MIT World Peace University

Software Engineering and Testing
Second Year B. Tech, Semester 4

Component Diagrams and Deployment Diagrams

Assignment No. 7

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

May 2, 2023

Contents

1 Aim

Object Oriented Analysis and design using UML diagrams: Component diagram, deployment diagram using Open Source Tool.

2 Objectives

1. To learn the relationships and notions of Component diagram.
2. To learn the relationships and notions of Deployment diagram.

3 Problem Statement

Draw Component Diagram and Deployment Diagram for The Following Problem:

The Purpose of an Attandence Assistant App is to help reduce the time taken for recording the attendance of a classroom in a school or college. The app will be able to record the attendance of a class in a matter of a few Seconds with minimum Energy Expended. It will record data on cloud, and be accessible to all the Teachers.

The tasks we have to do are:

1. You will have to identify the main entities (objects) for this system.
2. You will have to find out the relationships between these objects.
3. You will have to find the necessary attributes and functions that need to be associated with each object to implement the functionality mentioned above.
4. You will make a final comprehensive diagram show and all objects and their relations along with their attributes and functions.

4 Theory

4.1 Component Diagram

4.1.1 What is a Component Diagram

A component diagram is a type of UML diagram that shows the physical components of a system and how they are interconnected. It is used to visualize, specify, and document the architecture and structure of a software system.

4.1.2 What is the use of a Component Diagram

1. *Identifying system components*: Component diagrams can be used to identify the various components of a system, including hardware, software, and other physical elements.
2. *Visualizing system architecture*: Component diagrams provide a visual representation of the system architecture, showing the relationships and dependencies between different components.

3. *Designing software systems*: Component diagrams can be used in software design to map out the different components of a system and their interactions.
4. *Testing and debugging*: Component diagrams can be used in testing and debugging to identify potential problems or bottlenecks in the system architecture.
5. *Communication*: Component diagrams are a useful tool for communicating system architecture and design to stakeholders and team members.

4.1.3 Elements of a Component Diagram

1. *Component*: A component is a physical element or module of the system, such as a software module, hardware component, or database.
2. *Interface*: An interface defines the way in which a component communicates with other components in the system.
3. *Connector*: A connector shows the relationships and dependencies between different components and interfaces.
4. *Port*: A port is a specific point of connection between a component and an interface.
5. *Dependency*: A dependency shows the relationship between two components, indicating that one component depends on another for its operation

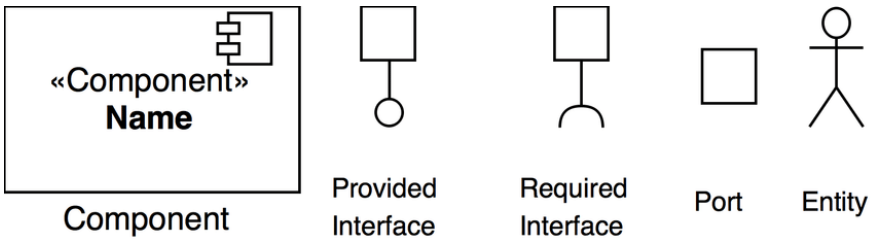

Figure 1: Elements of a Component Diagram

4.2 Deployment Diagram

4.2.1 What is a Deployment Diagram

A deployment diagram is a type of UML diagram that shows the physical deployment of software components on hardware nodes. It depicts the architecture of a system as it is deployed to different hardware configurations, such as servers, workstations, or mobile devices.

4.2.2 What is the use of a Deployment Diagram

The main use of a deployment diagram is to model and visualize the physical deployment of software components within a system. Some specific uses of deployment diagrams include:

1. *Understanding system architecture*: Deployment diagrams provide a high-level view of the system architecture, showing how software components are deployed on hardware nodes.

2. *Designing and planning deployment*: Deployment diagrams can be used to plan the deployment of a system, helping to identify potential issues and dependencies between different components.

3. *Communication*: Deployment diagrams are a useful tool for communicating system architecture and deployment plans to stakeholders and team members.

4. *Testing and debugging*: Deployment diagrams can be used in testing and debugging to identify potential issues or bottlenecks in the deployment architecture.

5. *Scaling and optimization*: Deployment diagrams can be used to identify opportunities for scaling and optimizing the system architecture for improved performance and efficiency.

4.2.3 Elements of a Deployment Diagram

1. *Node*: A node represents a physical device or server, such as a computer, printer, or mobile device.

2. *Component*: A component represents a software module or application that is deployed on a hardware node.

3. *Artifact*: An artifact is a physical element, such as a file or database, that is used by a software component.

4. *Deployment relationship*: A deployment relationship shows how components are deployed on nodes, including the mapping of artifacts to nodes.

5. *Communication path*: A communication path shows how nodes communicate with each other, including network connections, protocols, and other communication channels.

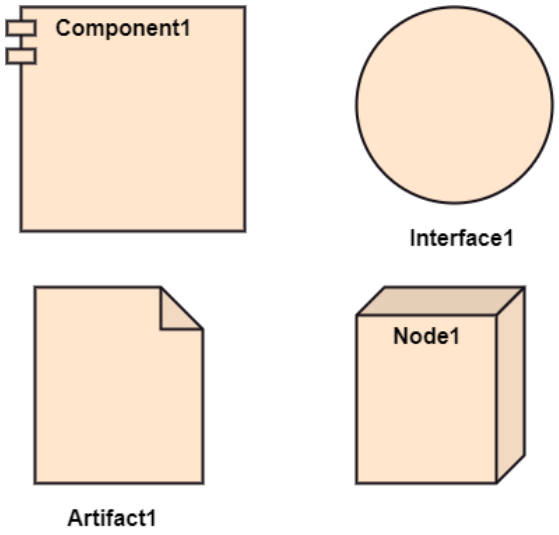

Figure 2: Elements of a Deployment Diagram

5 Diagrams

5.1 Component Diagram

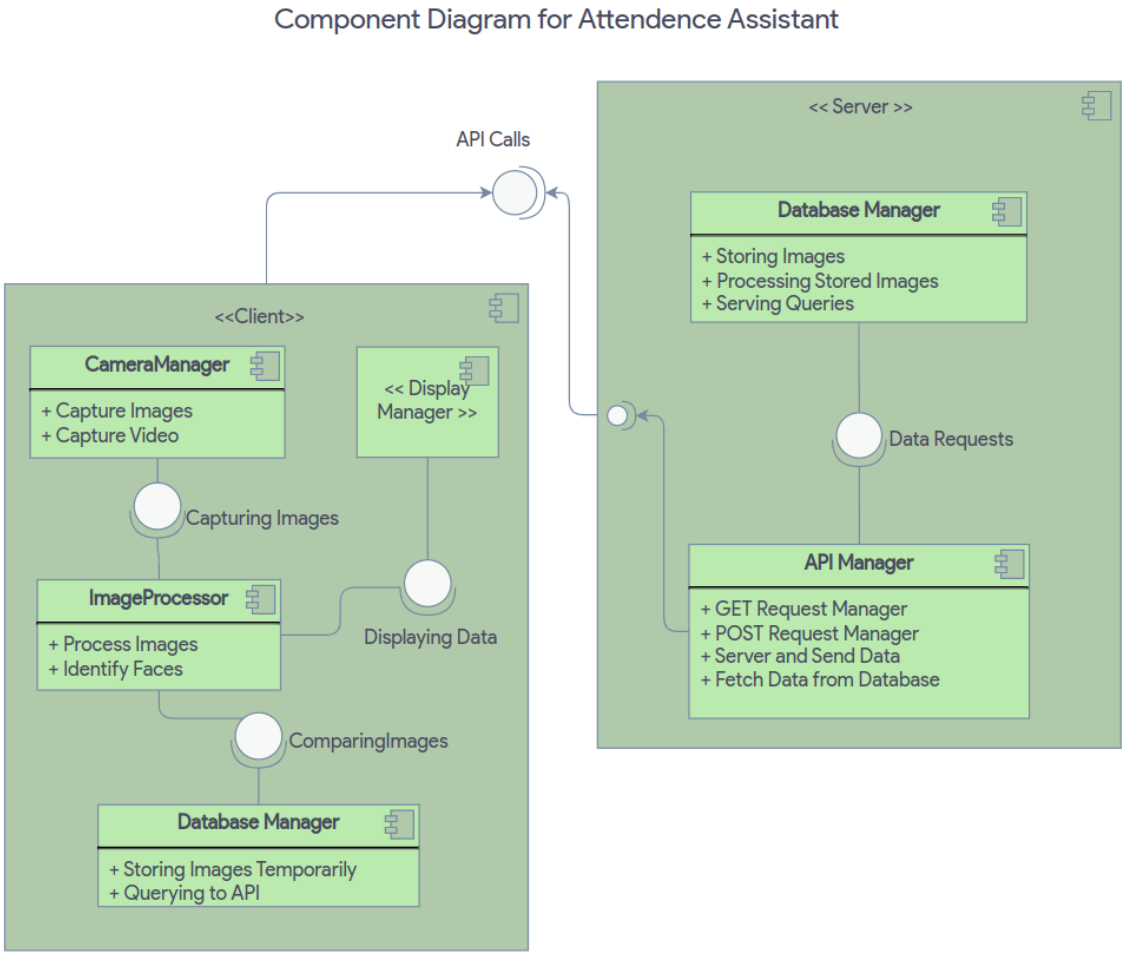

Figure 3: Use Case Diagram

5.2 Deployment Diagram

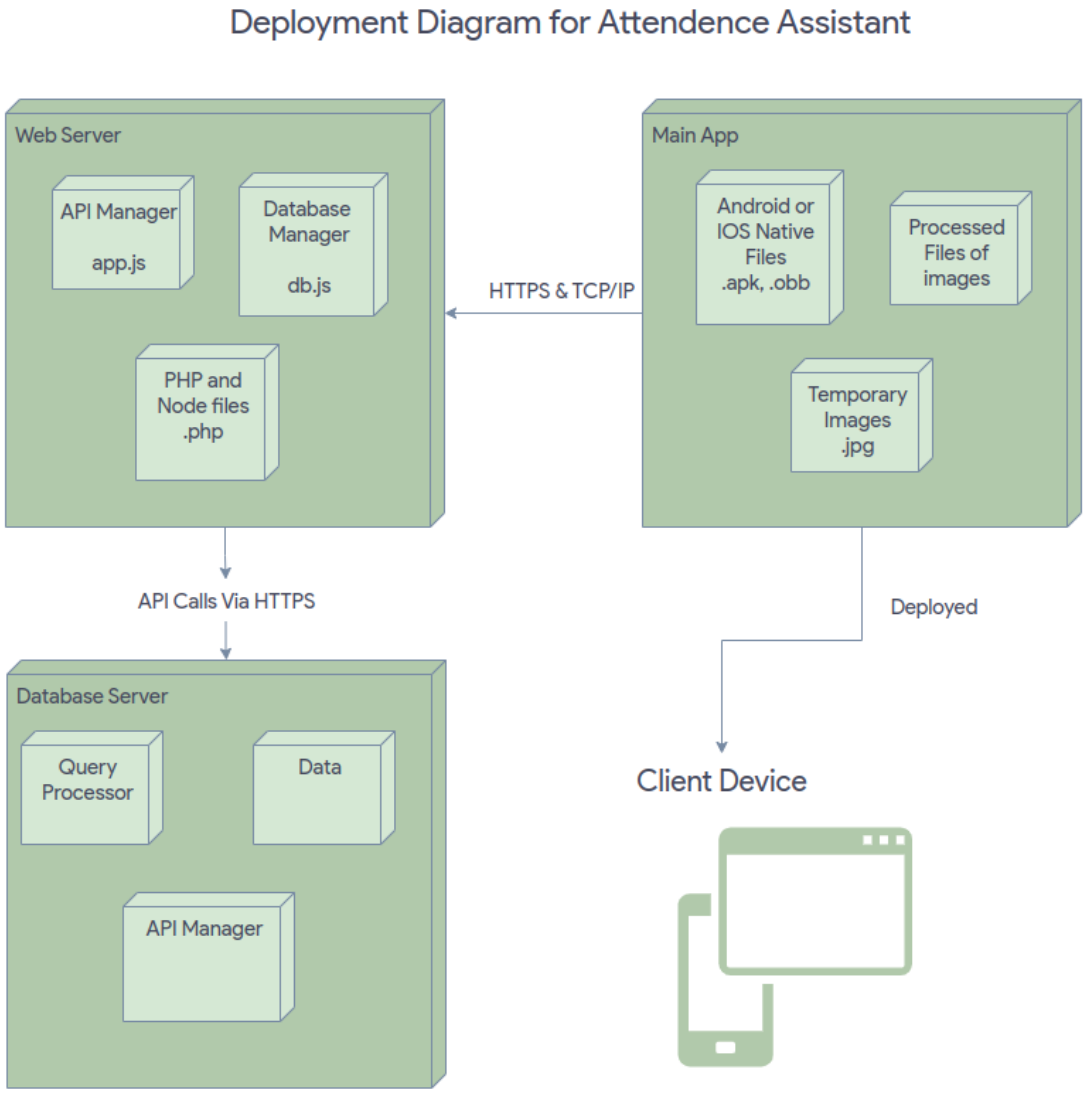

Figure 4: Deployment Diagram

6 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
External Programs for Diagrams : Draw.io

7 Conclusion

Thus, we learnt about Component diagrams and Deployment diagrams. We also learnt about the different types of diagrams and their uses in detail.

8 FAQ

1. **The term component is sometimes a difficult one to define. First provide a generic definition, and then provide more explicit definitions for object-oriented and traditional software. Finally, pick three programming languages with which you are familiar and illustrate how each defines a component.**

   A component is a modular, reusable, and independent part of a system that encapsulates a set of related functionality. It can be a physical or logical entity that can be easily integrated into other systems or applications.

   In object-oriented software, a component is typically a class or group of classes that provide a specific set of functionalities. It is designed to be easily integrated into other object-oriented systems and can be easily extended or modified.

   In traditional software, a component is typically a set of functions or procedures that provide a specific set of functionalities. It is designed to be easily integrated into other traditional systems and can be easily modified or replaced.

   Three programming languages that define a component are:

   - Java: In Java, a component is typically defined as a reusable software module that can be easily integrated into other Java applications. It is typically implemented as a Java class or set of classes that encapsulate a specific set of functionalities.
   - Python: In Python, a component is typically defined as a module or package that provides a specific set of functionalities. It is designed to be easily integrated into other Python applications and can be easily extended or modified.
   - C++: A component in C++ can be implemented as a class or set of classes that provide a specific set of functionalities, which can be easily integrated into other C++ applications. Additionally, C++ provides features like templates, namespaces, and libraries, which can be used to define and package reusable components for different purposes.

2. **What is a WebApp component?**

   A WebApp component is a modular and reusable part of a web application that encapsulates a specific set of functionalities. It can be a physical or logical entity that can be easily integrated into other web applications or systems. WebApp components typically include web pages, scripts, databases, and other resources that provide a specific set of functionalities within a web application.

3. **State the importance of deployment diagram.**
   The importance of a deployment diagram includes:

   - Providing a high-level view of the physical deployment of software components within a system.
   - Helping to identify potential issues and dependencies between different components.
   - Aiding in system design, planning, and communication.

- Identifying opportunities for scaling and optimizing the system architecture.
- Assisting in testing and debugging to identify potential issues or bottlenecks in the deployment architecture.