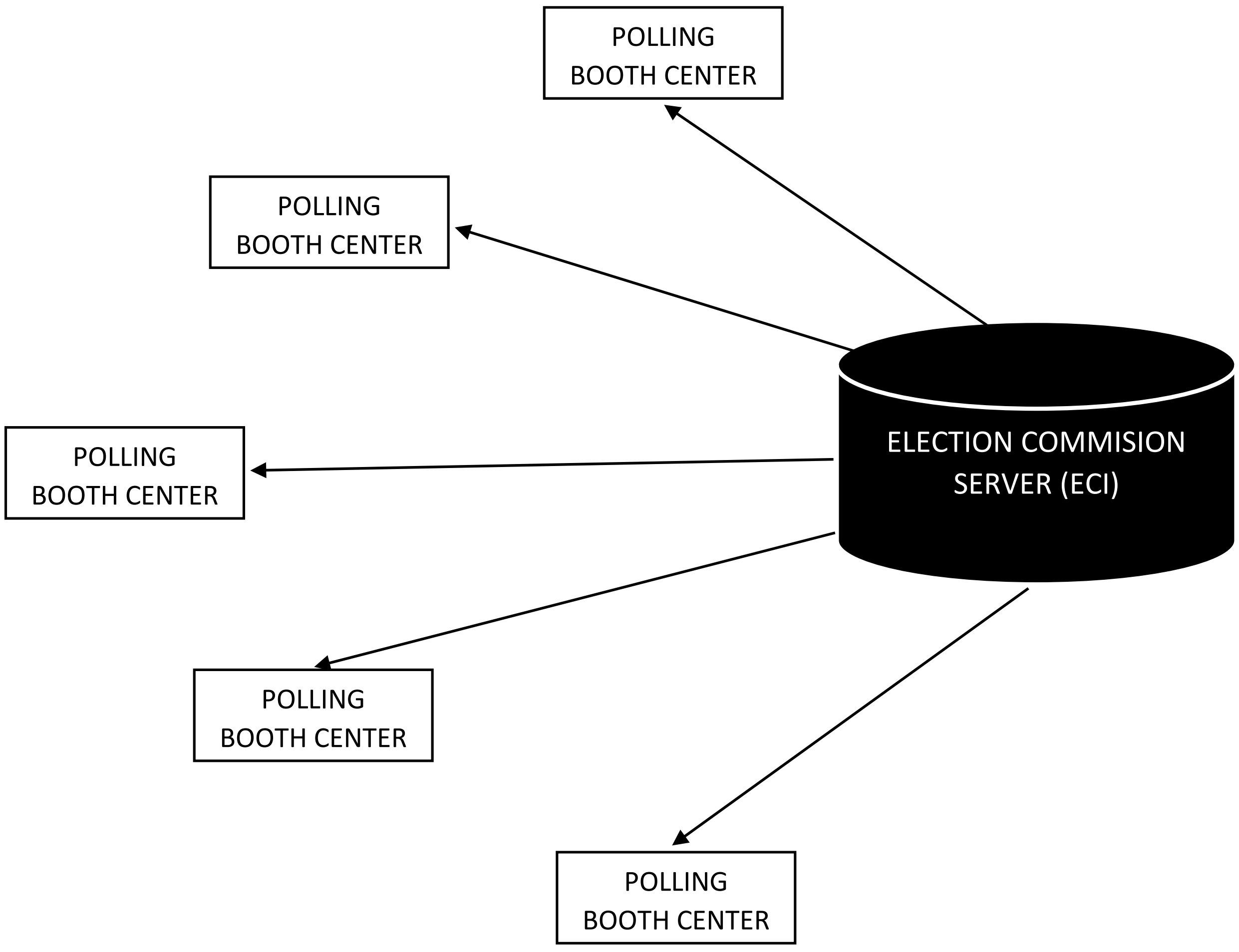
4. DESCRIPTION

In the client-server style of the C&C view type, components interact by requesting services of other components. The essence of this style is that communication is typically paired and initiated by the client. A request for service from a client is normally paired with the provision of that service. Servers in this style provide a set of services through one or more interfaces, and clients use zero or more services provided by other servers in the system. There may be one central server or several distributed ones.

Computational Model

Clients initiate activities, requesting services as needed from servers and waiting for the results of those requests.

5. DIAGRAMTTIC ILLUSTRATION



6. ADVANTAGES

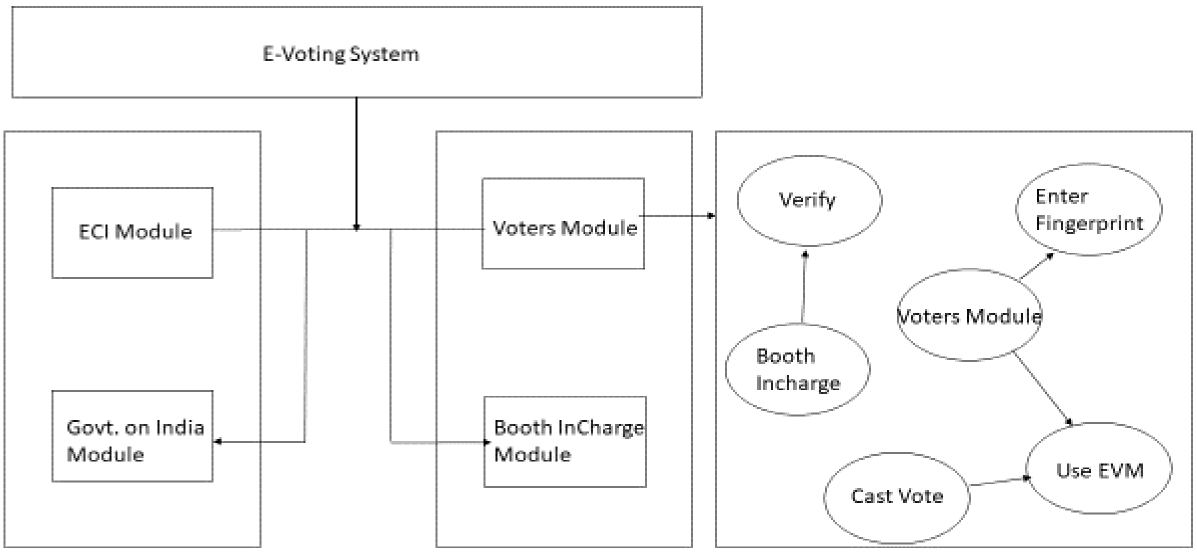
- The data is centralized within the system that is maintained in a single place.
- The model is efficient in delivering resources to the client and also requires low-cost maintenance.
- It is easy to manage, and the data can be easily delivered to the client.
- As the data is centralized, this system is more secure and serves added security to the data.
- Within this type of model, more clients and servers can be embedded into the server, which makes the performance outstanding and increases the model's overall flexibility.

7. DISADVANTAGES

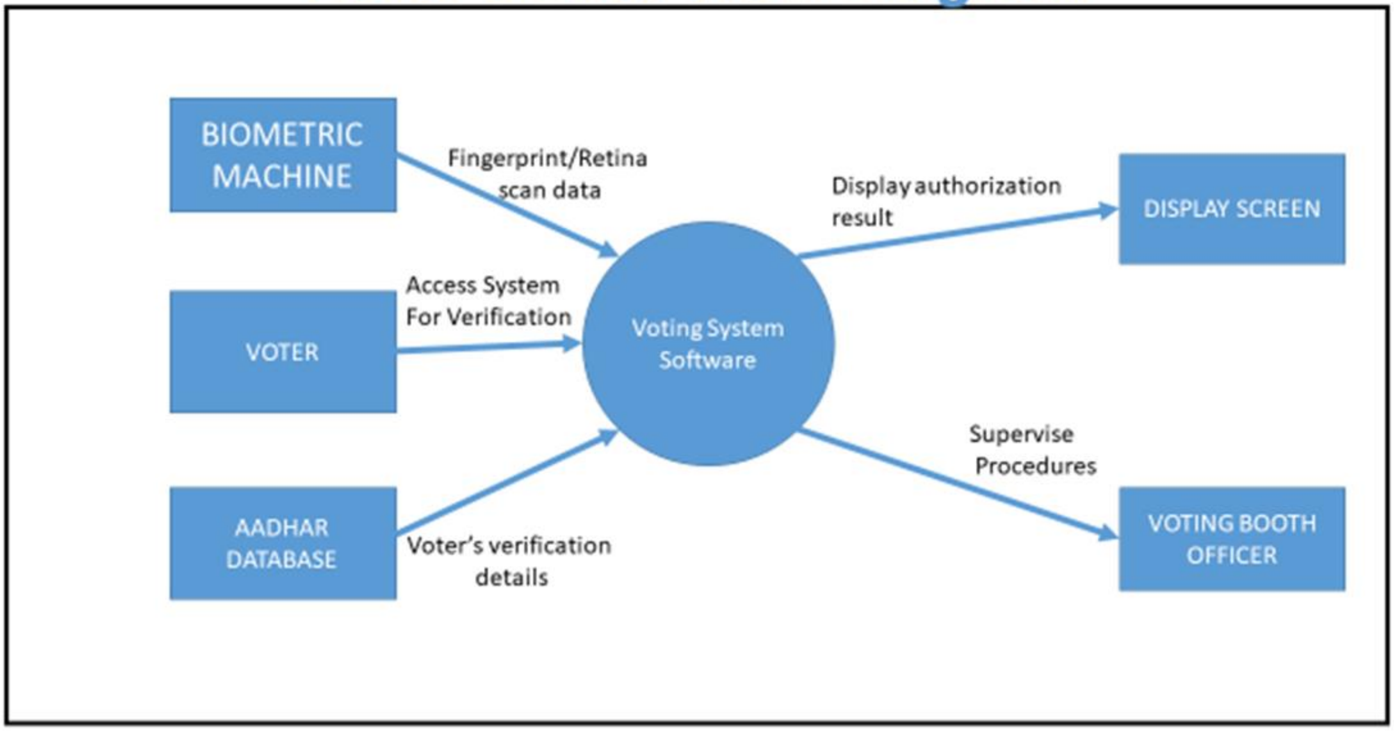
- Clients' systems can get a virus or any malicious scripts if any are running on the server.
- Extra security must be added so that the data does not get spoofed in between the transmission.
- The main problem can be server down. When the server is down, the client loses its connection and will not access the data.

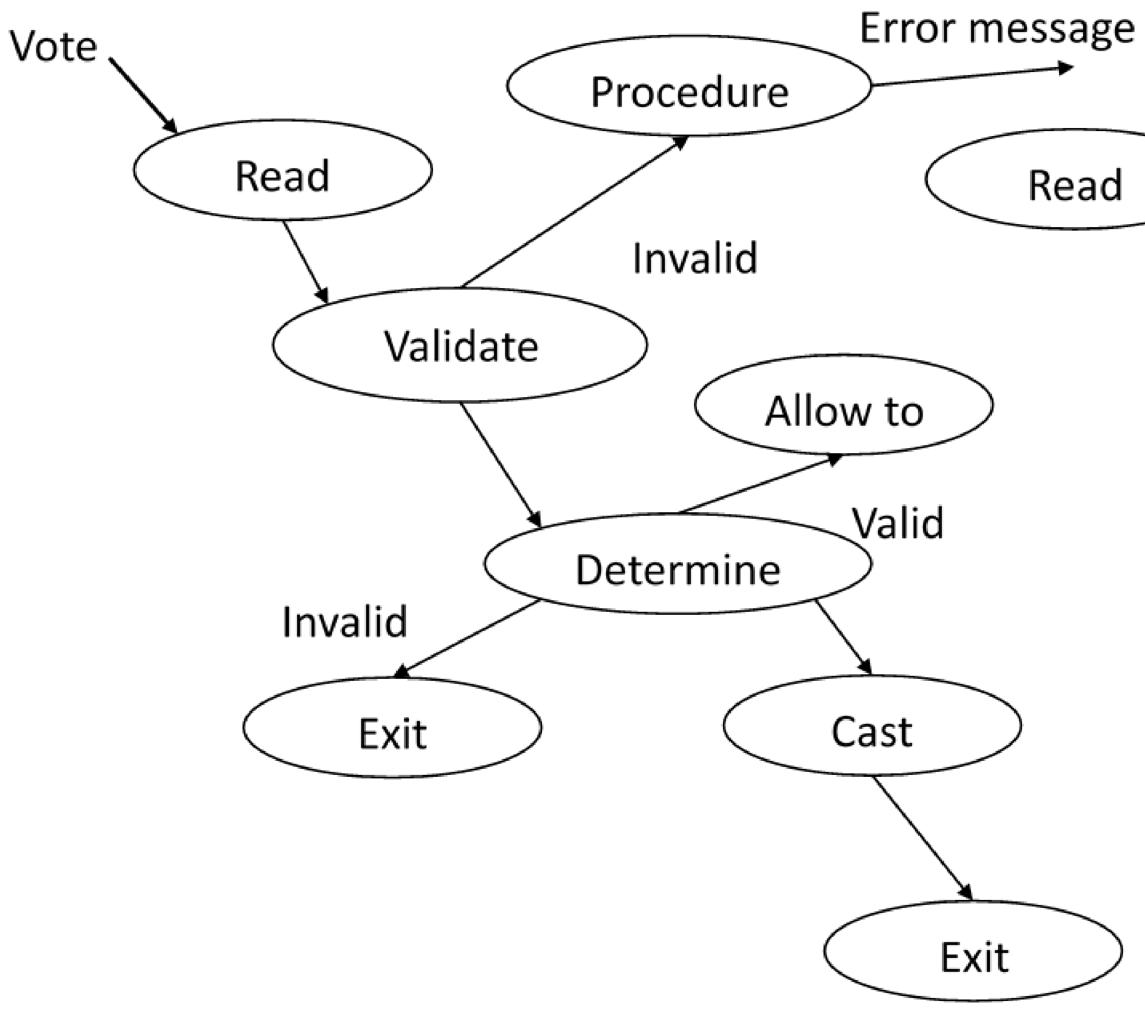
REVIEW 3

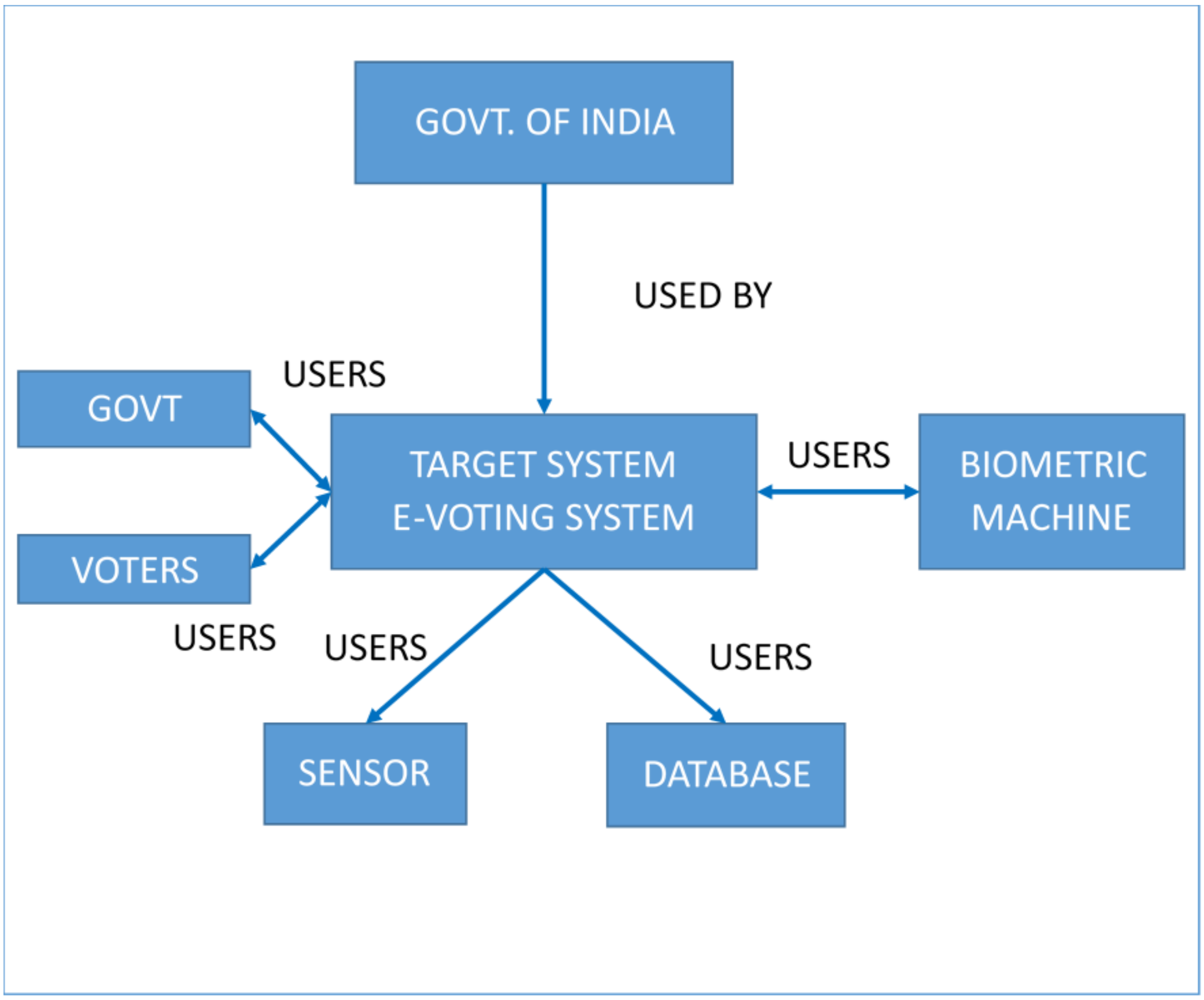
MAPPING OF ARCHITECTURE



Level 0 Data Flow Diagram

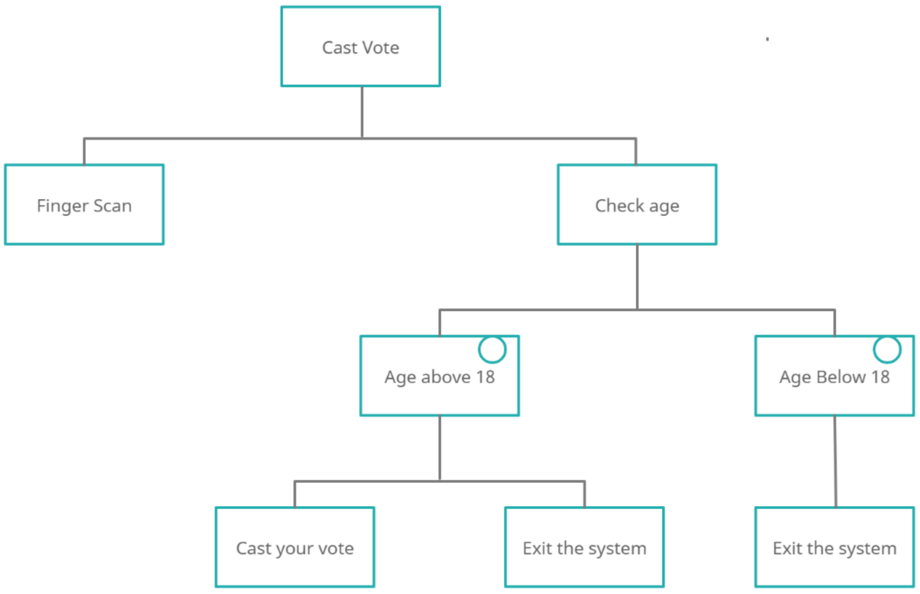

BIOMETRIC MACHINE
Fingerprint/Retina scan data
VOTER
Access System For Verification
AADHAR DATABASE
Voter's verification details
Voting System Software
Display authorization result
DISPLAY SCREEN
Supervise Procedures
VOTING BOOTH OFFICER


Vote
Read
Procedure
Error message
Read
Invalid
Validate
Allow to
Valid
Determine
Invalid
Exit
Cast
Exit


GOVT. OF INDIA
USED BY
USERS
GOVT
TARGET SYSTEM
E-VOTING SYSTEM
USERS
BIOMETRIC
MACHINE
VOTERS
USERS
USERS
USERS
SENSOR
DATABASE

JSP (NEEDED CORRECTION IN REVIEW 1)

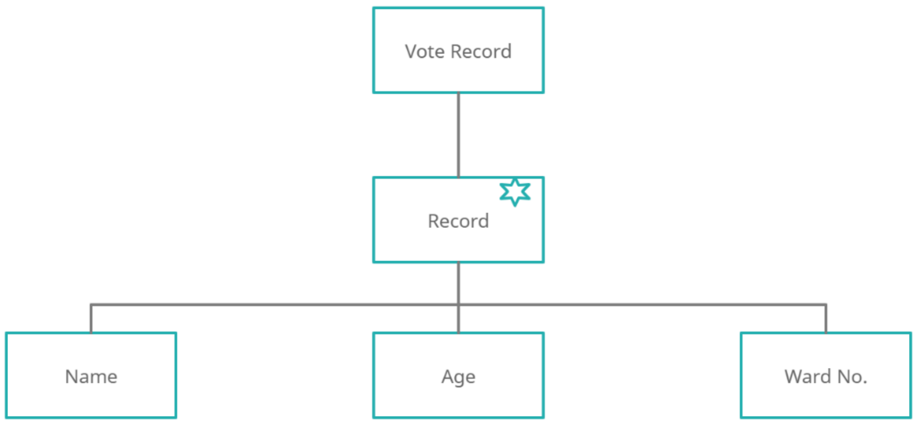
Input Data Stream



Level of Abstraction

- Record Fingerprint
- Check Age
    1)Valid
    2)Invalid
- Cast vote if age is valid

Output Data Stream



Level of Abstraction

• Details displayed at booth manager’s system

• Details include-:

1)Name

2)Age

3)Ward No

Operations of the program

1) Repeatedly take fingerprint and check authenticity.

2) Taking biometric data including scanning finger.

3) Authentication at booth manager’s system involves checking name, age and ward no.

4) Use of EVM to cast vote.

List the operations and allocate to program elements

Output

1)Name

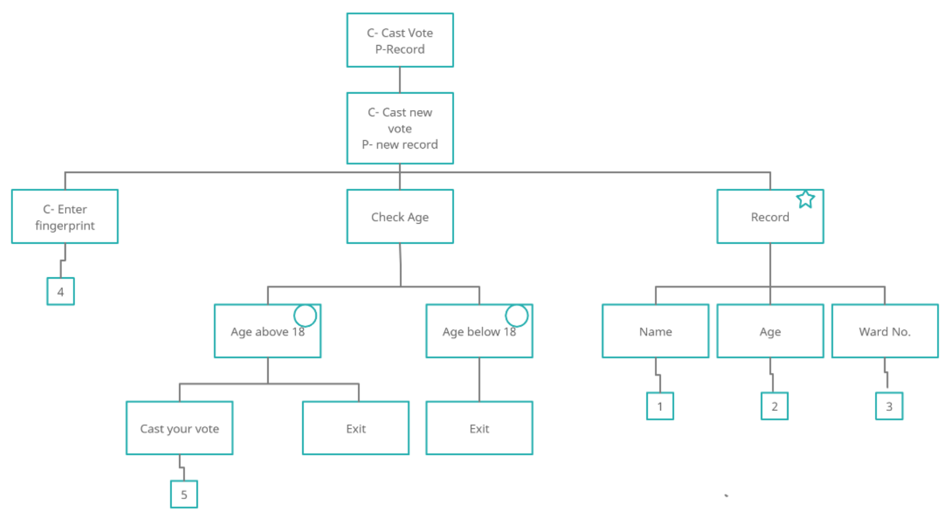
2)Age

3)Ward No.

Input

4)Obtain Finger Data

5)Cast Vote



Program to text without conditions

```
LOOP
   Get finger data
   Check age
   Allow to vote
END LOOP
```

Program to text with conditions

```
LOOP
    Get finger data
CASE OF TYPE
    Age above 18: Allow to vote
    Age below 18: Don't allow
END CASE
    Check name, age and ward no.
    Allow to vote
END LOOP
```