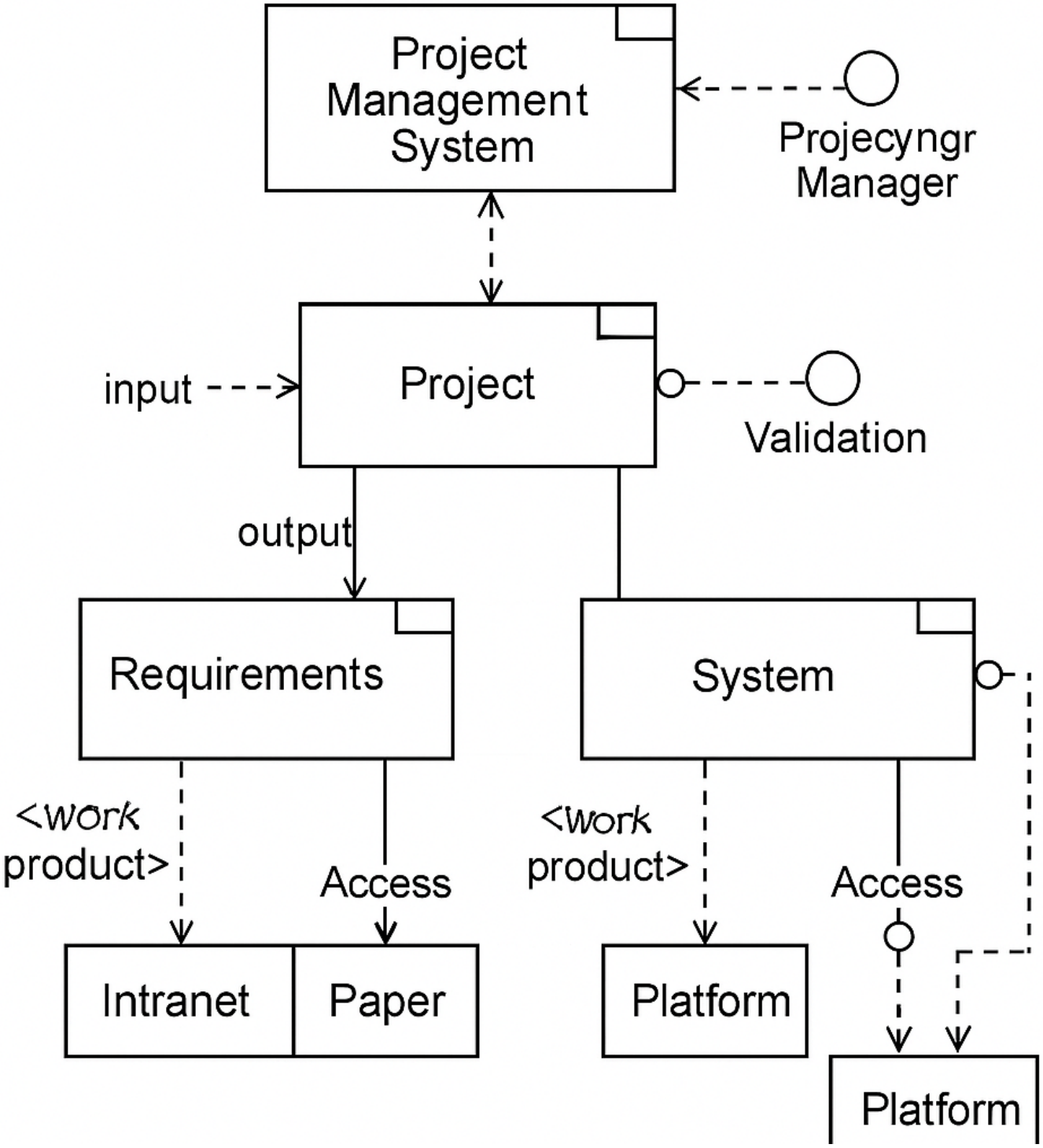
Q12.

Requirements:

Project Management System

A project manager uses the project management system to manage a project. The project manager leads a team to execute the project within the project's start and end dates. Once a project is created in the project management system, a manager may initiate and later terminate the project due to its completion or for some other reason. As input, a project uses requirements. As output, a project produces a system (or part of a system). The requirements and system are work products: things that are created, used, updated, and elaborated on throughout a project. Every work product has a description, is of some percent complete throughout the effort, and may be validated. However, validation is dependent on the type of work product. For example, the requirements are validated with users in workshops, and the system is validated by being tested against the requirements. Furthermore, requirements may be published using various types of media, including on an intranet or in paper form; and systems may be deployed onto specific platforms.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○

Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ◉ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Some text garbling ('projecyngr', 'work product'), strange use of interfaces, inputs and output represented as components

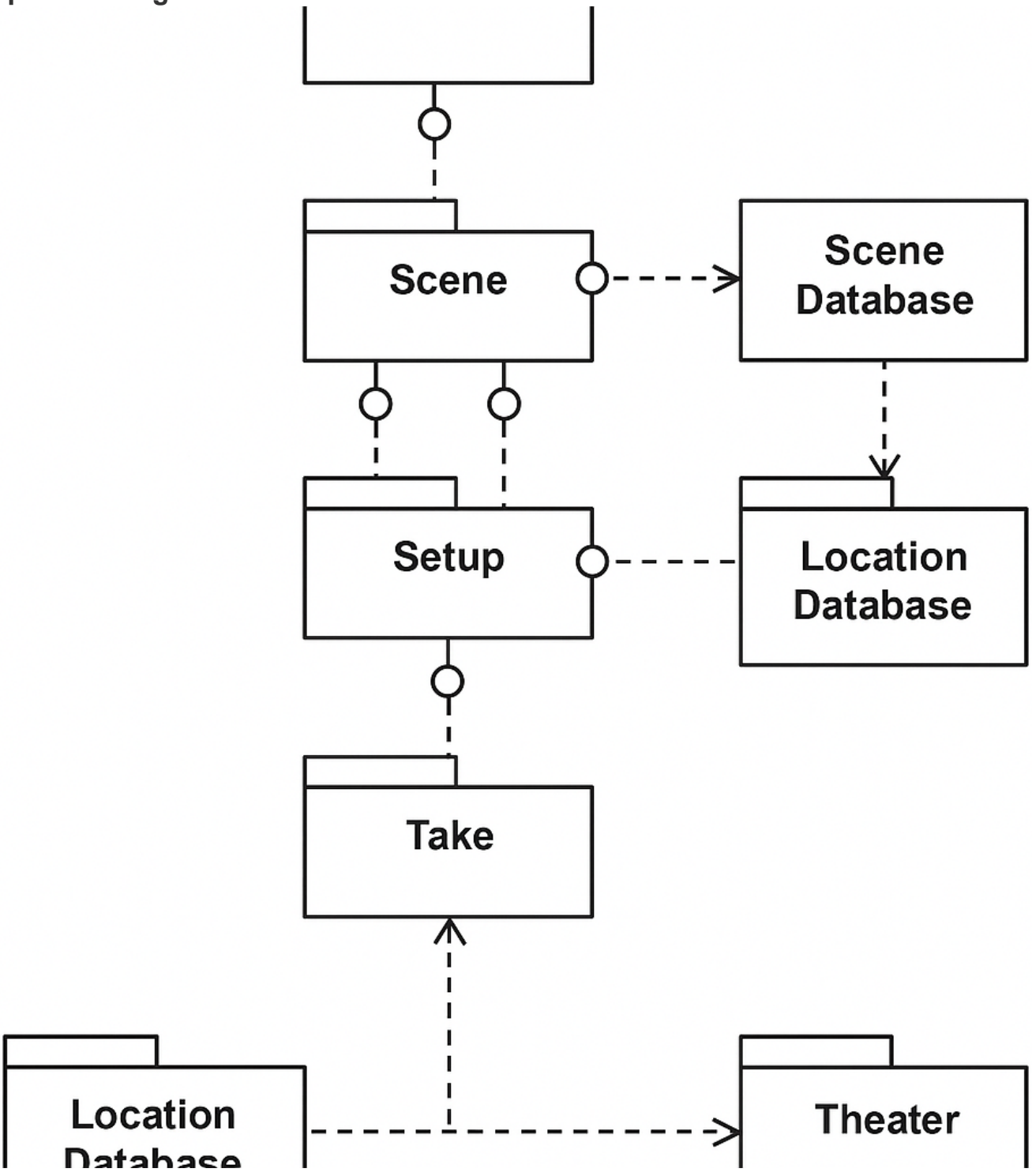
Q12.

Requirements:

Hollywood Approach

We are interested in building a software application to manage filmed scenes for realizing a movie, by following the so-called “Hollywood Approach”. Every scene is identified by a code (a string) and it is described by a text in natural language. Every scene is filmed from different positions (at least one), each of this is called a setup. Every setup is characterized by a code (a string) and a text in natural language where the photographic parameters are noted (e.g., aperture, exposure, focal length, filters, etc.). Note that a setup is related to a single scene. For every setup, several takes may be filmed (at least one). Every take is characterized by a (positive) natural number, a real number representing the number of meters of film that have been used for shooting the take, and the code (a string) of the reel where the film is stored. Note that a take is associated to a single setup. Scenes are divided into internals that are filmed in a theater, and externals that are filmed in a location and can either be “day scene” or “night scene”. Locations are characterized by a code (a string) and the address of the location, and a text describing them in natural language.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Diagram cuts off at top and bottom. Describes the data architecture but not the system architecture

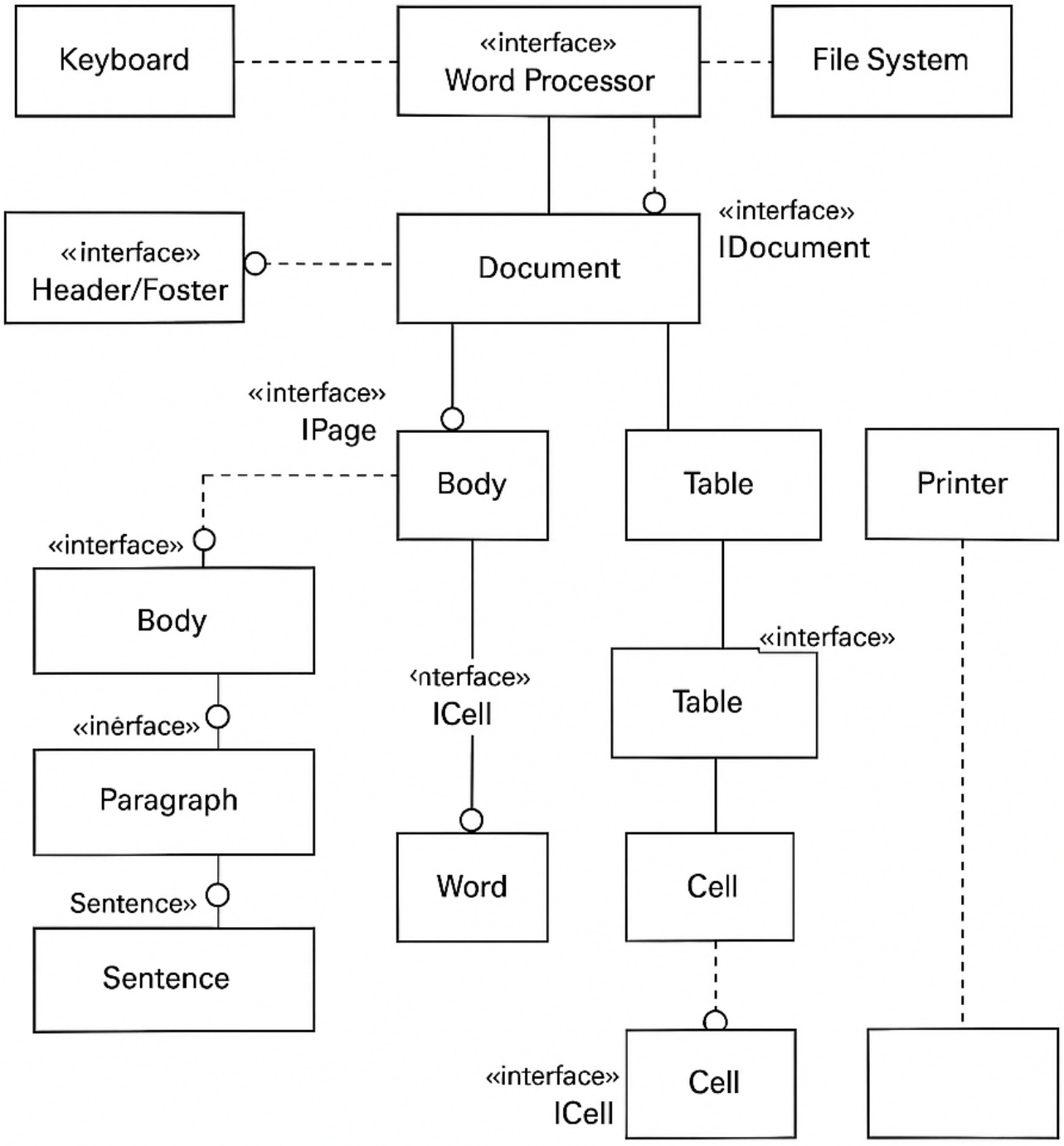
Q12.

Requirements:

Word Processor

A user can open a new or existing document. Text is entered through a keyboard. A document is made up of several pages and each page is made up of a header, body and footer. Date, time and page number may be added to header or footer. Document body is made up of sentences, which are themselves made up of words and punctuation characters. Words are made up of letters, digits and/or special characters. Pictures and tables may be inserted into the document body. Tables are made up of rows and columns and every cell in a table can contain both text and pictures. Users can save or print documents.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○

Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Empty box, disconnected printer. Does not account for save action, table should be connected with body, no mention of images. Data architecture instead of system architecture again

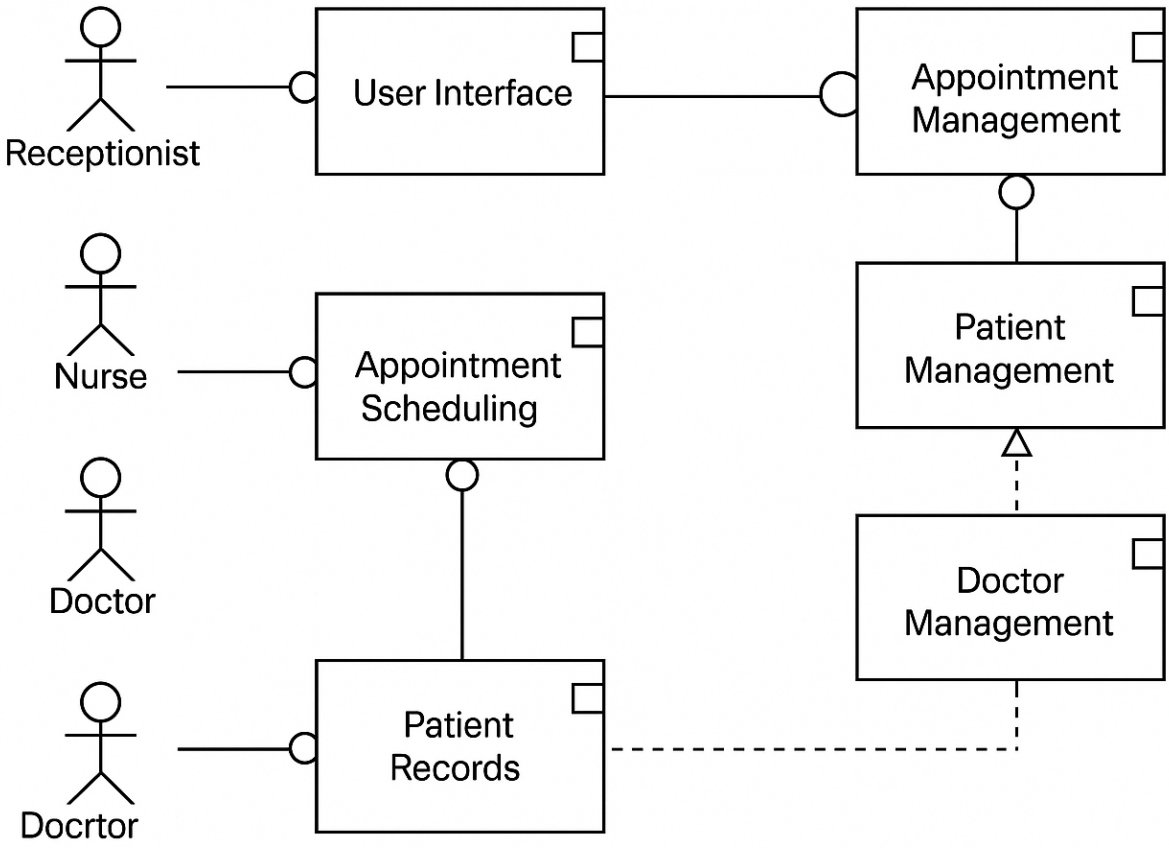
Q12.

Requirements:

Patient Record and Scheduling System

A patient record and scheduling system in a doctor's office is used by the receptionists, nurses, and doctors. The receptionists use the system to enter new patient information when first time patients visit the doctor. They also schedule all appointments. The nurses use the system to keep track of the results of each visit including diagnosis and medications. For each visit, free form text fields are used captures information on diagnosis and treatment. Multiple medications may be prescribed during each visit. The nurses can also access the information to print out a history of patient visits. The doctors primarily use the system to view patient history. The doctors may enter some patient treatment information and prescriptions occasionally, but most frequently they let the nurses enter this information. -- Each patient is assigned to a family. The head of family is responsible for the person with the primary medical coverage. Information about doctors is maintained since a family has a primary care physician, but different doctors may be the ones seeing the patient during the visit.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ◉ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Includes users like a use case, not a component diagram. Additional components that are not needed ("Doctor Management"). At least is a system architecture

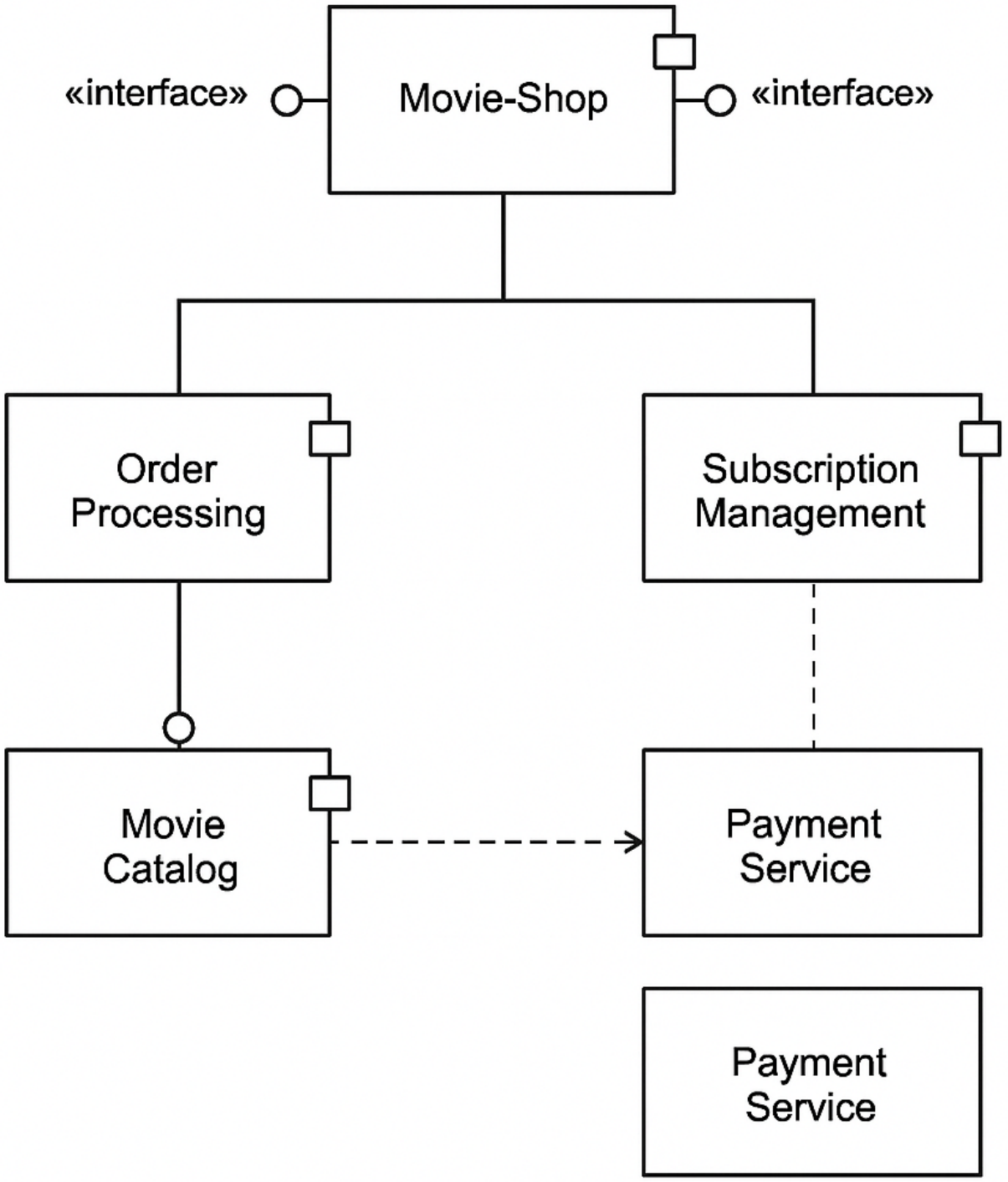
Q12.

Requirements:

Movie-Shop

♣ Design a system for a movie-shop, in order to handle ordering of movies and browsing of the catalogue of the store, and user subscriptions with rechargeable cards. ♣ Only subscribers are allowed hiring movies with their own card. ♣ Credit is updated on the card during rent operations. ♣ Both users and subscribers can buy a movie and their data are saved in the related order. ♣ When a movie is not available it is ordered .

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ◉ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box [ ]

Q14. Please provide any other comments you might have about this component diagram:

Double component on the payment service

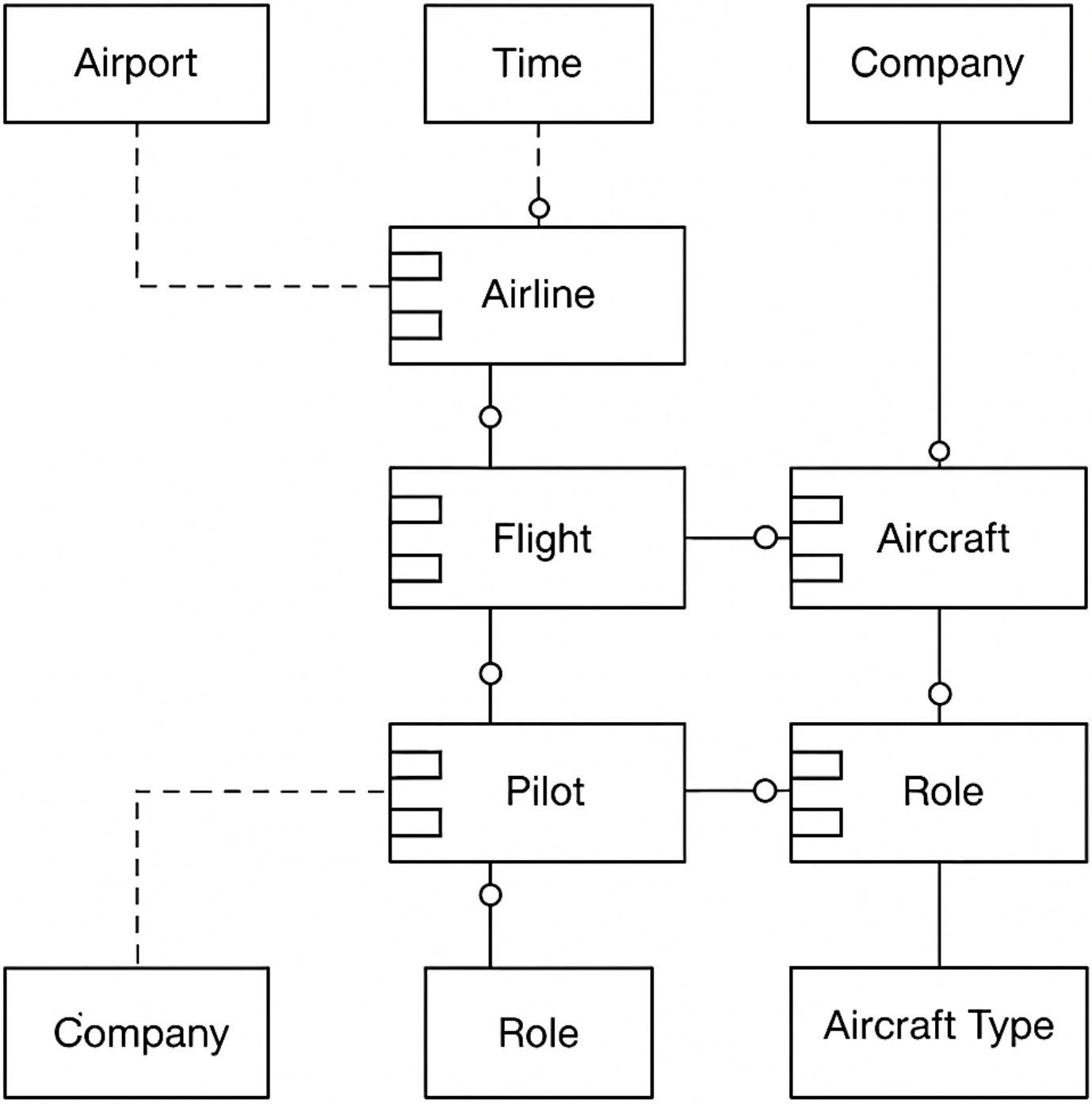
Q12.

Requirements:

Flights

We want to model a system for management of flights and pilots. An airline operates flights. Each airline has an ID. Each flight has an ID a departure airport and an arrival airport: an airport as a unique identifier. Each flight has a pilot and a co-pilot, and it uses an aircraft of a certain type; a flight has also a departure time and an arrival time. An airline owns a set of aircrafts of different types. An aircraft can be in a working state or it can be under repair. In a particular moment an aircraft can be landed or airborne. A company has a set of pilots: each pilot has an experience level: 1 is minimum, 3 is maximum. A type of aeroplane may need a particular number of pilots, with a different role (e.g.: captain, co-pilot, navigator): there must be at least one captain and one co-pilot, and a captain must have a level 3.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○

Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box [ ]

Q14. Please provide any other comments you might have about this component diagram:

Incorrect connections between components. Data architecture, not system architecture

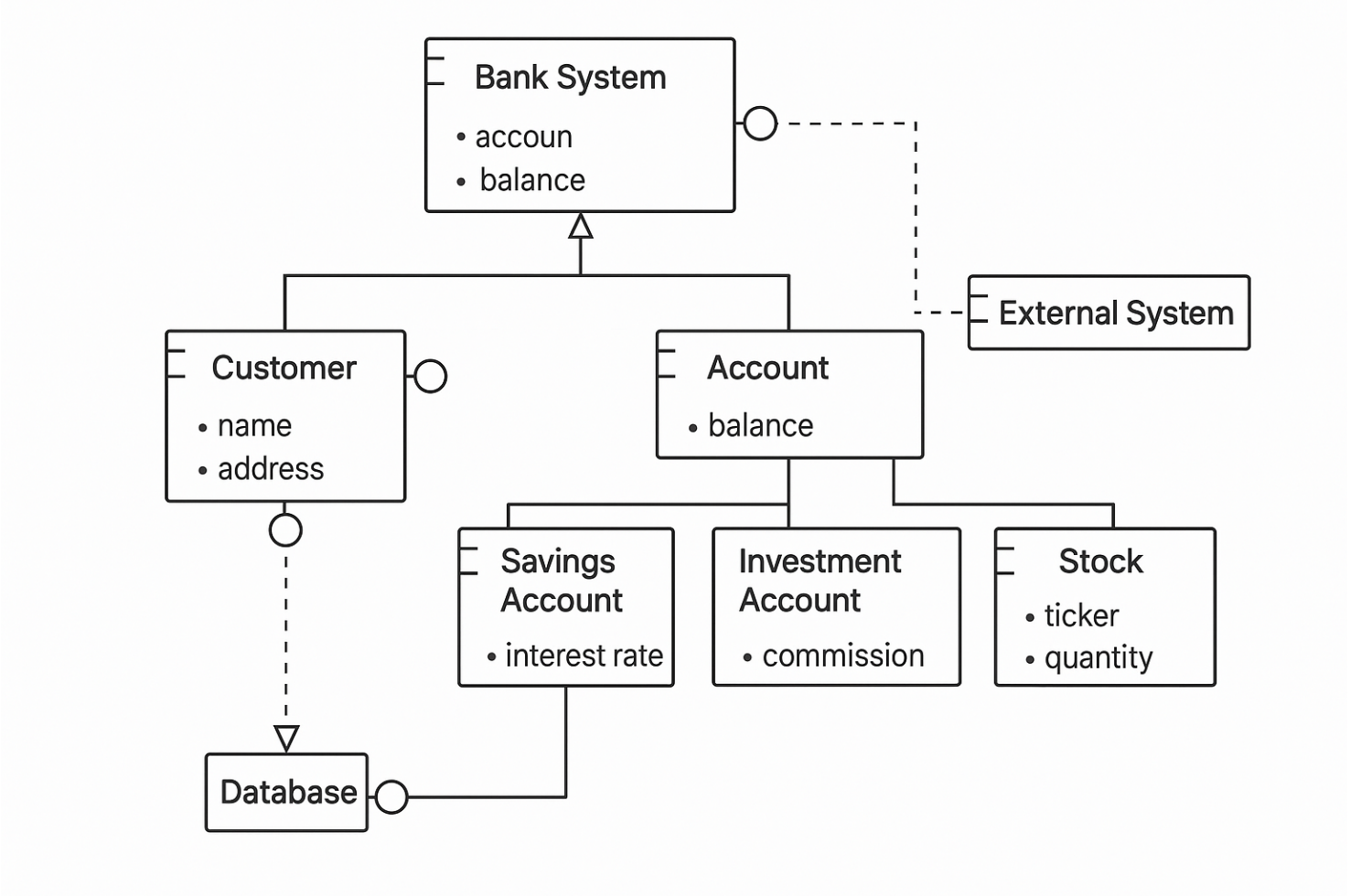
Q12.

Requirements:

Bank System

A bank system contains data on customers (identified by name and address) and their accounts. Each account has a balance and there are 2 type of accounts: one for savings which offers an interest rate, the other for investments, used to buy stocks. Stocks are bought at a certain quantity for a certain price (ticker) and the bank applies commission on stock orders.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	◉	○	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Again, more of a data architecture. Balance repeated across multiple components. Includes unmentioned external system.

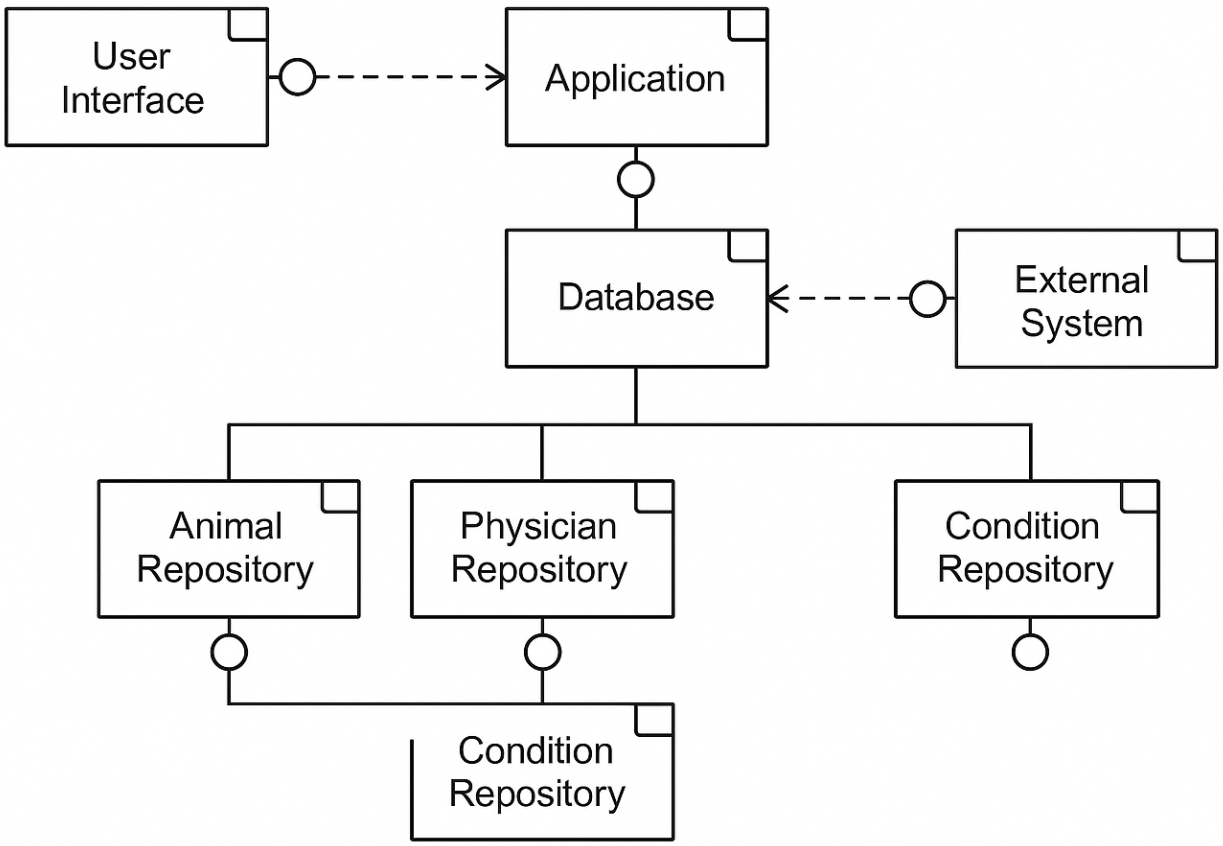
Q12.

Requirements:

Veterinary Clinic

The owner of a veterinary clinic wants to create a database to store information about all veterinary services performed. After some research he came up with the following requirements:
- For each admitted animal, its name, breed (if any) and owner must be stored. Each animal should be given an unique numeric identifier.
- For each owner, its name, address and phone number should be stored. An unique numeric identifier should also be generated for each one of them.
- An animal might be owner-less. This happens frequently as the clinic often rescues abandoned dogs from the streets in order to treat them and get them new owners.
- It should be possible to store information about a specific breed even if no animals of that breed have been treated at the clinic.
- Each appointement always has a responsible physician. All appointements start at a certain date and time; and are attended by an animal (and of course its owner).
- For each physician, his name, address and phone number should be stored. An unique numeric identifier should also be generated for each one of them.
- In an appointement, several medical conditions might be detected. Each condition has a common name and a scientific name. No two conditions have the same scientific name.
- It should be possible to store information about the most common conditions for each different

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ◉ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Shows clear (if somewhat useless) layers between UI, application, and database. Maps the different types of data, not the functionality

Q12.

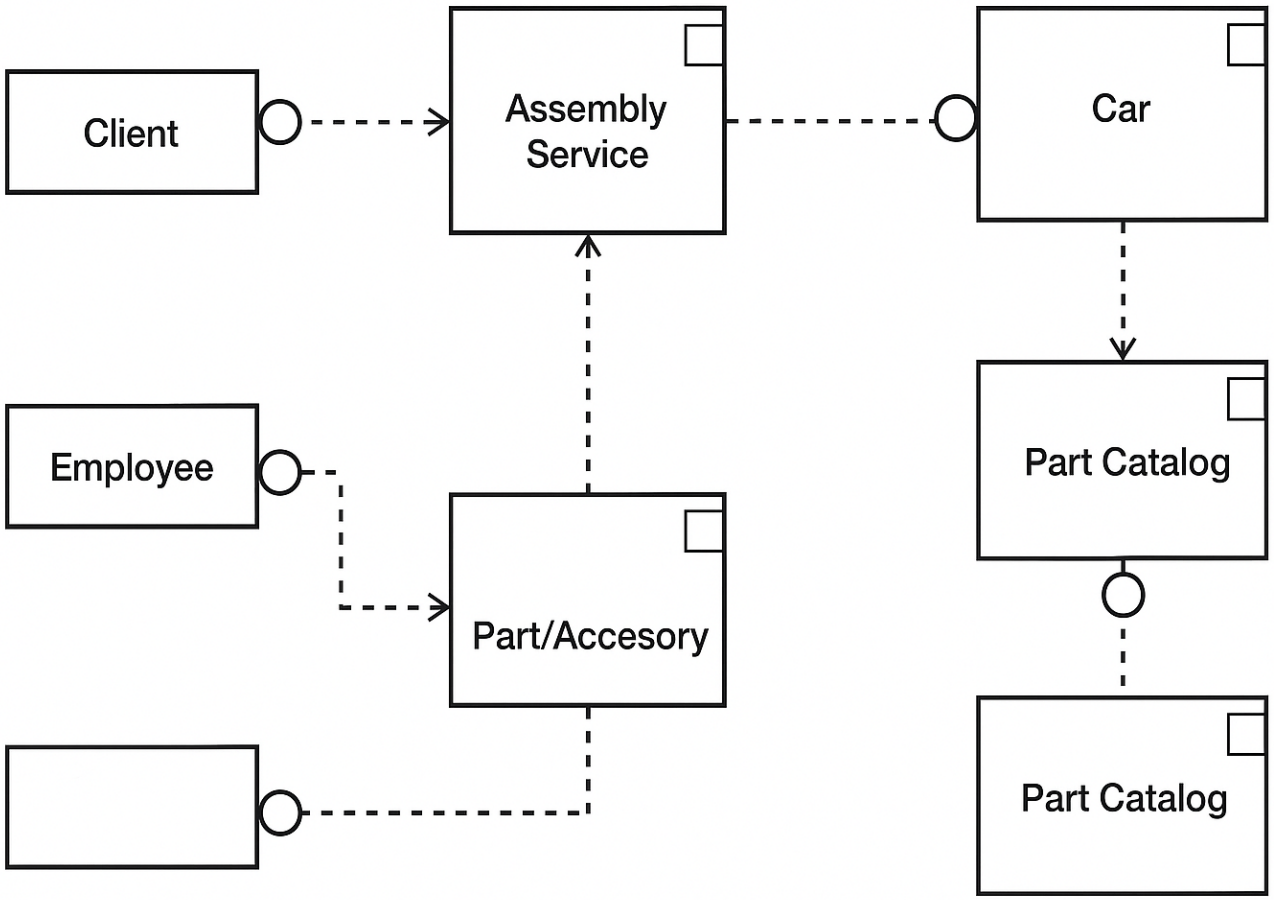
Requirements:

Auto Repair

An auto repair shop, that sells and mounts parts and accessories for all kinds of vehicles, wants a new information system to manage their clients, parts, accessories and assembly services:

- There are several employees. Each one of them has an unique identifying number, a name and an address.
- In this shop, assembly services, where parts and accessories are installed in a vehicle, are executed. For each one these services the following data must be stored: In which car the service was executed, how many kms had the car at the time, who was the responsible employee, which parts and accessories were fitted, how many work hours did it take and the admission and finish dates.
- Parts and accessories are only sold together with an assembly service.
- Each part/accessory only fits in some car models. Therefore, it is important to store that information.
- Each part/accessory has a category (radio, tyre, …), a serial number and a price.
- Each car has a license plate, a make, a model, a color and an owner. Each owner has a name, identifying number, address and a phone.
- One person can own more than one car but one car only has one owner.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Repeated / blank components. Represents data but not behavior

Q12.

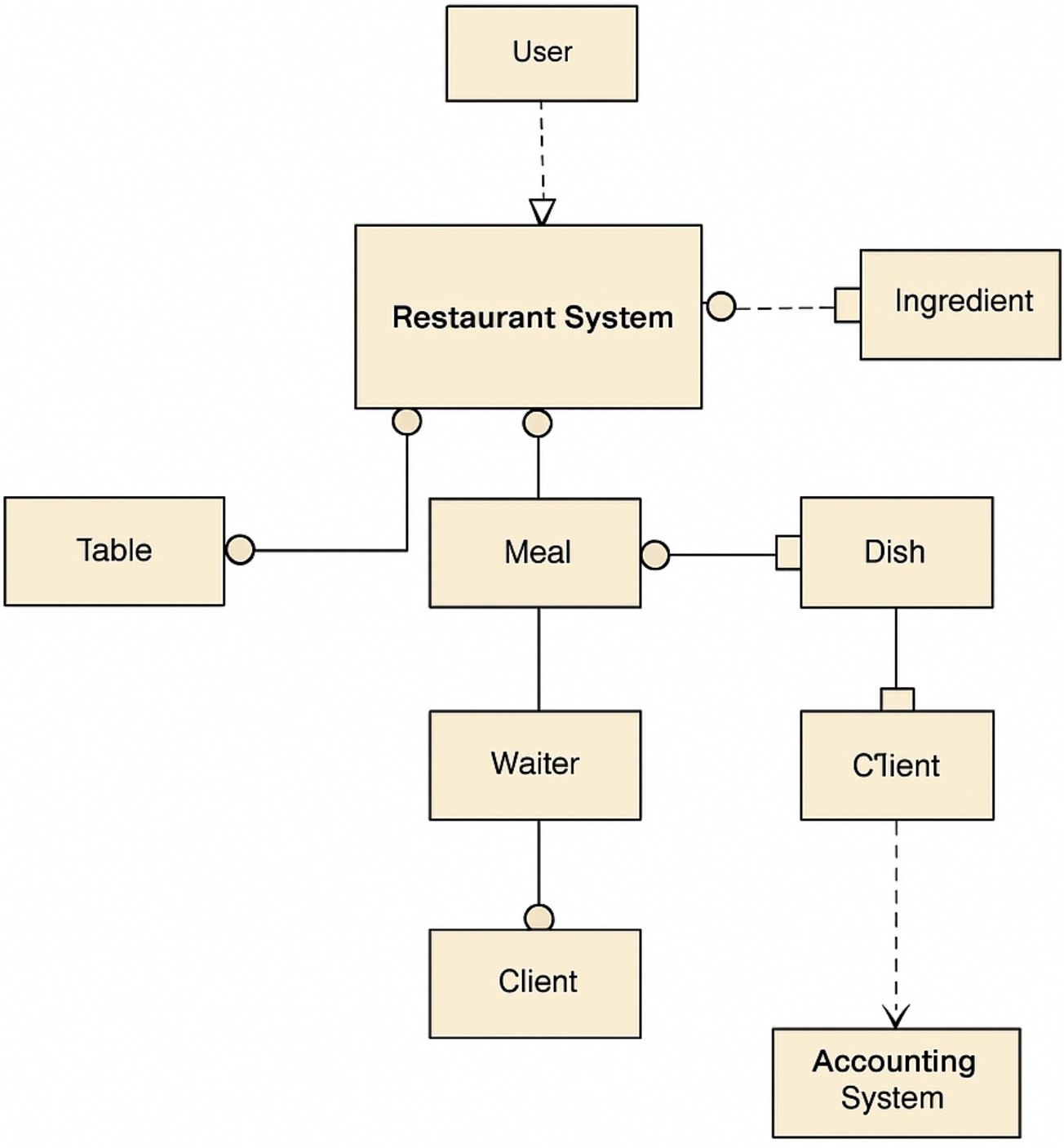
Requirements:

10. Restaurant

The owner of a small restaurant wants a new information system to store data for all meals consumed there and also to keep a record of ingredients kept in stock. After some research he reached the following requirements list:

- Each ingredient has a name, a measuring unit (e.g. olive oil is measured in liters, while eggs are unit based) and a quantity in stock. There are no two ingredients with the same name.
- Each dish is composed of several ingredients in a certain quantity. An ingredient can, of course, be used in different dishes.
- A dish has an unique name and a numeric identifier.
- There are several tables at the restaurant. Each one of them has an unique numeric identifier and a maximum ammount of people that can be seated there.
- In each meal, several dishes are consumed at a certain table. The same dish can be eaten more than once in the same meal.
- A meal takes place in a certain date and has a start and end time. Each meal has a responsible waiter.
- A waiter has an unique numerical identifier, a name, an address and a phone number.
- In some cases it is important to store information about the client that consumed the meal. A client has a tax identification number, a name and an address.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ◉ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box [ ]

Q14. Please provide any other comments you might have about this component diagram:

Seems to be all clumped under "restaurant service" with some data design underneath. Duplication of Client component

Q12.

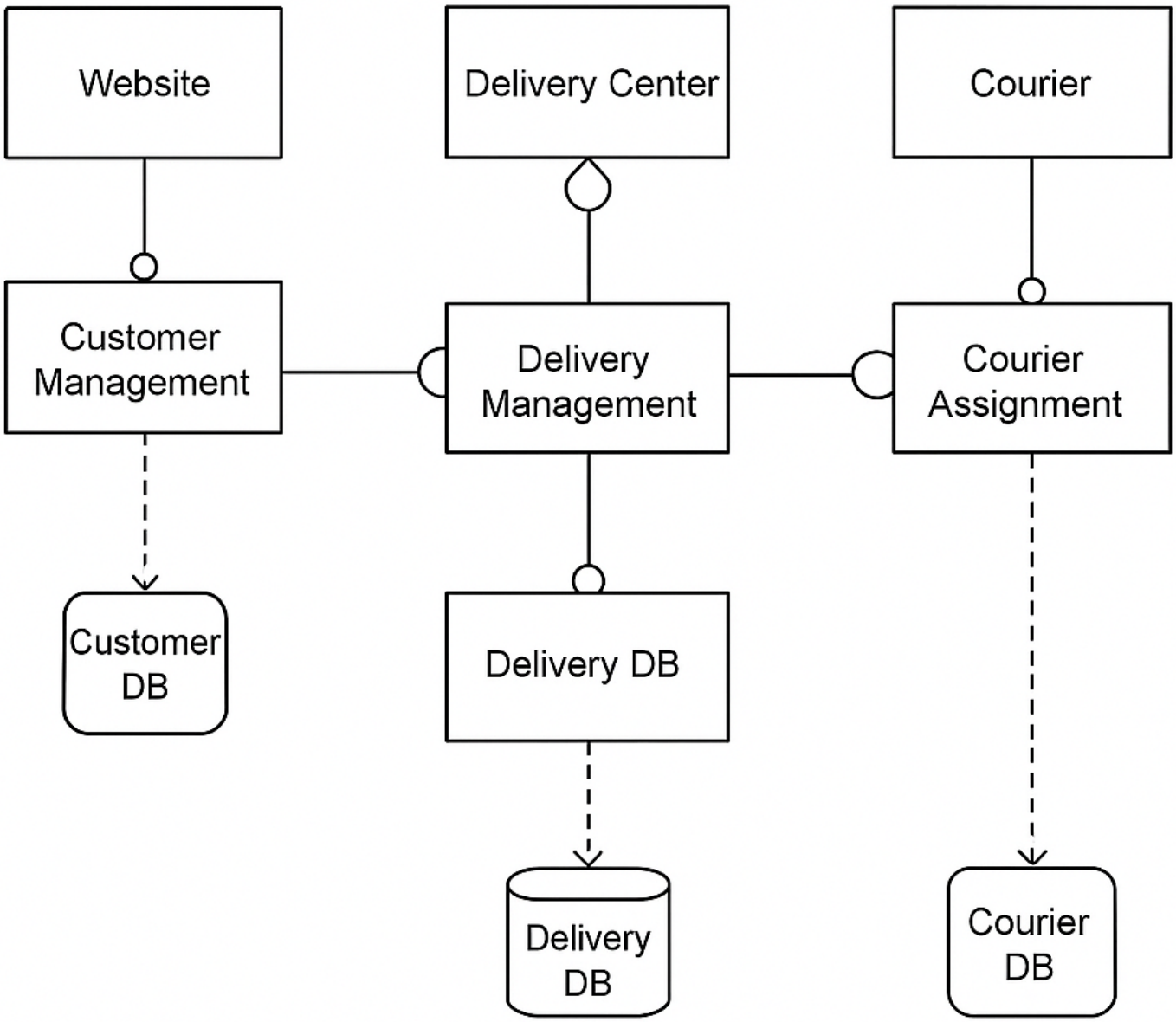
Requirements:

Deliveries

The owner of a small delivery company plans to have an information system that allows him to save data about his customers and deliveries. After some time studying the problem, he reached the following requirements:

- Each customer has a VAT number, a name, a phone number and an address. There are no two clients with the same VAT number.
- When a customer wants to send a package to another customer, he just has to login to the company website, select the customer he wants to send the package to, enter the package's weight and if the delivery is normal or urgent. He then receives an unique identifier code that he writes on the package.
- The package is then delivered by the customer at the delivery center of his choosing. A delivery center has a unique name and an address.
- Each client has an associated delivery center. This delivery center is chosen by the company and it is normally the one closest to the customer's house.
- The package is them routed through an internal system until it reaches the delivery center of the recipient.
- The package is then delivered by hand from that delivery center to the recipient by a courier.
- Couriers have a single VAT number, a name and a phone number. Each courier works in a single

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○

Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ◉ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box [ ]

Q14. Please provide any other comments you might have about this component diagram:

Missing many "domain knowledge" kinds of components, e.g. service to determine closest delivery center.

Q12.

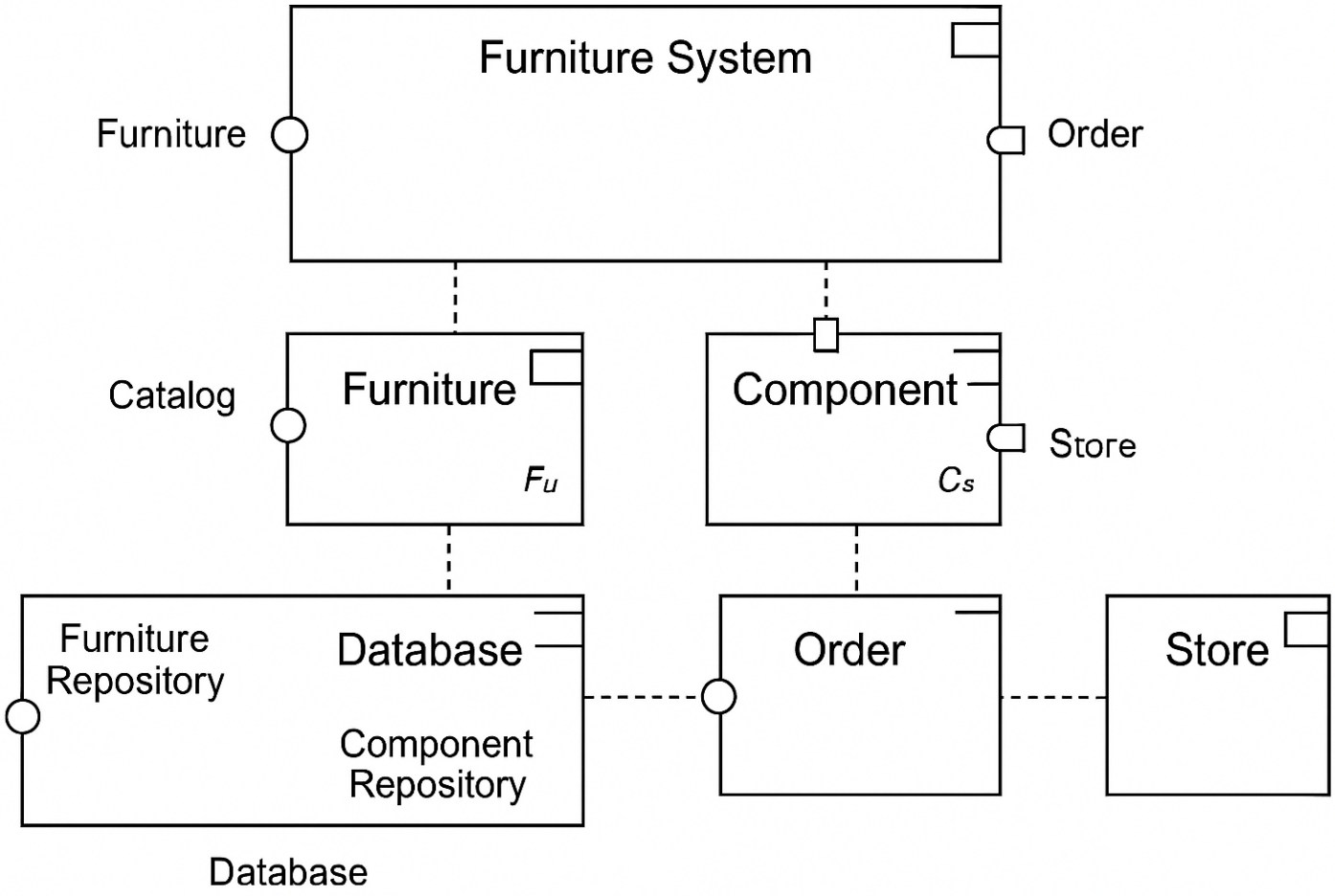
Requirements:

Furniture

The known furniture factory Hi-Key-Ah, intends to implement an information system to store all data on the different types of furniture and components it produces:

- The factory produces several lines of furniture, each with a different name and consisting of several pieces of furniture of different types (beds, tables, chairs, …).
- All furniture pieces have a type, a single reference (eg CC6578) and a selling price.
- The major competitive advantage of this innovative plant is the fact that each component produced can be used in more than one piece of furniture.
- Each piece of furniture is thus composed of several components. The same component can be used more than once in the same piece.
- Every type of component produced is assigned a unique numerical code, a manufacturing price and a type (screw, hinge, shelf …).
- The furniture is then sold in various stores throughout the world. Each store has a different address and a fax number.
- To make the manufacturing process more efficient, stores have to place orders everytime they need to replenish their stock. These orders must also be stored in the database.
- Each order has a order number, a date, the store that placed the order as well as a list of all the ordered furniture and their quantities.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ◉ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Data grouped under the aegis of the "Furniture System"

Q12.

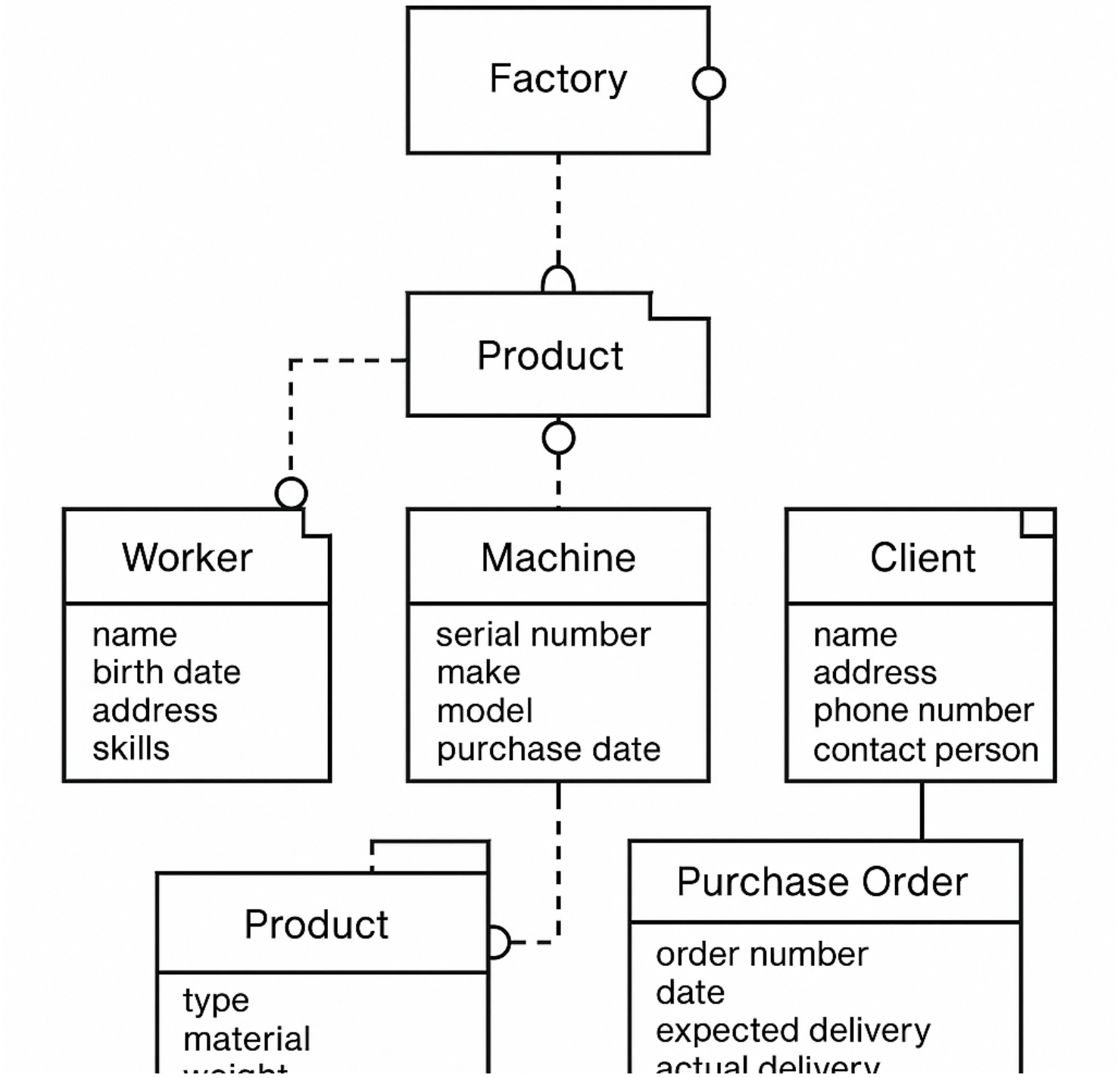
Requirements:

Factory

Create a database for a factory with the following requirements. Don’t forget to add unique identifiers for each one of the entities if needed.

- A factory has several machines. Each one of them is operated by several workers.
- A worker might work in more than one machine.
- In this factory, several products of different types, are produced. Each different type of product is produced in a single machine. But, the same machine can produce more than one type of product.
- Products from the same type are all produced from the same single material and have the same weigth.
- Clients can issue purchase orders. Each order has a list of the desired products and their quantity.
- For each worker, the following data should be stored in the database: name (first and last), birth date, address and a list of his skills.
- For each machine, the following data should be stored: serial number, make, model and purchase date.
- For each client, the followig data should be stored: name, address, phone number and name of the contact person (if any).

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box [ ]

Q14. Please provide any other comments you might have about this component diagram:

This is clearly a database design, not a software design (database was specifically requested in the requirements)

Q12.

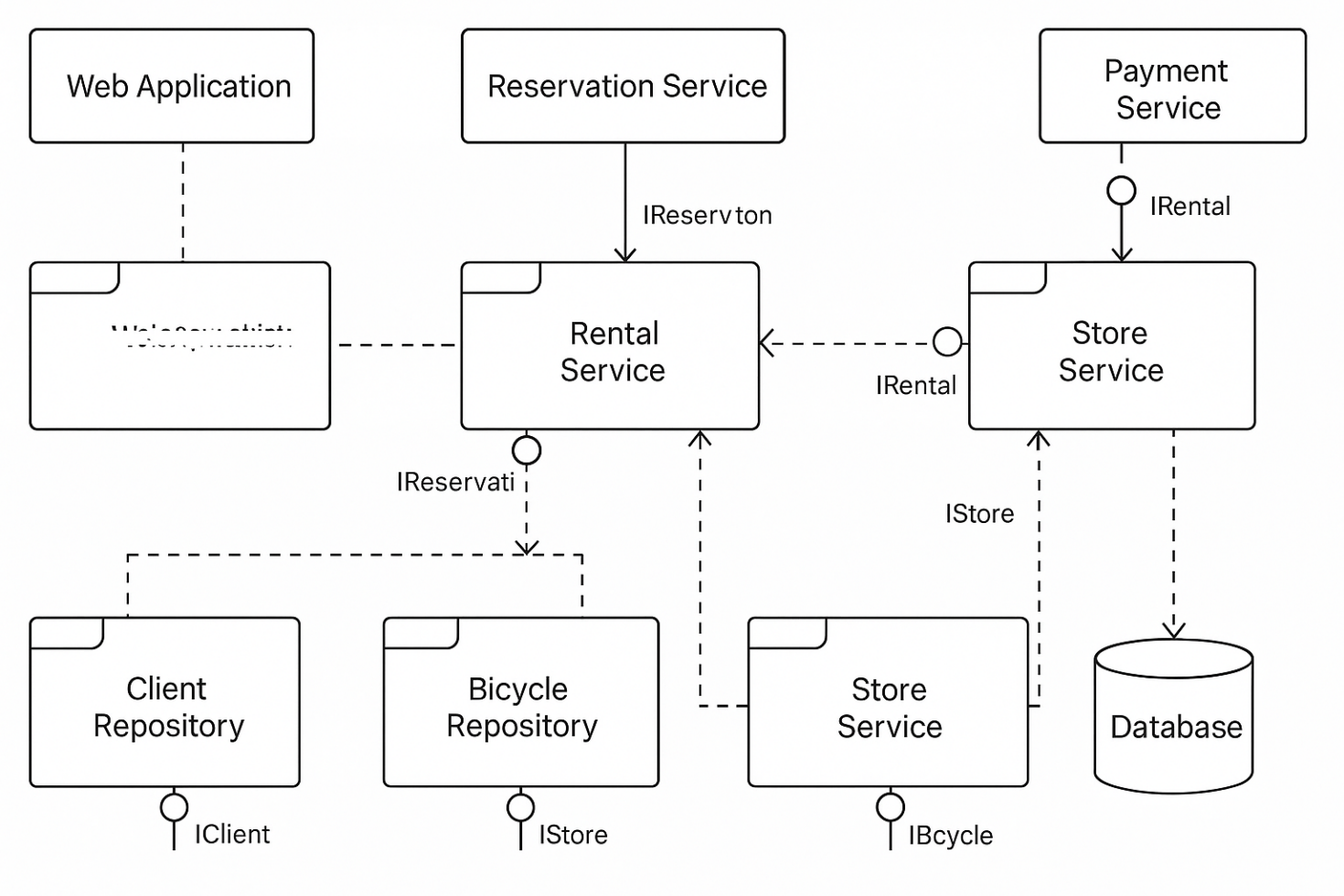
Requirements:

Bycicle Rental

A bicycle renting company wants to create an information system that allows it to store the data regarding all their reservations and rentals. The system should follow these requirements:

- It should be possible to store the national id number (NIN), tax identification number (TIN), name and address for every client. The NIN and TIN must be different for every client and all clients should have at least a TIN and a name.
- The database should also contain information about the bicycle models that can be rented- Each model has an unique name, a type (that can only be road, mountain, bmx or hybrid) and the number of gears.
- Each bicycle has a unique identifying number and a model.
- The company has several different stores where bicycles can be picked up and returned. Each one of these stores is identified by an unique name and has an address (both mandatory).
- When a reservation is made, the following data must be known: which client did the reservation, when will he pick up the bike (day), which bike model he wants and where will he pick up the bike (store).
- When a bike is picked up, the actual bike that was picked up must be stored in the database.
- When a bike is returned, the return date should be stored in the database.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ◉ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Duplicate Store Service, strange...other thing on the left. Some text issues.

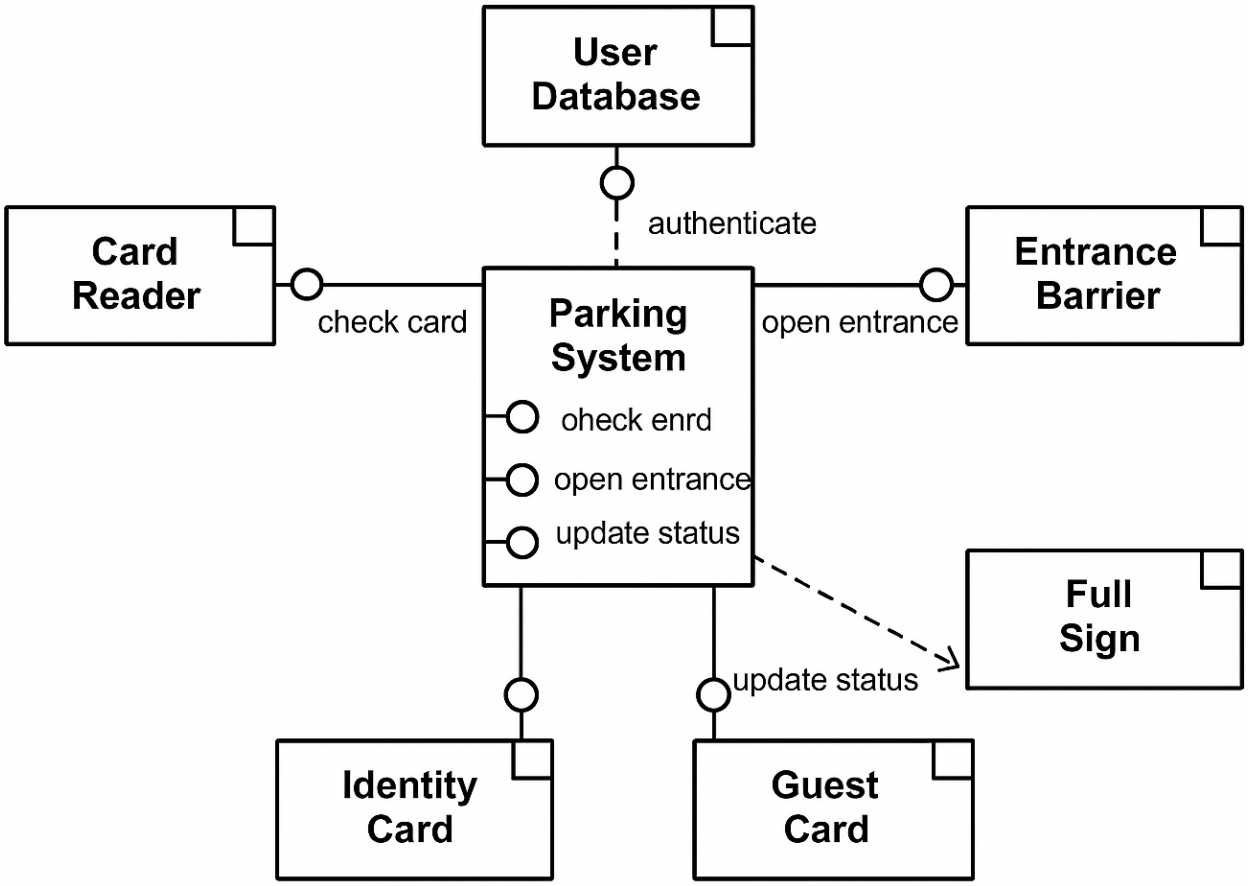
Q12.

Requirements:

Saturn Int. Management

Saturn Int. management wants to improve their security measures, both for their building and on site. They would like to prevent people who are not part of the company to use their car park. Saturn Int. has decided to issue identity cards to all employees. Each card records the name, department and number of a company staff, and give them access to the company car park. Employees are asked to wear the cards while on the site.
There is a barrier and a card reader placed at the entrance to the car park. When a driver drives his car into the car park, he/she inserts his or her identity card into the card reader. The card reader then verify the card number to see if it is known to the system. If the number is recognized, the reader sends a signal to trigger the barrier to rise. The driver can then drive his/her car into the car park.
There is another barrier at the exit of the car park, which is automatically raised when a car wishes to leave the car park.
A sign at the entrance display "Full" when there are no spaces in the car park. It is only switched off when a car leaves.
There is another type of card for guests, which also permits access to the car park. The card records a number and the current date. Such cards may be sent out in advance, or collected from reception. All guest cards must be returned to reception when the visitor leaves Saturn Int.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ◉ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

This one isn't bad (apart from, say "oheck enrd"). Includes functionality as interface names

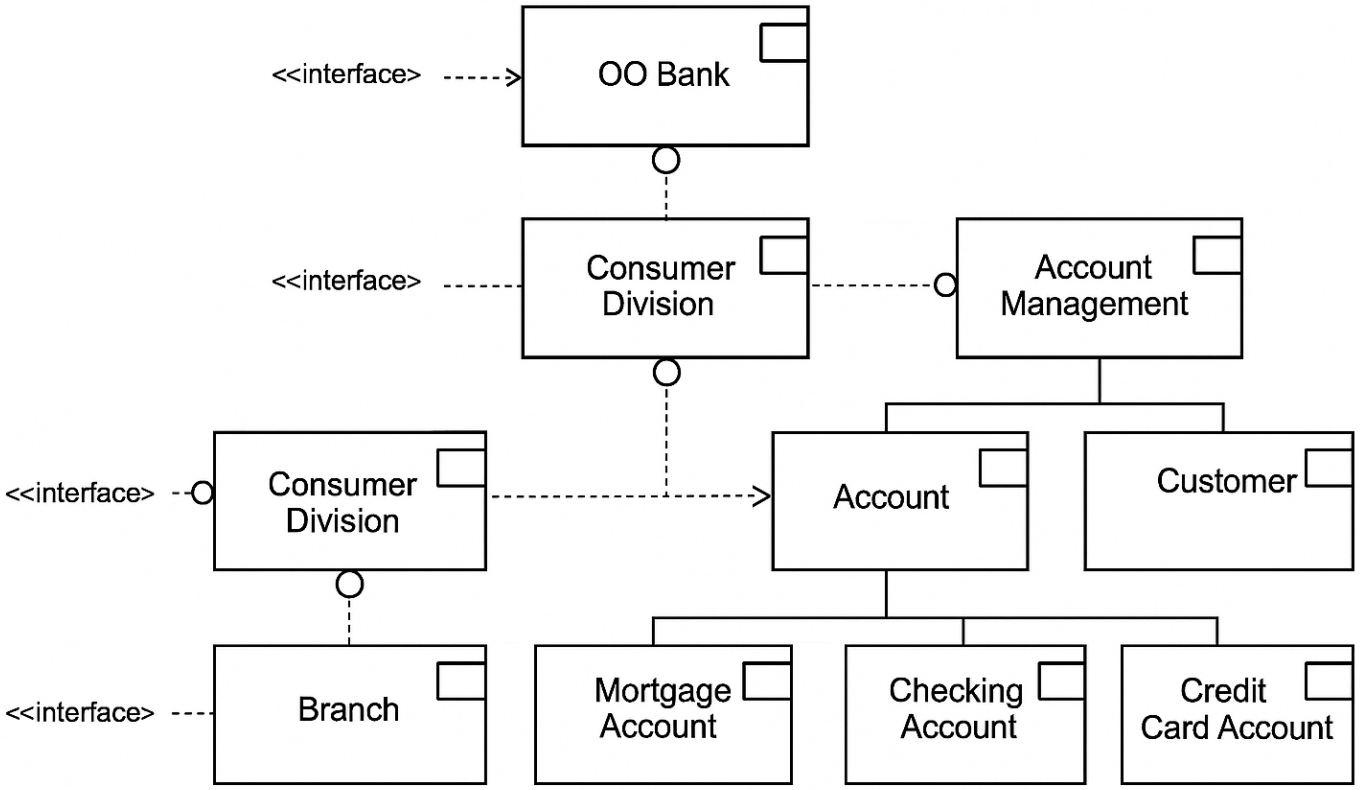
Q12.

Requirements:

OOBank

This system provides the basic services to manage bank accounts at a bank called OOBank. OOBank has many branches, each of which has an address and branch number. A client opens accounts at a branch. Each account is uniquely identified by an account number; it has a balance and a credit or overdraft limit. There are many types of accounts, including: a mortgage account (which has a property as collateral), a checking account, and a credit card account (which has an expiry date and can have secondary cards attached to it). It is possible to have a joint account (e.g. for a husband and wife). Each type of account has a particular interest rate, a monthly fee and a specific set of privileges (e.g. ability to write checks, insurance for purchases etc.). OOBank is divided into divisions and subdivisions (such as Planning, Investments and Consumer); the branches are considered subdivisions of the Consumer Division. Each division has a manager and a set of other employees. Each customer is assigned a particular employee as his or her 'personal banker'.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ◉ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

Diagram layers are pretty well defined, although not necessarily correct. Duplicate consumer division

Q12.

Requirements:

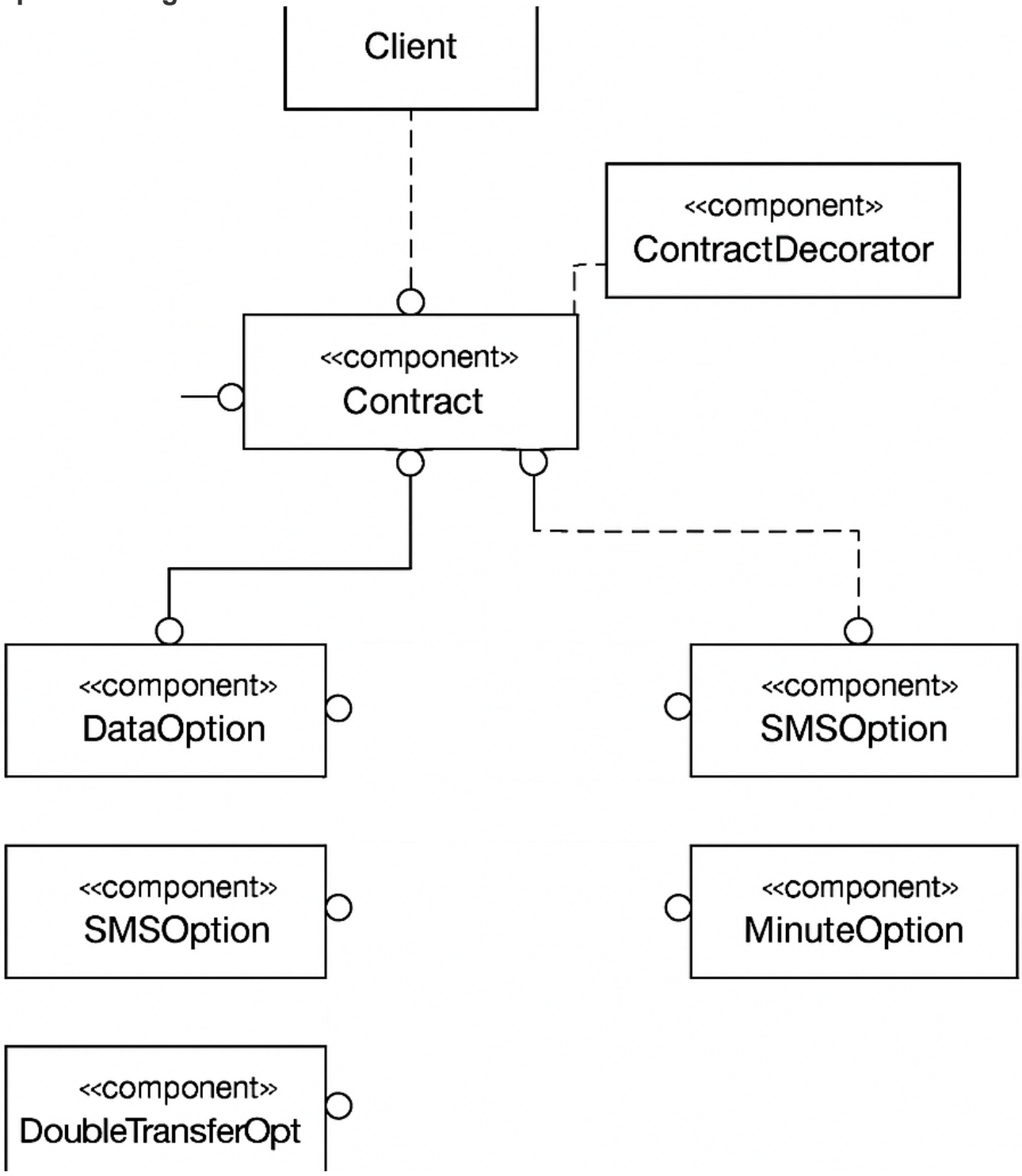
Prepaid Cell Phone

The contract of a prepaid cell phone should be modelled and implemented. A basic contract has a contract number (of type int) and a balance (of type double), but no monthly charges. The contract number is not automatically generated, but is to be set as a parameter by the constructor as well as the initial balance. The balance has a getter and a setter. The following options can be added to a contract (if needed also several times):

- 100 MB of data (monthly charge 1.00€)
- 50 SMS (monthly charge 0.50€)
- 50 minutes (monthly charge 1.50€)
- Double Transfer Rate (monthly charge 2.00€) implement this requirement with the help of the decorator pattern. All contract elements should be able to understand the methods getCharges():double, getBalance():double and setBalance(double).

The method getCharges() should provide the monthly charge of a contract with all its options selected. The methods getBalance() and setBalance() should be passed through and access the basic contract.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ◉ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box [ ]

Q14. Please provide any other comments you might have about this component diagram:

Everything nested under contract. Seems focused on data model design, not system design

Q12.

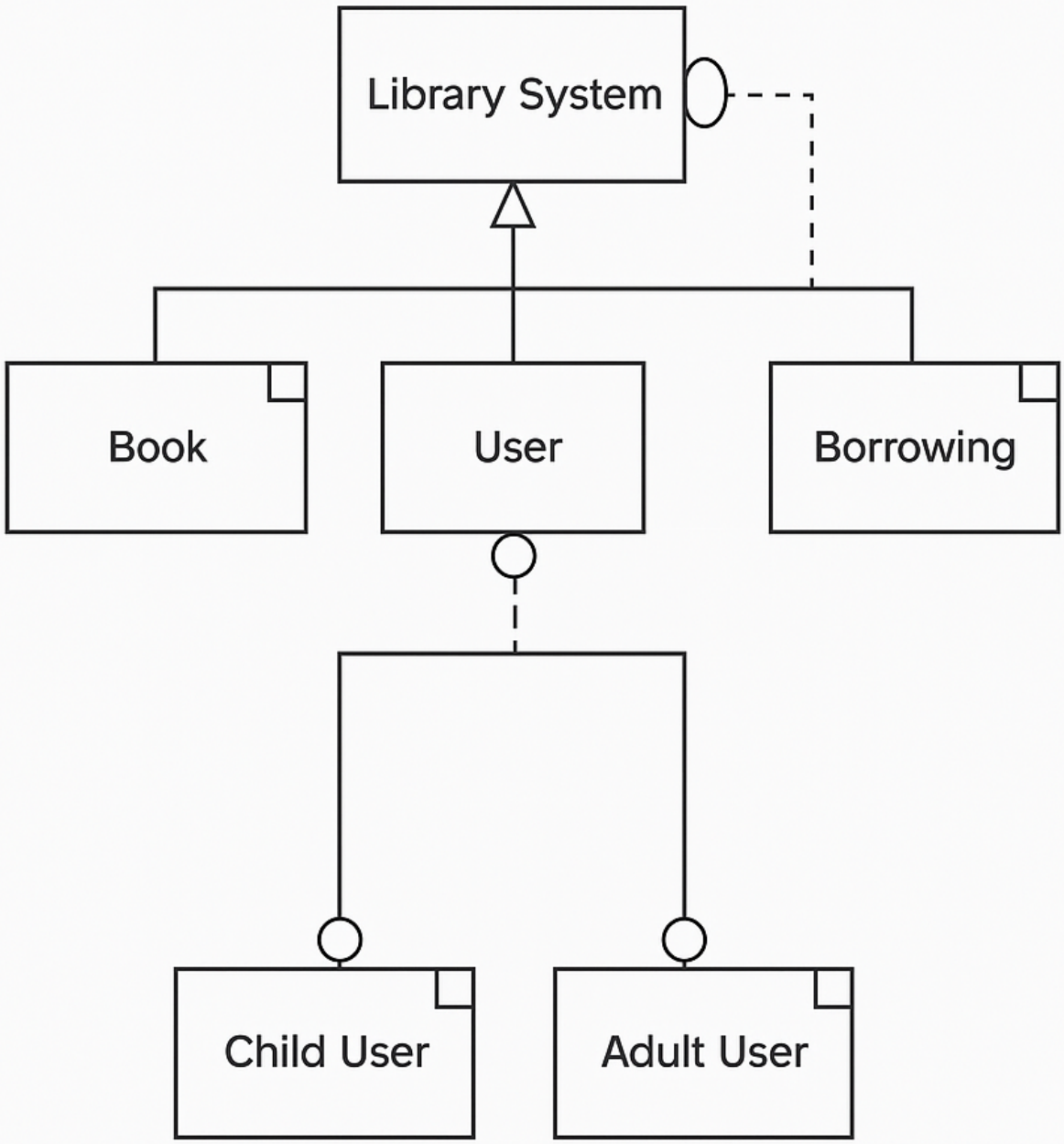
Requirements:

Library System

The exercise id to design a class structure for a library system. It should fulfil those requirements:

- There are two type of users – under-aged and adults.
- Under-aged users are identified with usage of their full name and student card.
- Adult users are identified with usage of their full name and ID card.
- The library contains books.
- There is basic information about every book (title, author, etc).
- The user can borrow at most 4 books at the same time.
- There is a history of previously borrowed books for every user (along with all the dates).

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○

Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

The prompt specifically calls out class design, so it made an OO class structure, not a real system architecture. Might need to change the prompt.

Q12.

Requirements:

MyDoctor

The MyDoctor application aims to be a management tool for the appointments of a doctor.
A hospital has multiple offices.
The users of the application can be doctors and patients.
The doctors can apply to practice in offices and create a schedule for an office. The schedules in different offices can't overlay.

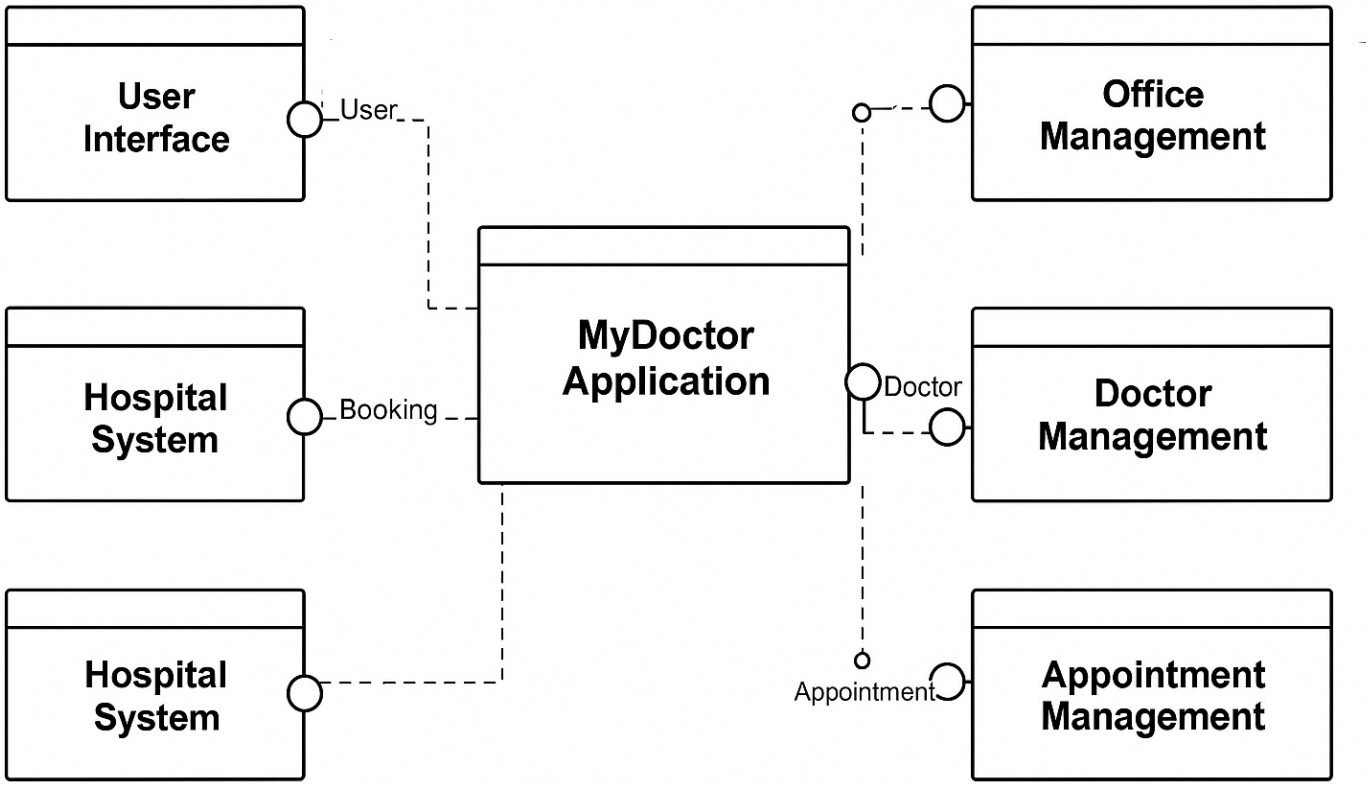
Example:
Doctor Ana is available in Office 4 on the 4th of September during 1 PM - 5PM.
Doctor Ana can't practice in Office 5 on the 4th of September during 3PM - 8 PM, but she can practice in Office 5 on the 4th of September during 5:30PM - 8 PM.
The patients can see the existing doctors in the system, the schedule of the offices and can book appointments for specific doctors and for specific schedules. The appointments can be of 3 types:

- Blood Test - 15 mins
- Consultation - 30 mins
- Surgery - 60 mins

The booking of an appointment will not be possible if another appointment is already booked at the same time frame. An email is sent to the patient with the confirmation of the appointment.

Example:

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○

Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ○ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ◉ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box

Q14. Please provide any other comments you might have about this component diagram:

All reference the central MyDoctor Application. Very high level

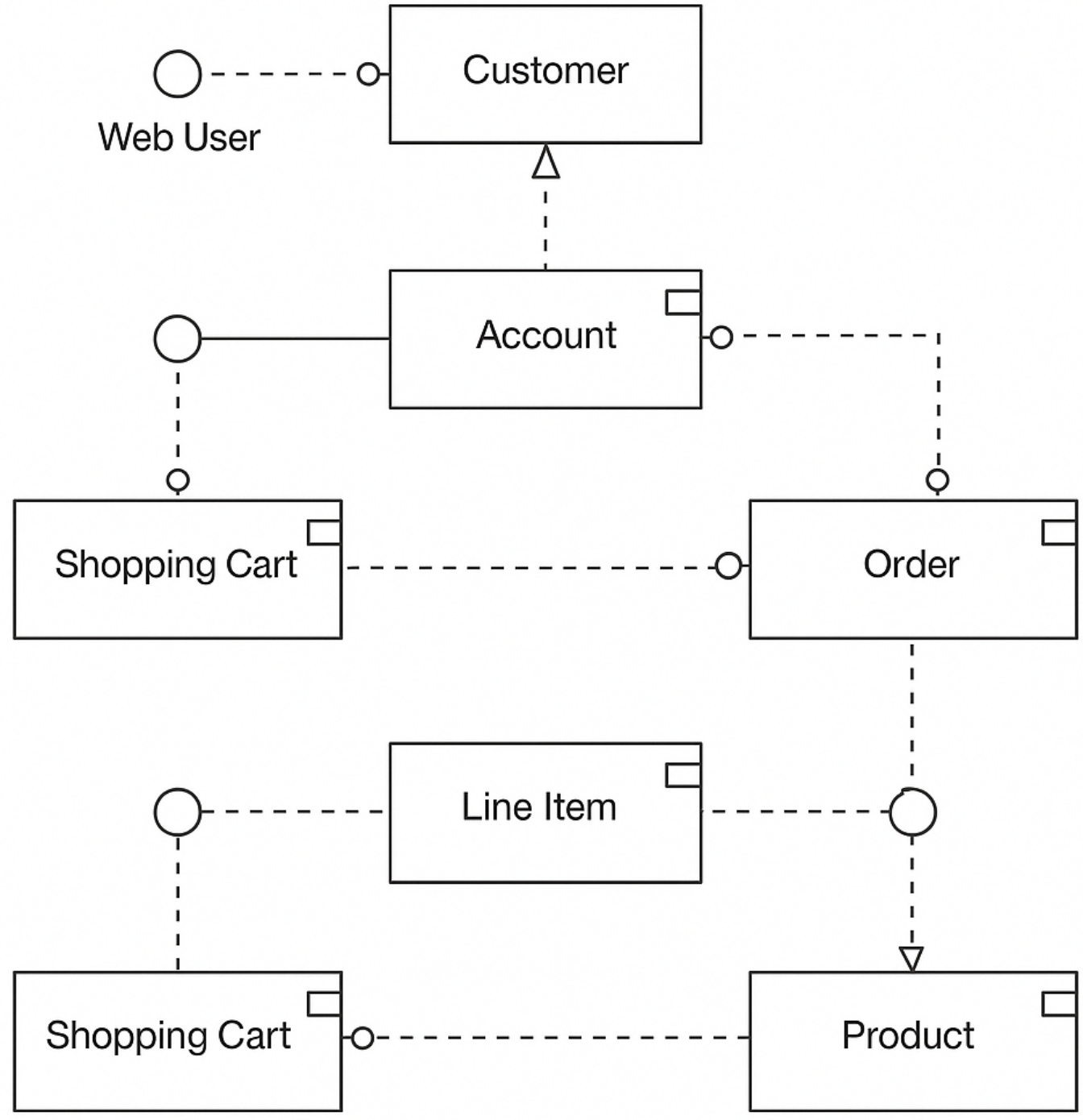
Q12.

Requirements:

Online Shopping

Each customer has unique id and is linked to exactly one account. Account owns shopping cart and orders. Customer could register as a web user to be able to buy items online. Customer is not required to be a web user because purchases could also be made by phone or by ordering from catalogues. Web user has login name which also serves as unique id. Web user could be in several states - new, active, temporary blocked, or banned, and be linked to a shopping cart. Shopping cart belongs to account.
Account owns customer orders. Customer may have no orders. Customer orders are sorted and unique. Each order could refer to several payments, possibly none. Every payment has unique id and is related to exactly one account.
Each order has current order status. Both order and shopping cart have line items linked to a specific product. Each line item is related to exactly one product. A product could be associated to many line items or no item at all.

Component Diagram:



Q13. Using the requirements specification above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○

Q13. Based on the diagram above, which of the following architectural styles best describes the architecture presented in the diagram?

- ○ **Monolithic:** The system is represented as a single, indivisible unit.
- ○ **Layered:** Components are divided into well-defined layers, with each layer interfacing only with components in the layers above or below.
- ○ **Client / Server:** Components are divided into service requestors (clients) and service providers (servers)
- ○ **Service Oriented:** Components represent loosely-coupled services distributed over a network that each independently manage some functionality.
- ◉ **Microservice / Object-oriented:** Components represent abstract data types / objects / microservices, each responsible for managing and hiding its own data / state.
- ○ **Pipe and Filter:** Each component has a set of inputs and a set of outputs connected via data streams from one component to another.
- ○ **Event-driven:** Components independently subsribed to and publish events to a shared event broker or bus.
- ○ **Repository:** The system is primarily driven by a large shared data structure representing the current state of the system with other components interacting to transform that state.
- ○ **Other:** Please briefly describe in the text box [ ]

Q14. Please provide any other comments you might have about this component diagram:

Unnecessary cycles. Web user as an interface?

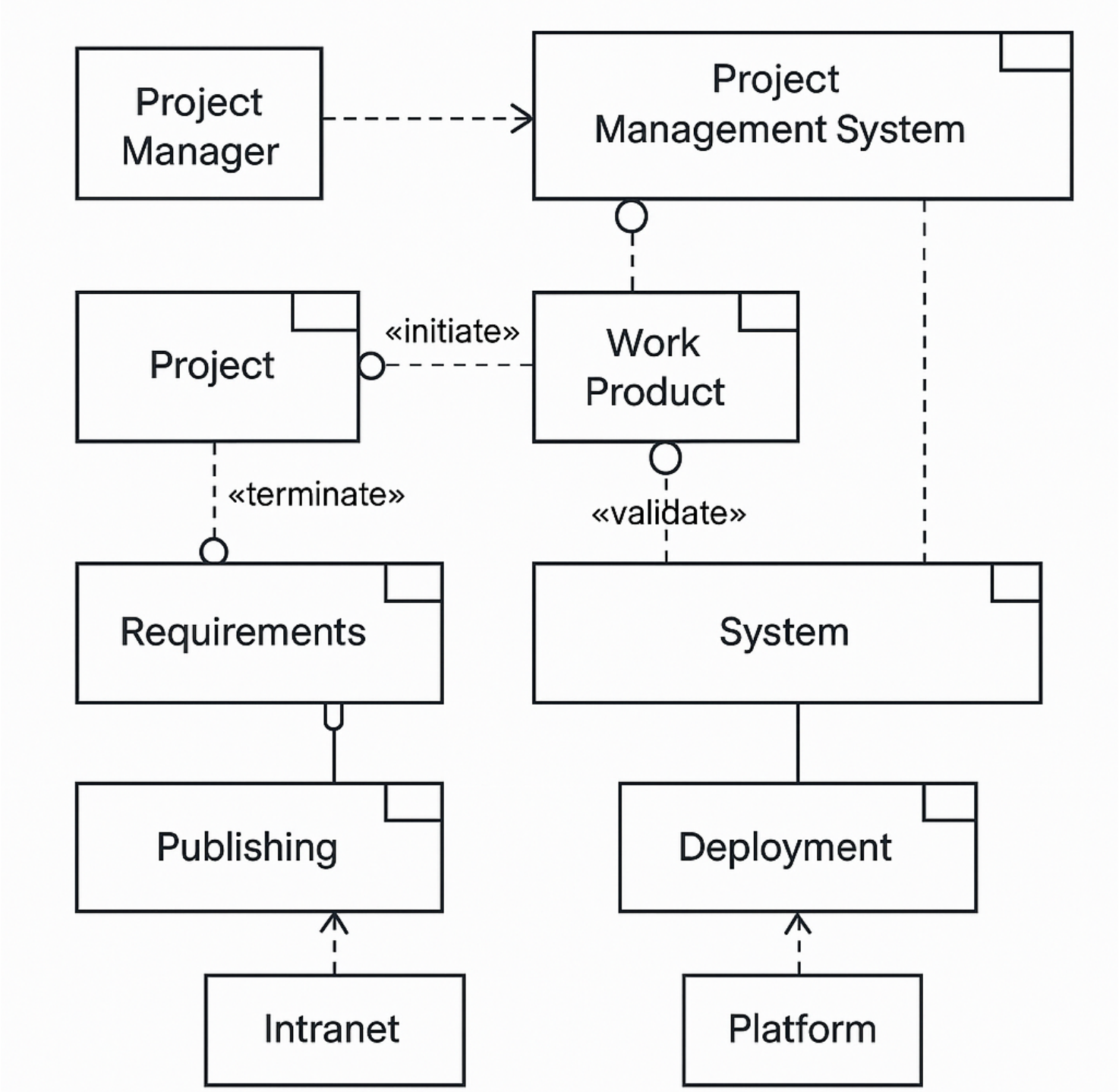
REQUIREMENTS.

Requirements:

Project Management System

A project manager uses the project management system to manage a project. The project manager leads a team to execute the project within the project's start and end dates. Once a project is created in the project management system, a manager may initiate and later terminate the project due to its completion or for some other reason. As input, a project uses requirements. As output, a project produces a system (or part of a system). The requirements and system are work products: things that are created, used, updated, and elaborated on throughout a project. Every work product has a description, is of some percent complete throughout the effort, and may be validated. However, validation is dependent on the type of work product. For example, the requirements are validated with users in workshops, and the system is validated by being tested against the requirements. Furthermore, requirements may be published using various types of media, including on an intranet or in paper form; and systems may be deployed onto specific platforms.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

Direction of calls between layers are not clearly defined, hard to say which way they go

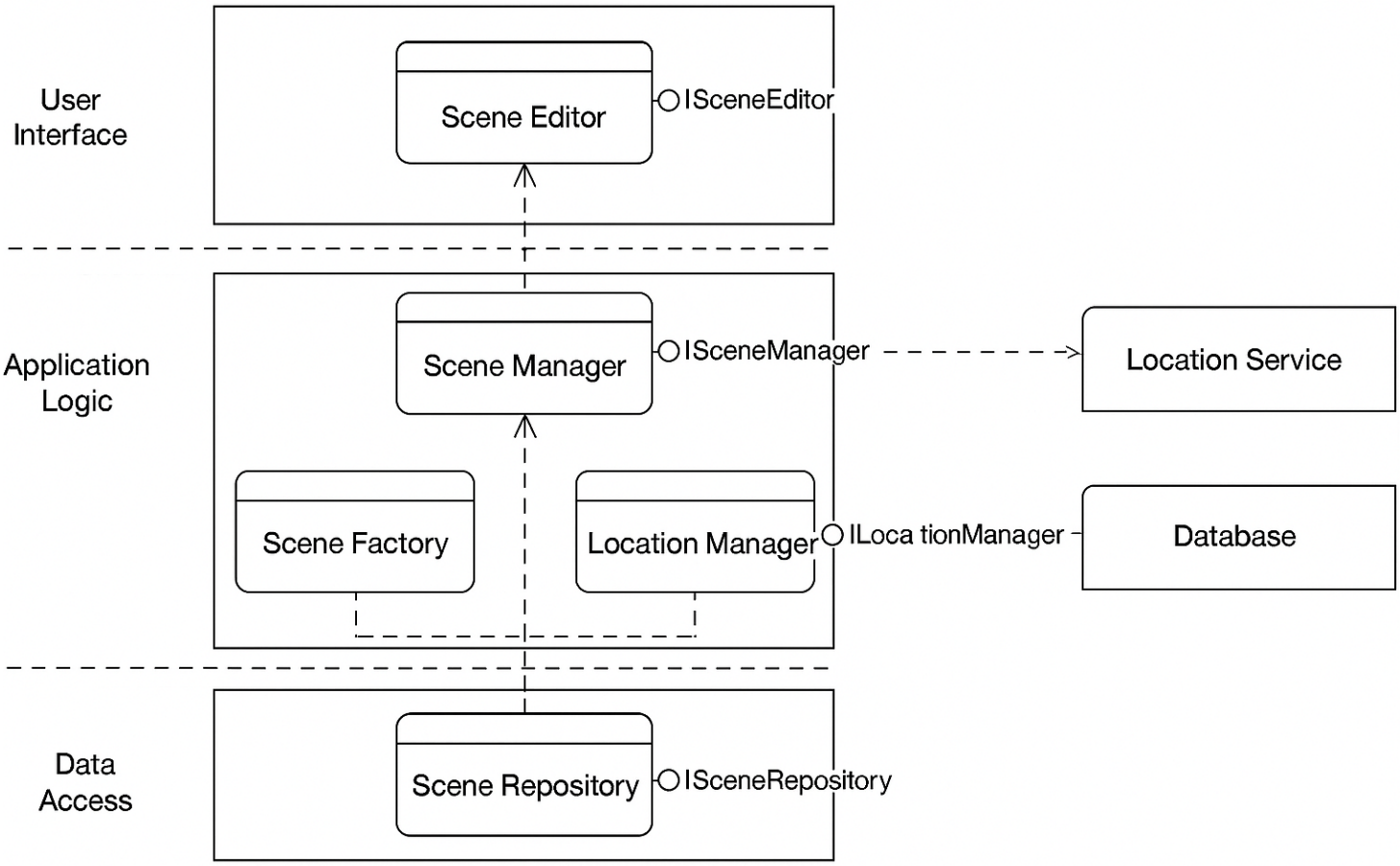
REQUIREMENTS.

Requirements:

Hollywood Approach

We are interested in building a software application to manage filmed scenes for realizing a movie, by following the so-called “Hollywood Approach”. Every scene is identified by a code (a string) and it is described by a text in natural language. Every scene is filmed from different positions (at least one), each of this is called a setup. Every setup is characterized by a code (a string) and a text in natural language where the photographic parameters are noted (e.g., aperture, exposure, focal length, filters, etc.). Note that a setup is related to a single scene. For every setup, several takes may be filmed (at least one). Every take is characterized by a (positive) natural number, a real number representing the number of meters of film that have been used for shooting the take, and the code (a string) of the reel where the film is stored. Note that a take is associated to a single setup. Scenes are divided into internals that are filmed in a theater, and externals that are filmed in a location and can either be “day scene” or “night scene”. Locations are characterized by a code (a string) and the address of the location, and a text describing them in natural language.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	◉	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	○	◉

Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	○	◉

Q2. Please provide any other comments you might have about this component diagram:

Solid start on this one. Doesn't really get into the "setups", but overall a good high level architecture. Clear layers, clear communication between

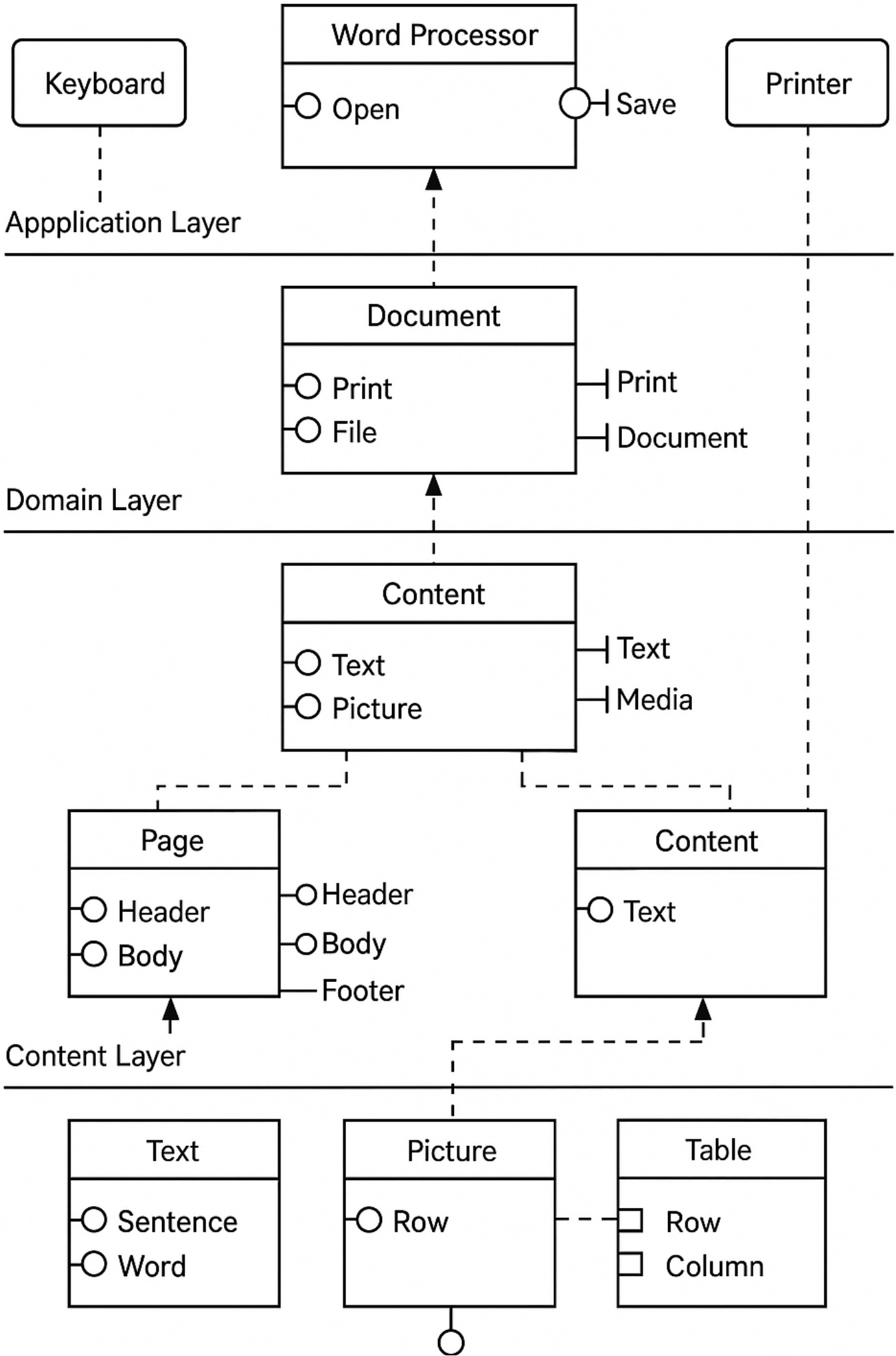
REQUIREMENTS.

Requirements:

```
Word Processor

A user can open a new or existing document. Text is entered through a keyboard. A document is
made up of several pages and each page is made up of a header, body and footer. Date, time and
page number may be added to header or footer. Document body is made up of sentences, which are
themselves made up of words and punctuation characters. Words are made up of letters, digits
and/or special characters. Pictures and tables may be inserted into the document body. Tables are
made up of rows and columns and every cell in a table can contain both text and pictures. Users
can save or print documents.
```

Component Diagram:


Keyboard
Word Processor
Open
Save
Printer
Appplication Layer
Document
Print
File
Print
Document
Domain Layer
Content
Text
Picture
Text
Media
Page
Header
Body
Header
Body
Footer
Content
Text
Content Layer
Text
Sentence
Word
Picture
Row
Table
Row
Column

File System

Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q2. Please provide any other comments you might have about this component diagram:

Layers are clear, might be a bit of weirdness in the "Domain Layer"? Some repetition

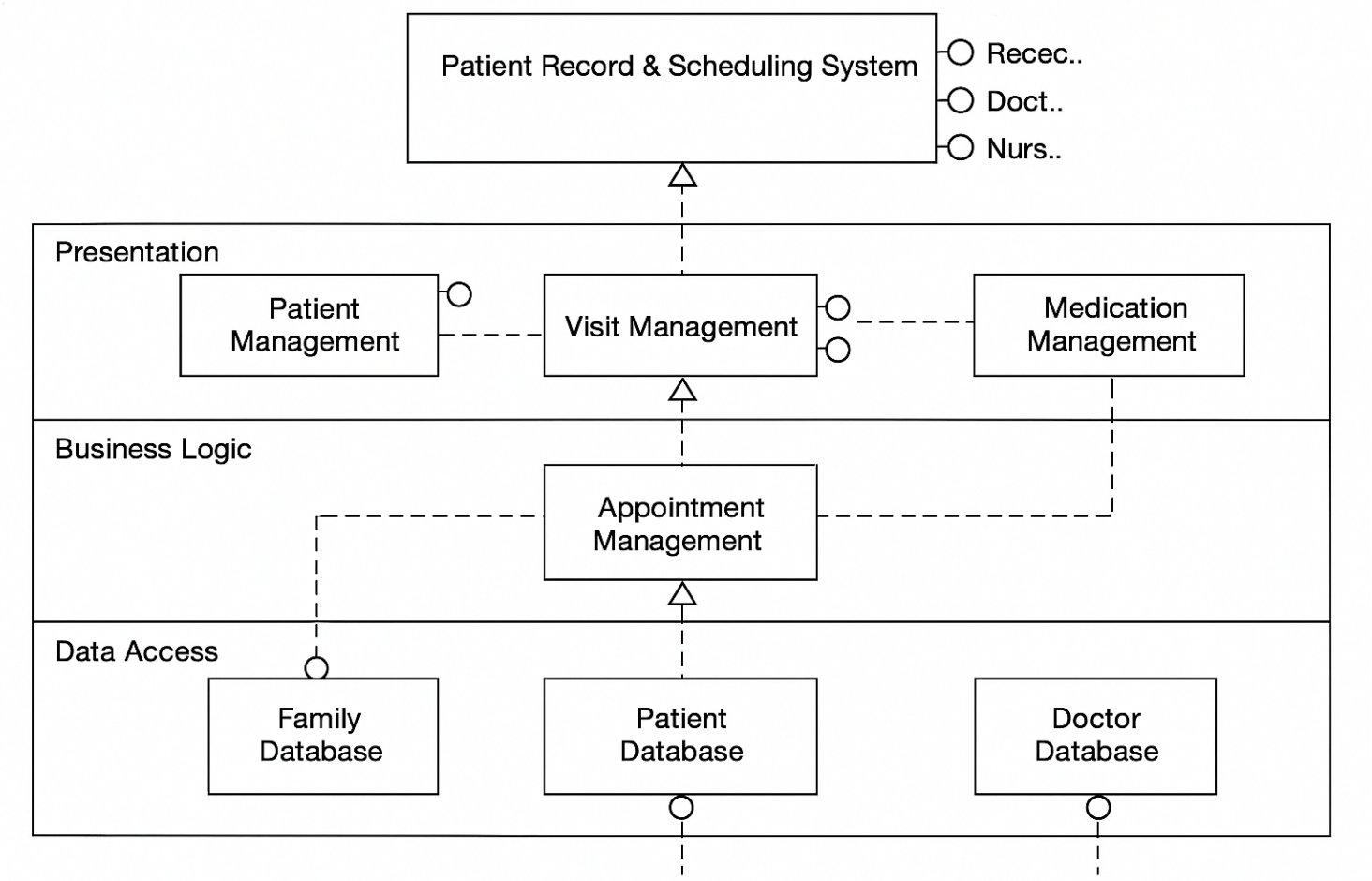
REQUIREMENTS.

Requirements:

Patient Record and Scheduling System

A patient record and scheduling system in a doctor's office is used by the receptionists, nurses, and doctors. The receptionists use the system to enter new patient information when first time patients visit the doctor. They also schedule all appointments. The nurses use the system to keep track of the results of each visit including diagnosis and medications. For each visit, free form text fields are used captures information on diagnosis and treatment. Multiple medications may be prescribed during each visit. The nurses can also access the information to print out a history of patient visits. The doctors primarily use the system to view patient history. The doctors may enter some patient treatment information and prescriptions occasionally, but most frequently they let the nurses enter this information. -- Each patient is assigned to a family. The head of family is responsible for the person with the primary medical coverage. Information about doctors is maintained since a family has a primary care physician, but different doctors may be the ones seeing the patient during the visit.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	◉	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	○	◉

Q2. Please provide any other comments you might have about this component diagram:

Good layers, generally clear. Maybe missing some mention of the diagnoses. Vestigial interfaces coming out the bottom should go away.

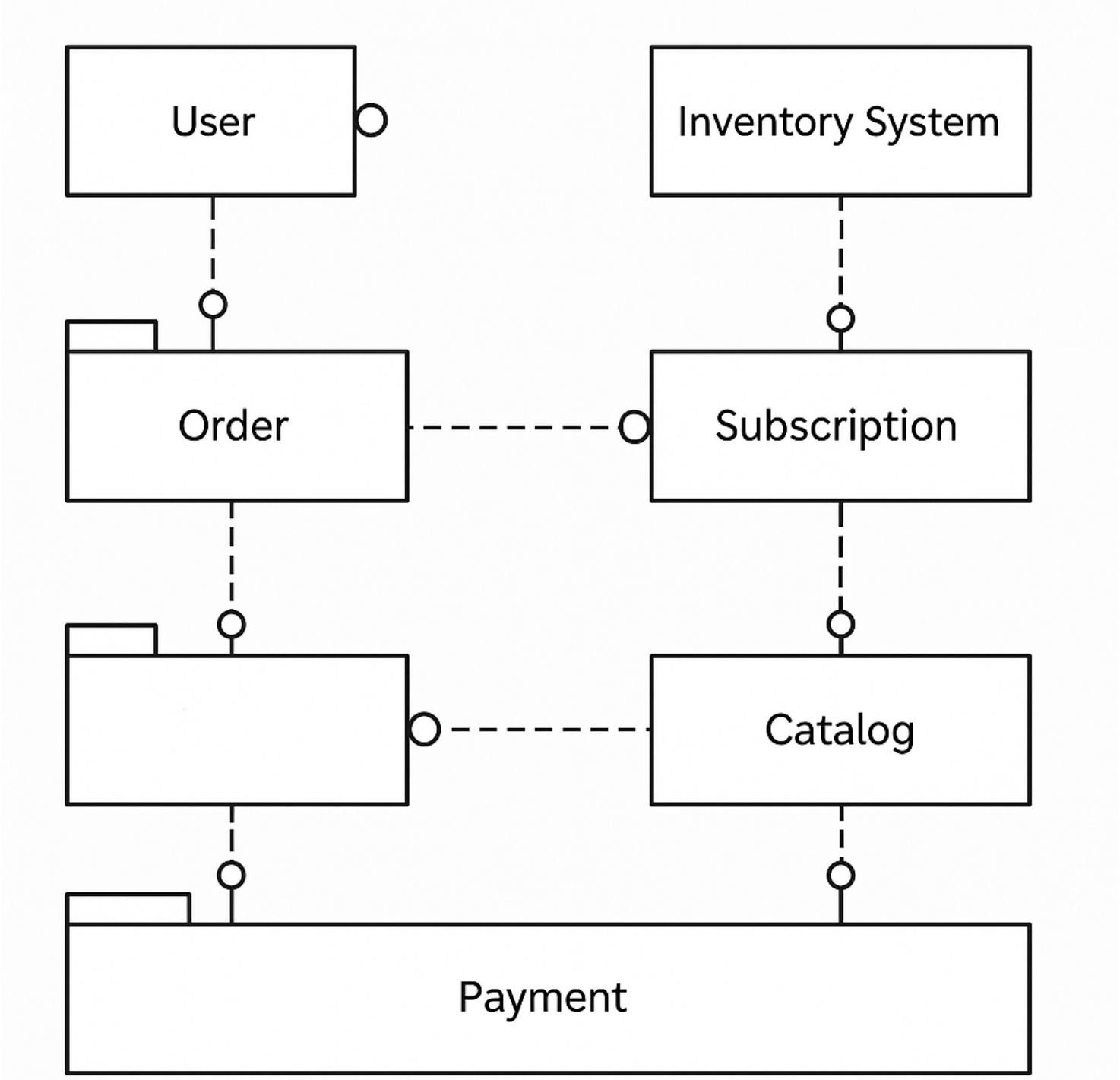
REQUIREMENTS.

Requirements:

```
Movie-Shop

♣ Design a system for a movie-shop, in order to handle ordering of movies and browsing of the
catalogue of the store, and user subscriptions with rechargeable cards. ♣ Only subscribers are
allowed hiring movies with their own card. ♣ Credit is updated on the card during rent
operations. ♣ Both users and subscribers can buy a movie and their data are saved in the related
order. ♣ When a movie is not available it is ordered .
```

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	◉	○	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

No clear layers of interaction here, just positioning. Empty boxes. Overall bad.

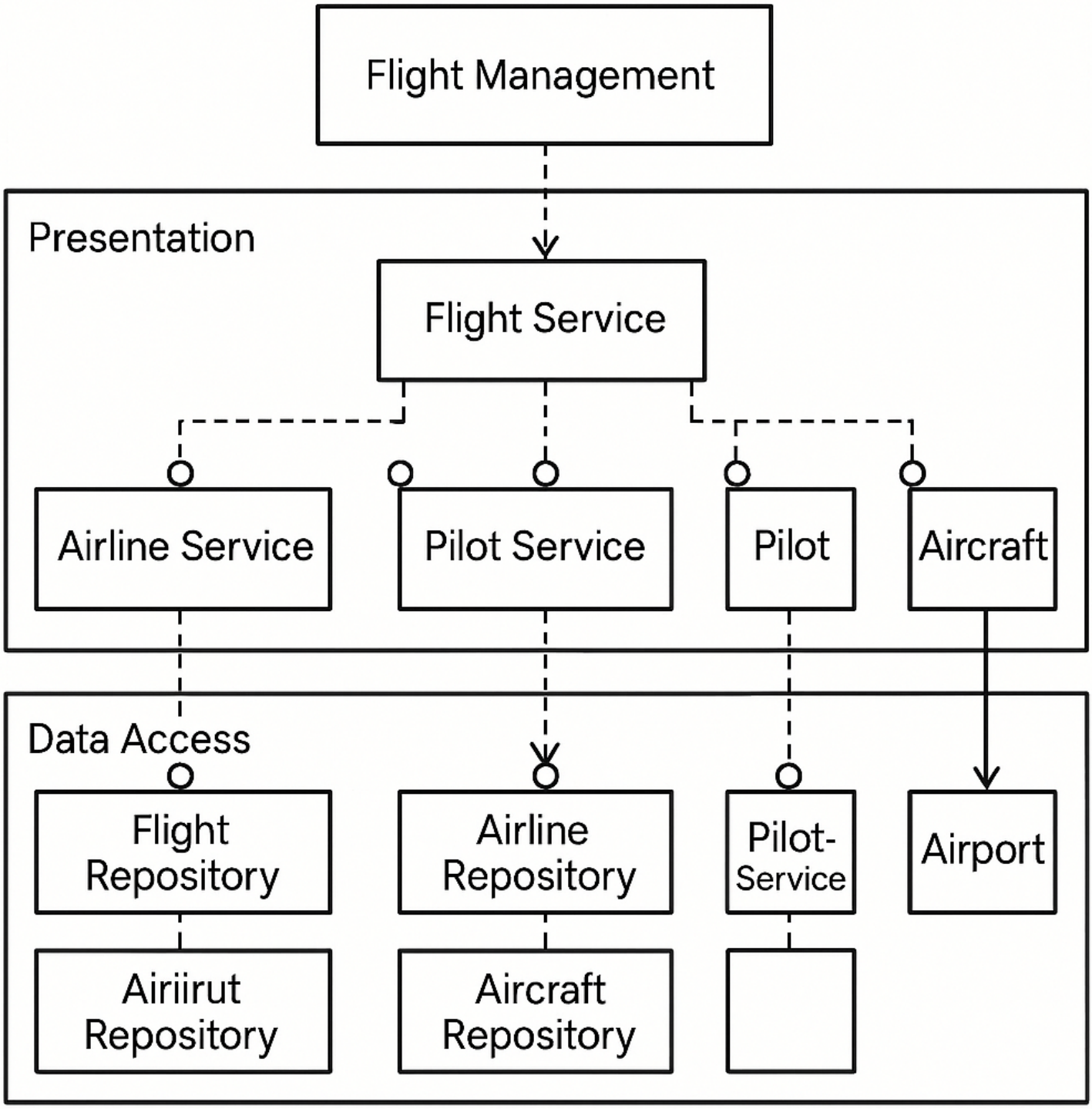
REQUIREMENTS.

Requirements:

```
Flights

We want to model a system for management of flights and pilots. An airline operates flights. Each
airline has an ID. Each flight has an ID a departure airport and an arrival airport: an airport
as a unique identifier. Each flight has a pilot and a co-pilot, and it uses an aircraft of a
certain type; a flight has also a departure time and an arrival time. An airline owns a set of
aircrafts of different types. An aircraft can be in a working state or it can be under repair. In
a particular moment an aircraft can be landed or airborne. A company has a set of pilots: each
pilot has an experience level: 1 is minimum, 3 is maximum. A type of aeroplane may need a
particular number of pilots, with a different role (e.g.: captain, co-pilot, navigator): there
must be at least one captain and one co-pilot, and a captain must have a level 3.
```

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q2. Please provide any other comments you might have about this component diagram:

Some layers, but too "thick". Calls clearly one direction. Strange boxes.

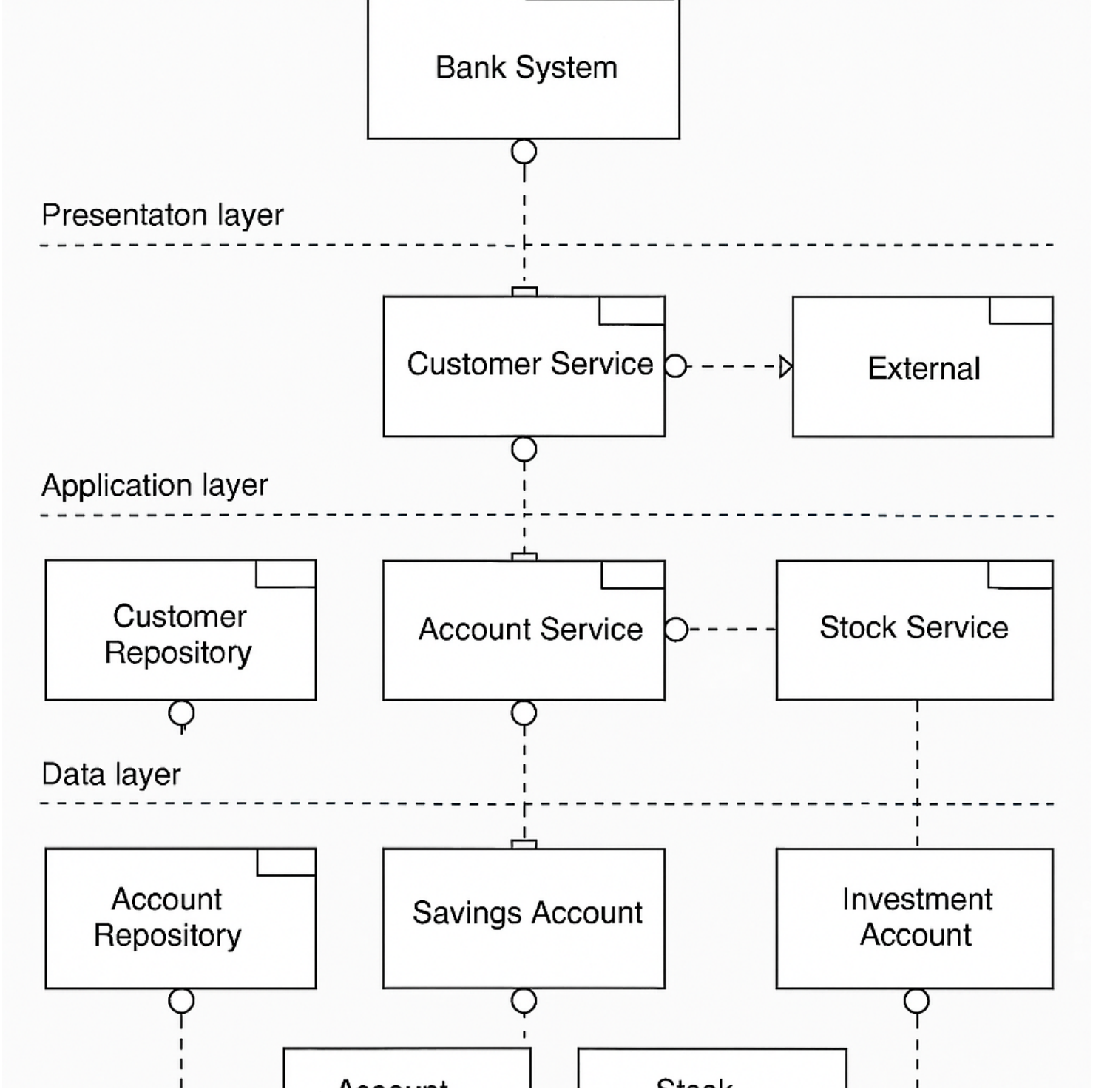
REQUIREMENTS.

Requirements:

```
Bank System

A bank system contains data on customers (identified by name and address) and their accounts.
Each account has a balance and there are 2 type of accounts: one for savings which offers an
interest rate, the other for investments, used to buy stocks. Stocks are bought at a certain
quantity for a certain price (ticker) and the bank applies commission on stock orders.
```

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q2. Please provide any other comments you might have about this component diagram:

Clear layers. Disconnected Customer Repository should be one layer down. Missing bottom of picture

REQUIREMENTS.

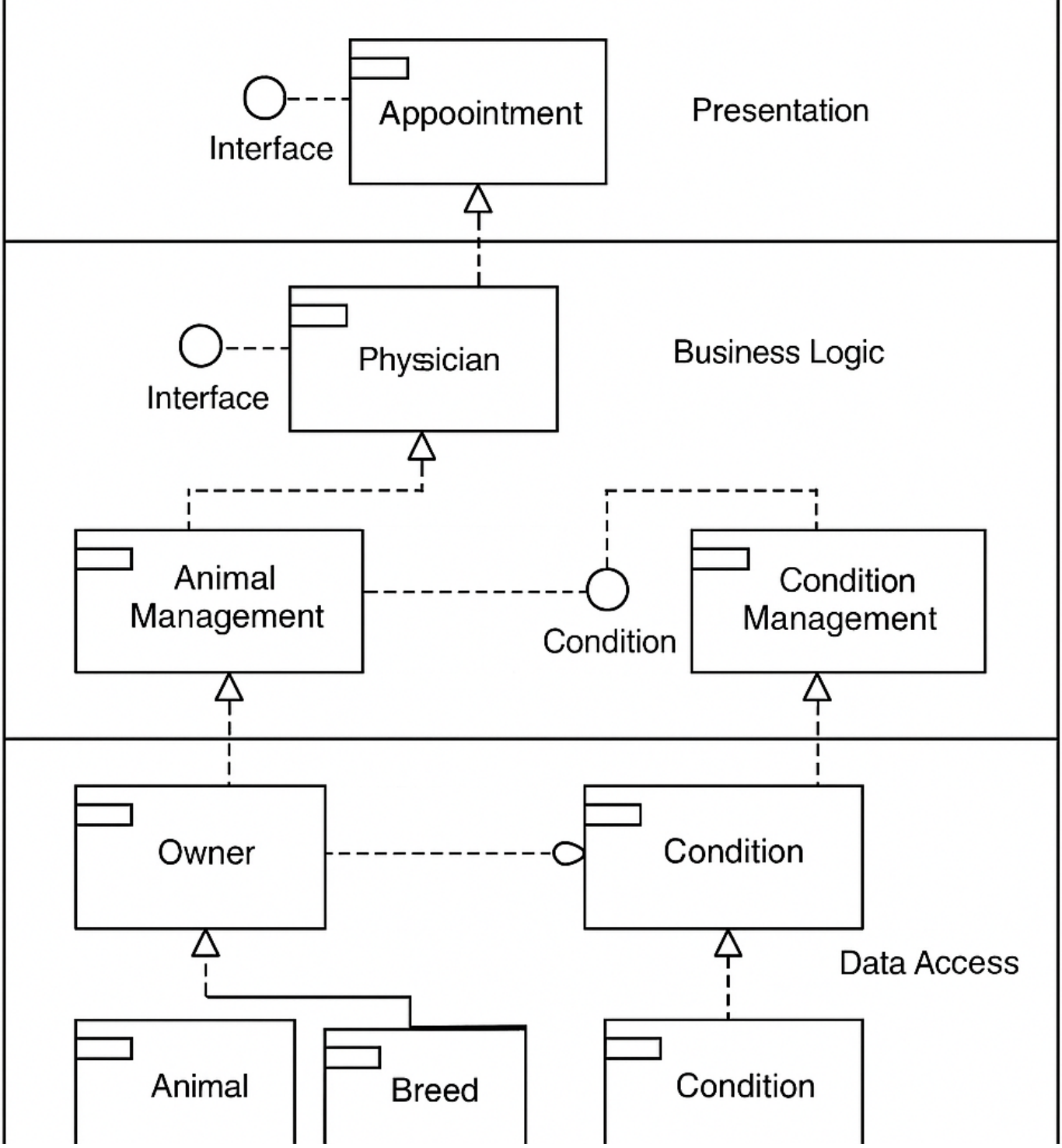
Requirements:

Veterinary Clinic

The owner of a veterinary clinic wants to create a database to store information about all veterinary services performed. After some research he came up with the following requirements:

- For each admitted animal, its name, breed (if any) and owner must be stored. Each animal should be given an unique numeric identifier.
- For each owner, its name, address and phone number should be stored. An unique numeric identifier should also be generated for each one of them.
- An animal might be owner-less. This happens frequently as the clinic often rescues abandoned dogs from the streets in order to treat them and get them new owners.
- It should be possible to store information about a specific breed even if no animals of that breed have been treated at the clinic.
- Each appointement always has a responsible physician. All appointements start at a certain date and time; and are attended by an animal (and of course its owner).
- For each physician, his name, address and phone number should be stored. An unique numeric identifier should also be generated for each one of them.
- In an appointement, several medical conditions might be detected. Each condition has a common name and a scientific name. No two conditions have the same scientific name.
- It should be possible to store information about the most common conditions for each different

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	◉	○	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q2. Please provide any other comments you might have about this component diagram:

Clear layers, pretty standard. Weird connections between components, though, text is a mess

REQUIREMENTS.

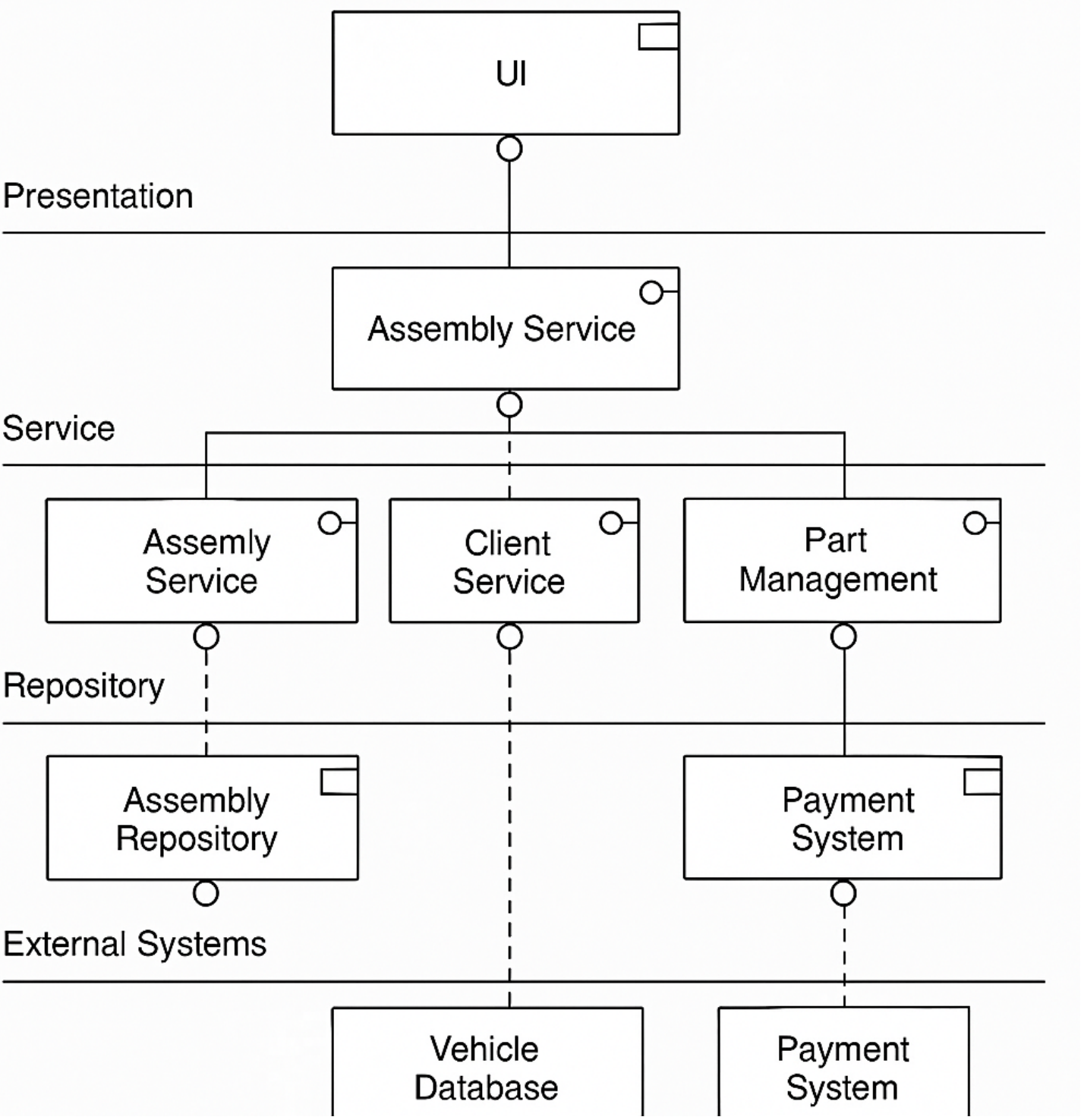
Requirements:

Auto Repair

An auto repair shop, that sells and mounts parts and accessories for all kinds of vehicles, wants a new information system to manage their clients, parts, accessories and assembly services:

- There are several employees. Each one of them has an unique identifying number, a name and an address.
- In this shop, assembly services, where parts and accessories are installed in a vehicle, are executed. For each one these services the following data must be stored: In which car the service was executed, how many kms had the car at the time, who was the responsible employee, which parts and accessories were fitted, how many work hours did it take and the admission and finish dates.
- Parts and accessories are only sold together with an assembly service.
- Each part/accessory only fits in some car models. Therefore, it is important to store that information.
- Each part/accessory has a category (radio, tyre, …), a serial number and a price.
- Each car has a license plate, a make, a model, a color and an owner. Each owner has a name, identifying number, address and a phone.
- One person can own more than one car but one car only has one owner.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

There are layers, but generally seem disconnected from components listed.

REQUIREMENTS.

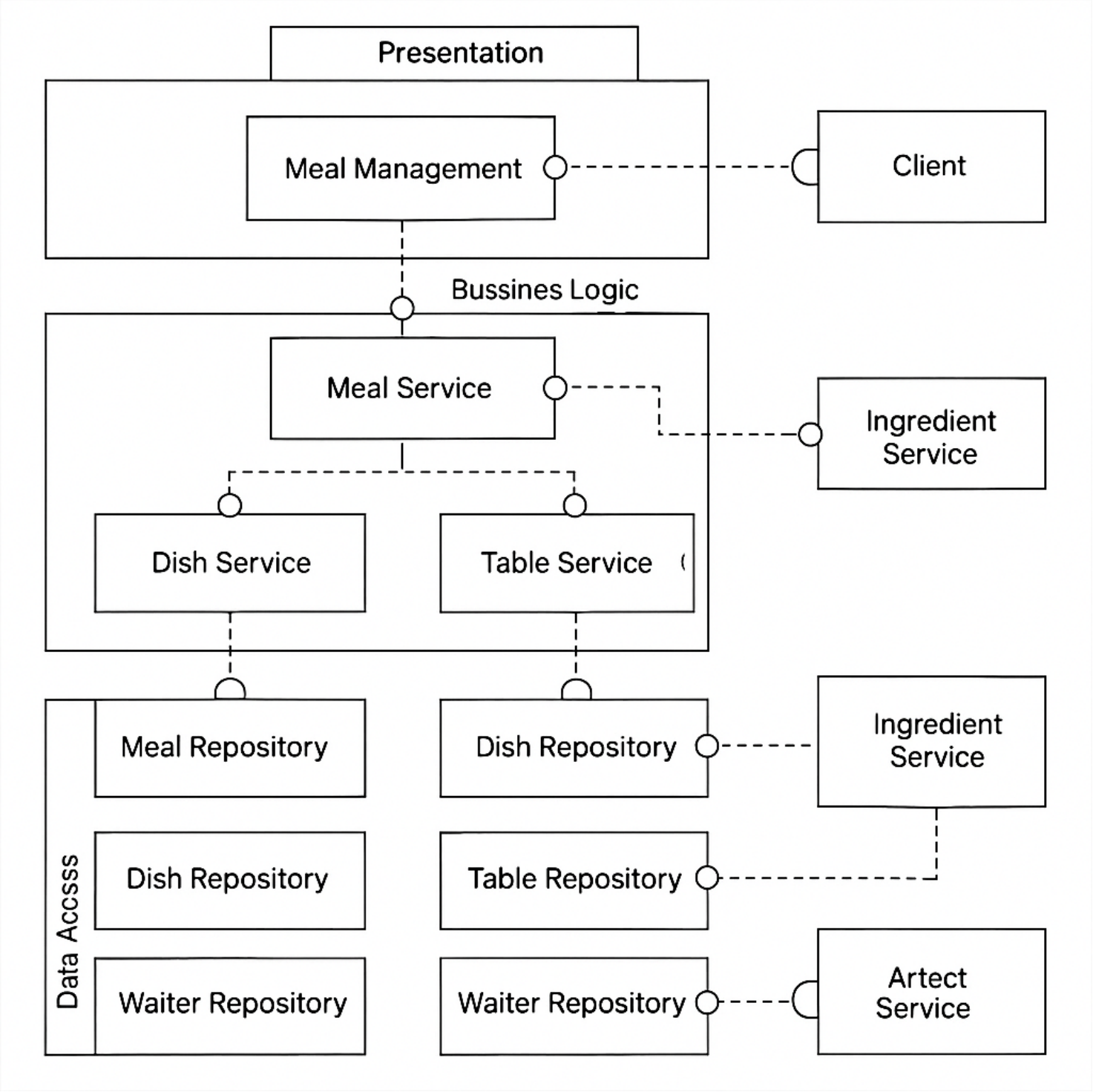
Requirements:

10. Restaurant

The owner of a small restaurant wants a new information system to store data for all meals consumed there and also to keep a record of ingredients kept in stock. After some research he reached the following requirements list:

- Each ingredient has a name, a measuring unit (e.g. olive oil is measured in liters, while eggs are unit based) and a quantity in stock. There are no two ingredients with the same name.
- Each dish is composed of several ingredients in a certain quantity. An ingredient can, of course, be used in different dishes.
- A dish has an unique name and a numeric identifier.
- There are several tables at the restaurant. Each one of them has an unique numeric identifier and a maximum ammount of people that can be seated there.
- In each meal, several dishes are consumed at a certain table. The same dish can be eaten more than once in the same meal.
- A meal takes place in a certain date and has a start and end time. Each meal has a responsible waiter.
- A waiter has an unique numerical identifier, a name, an address and a phone number.
- In some cases it is important to store information about the client that consumed the meal. A client has a tax identification number, a name and an address.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	◉	○	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

Very bad text issues. Layers seem random, not like they are made with any specific organization in mind.

REQUIREMENTS.

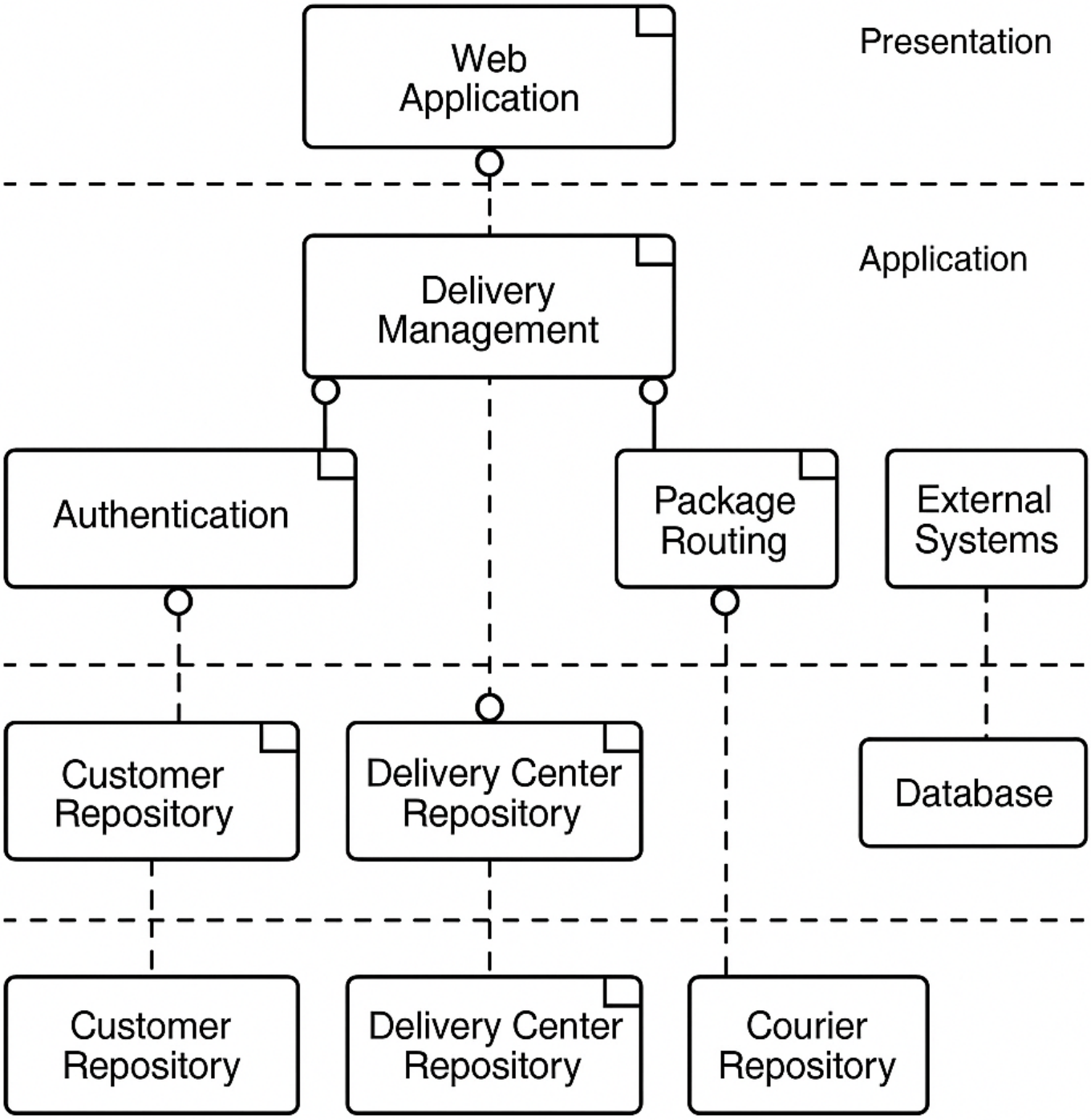
Requirements:

Deliveries

The owner of a small delivery company plans to have an information system that allows him to save data about his customers and deliveries. After some time studying the problem, he reached the following requirements:

- Each customer has a VAT number, a name, a phone number and an address. There are no two clients with the same VAT number.
- When a customer wants to send a package to another customer, he just has to login to the company website, select the customer he wants to send the package to, enter the package's weight and if the delivery is normal or urgent. He then receives an unique identifier code that he writes on the package.
- The package is then delivered by the customer at the delivery center of his choosing. A delivery center has a unique name and an address.
- Each client has an associated delivery center. This delivery center is chosen by the company and it is normally the one closest to the customer's house.
- The package is them routed through an internal system until it reaches the delivery center of the recipient.
- The package is then delivered by hand from that delivery center to the recipient by a courier.
- Couriers have a single VAT number, a name and a phone number. Each courier works in a single

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q2. Please provide any other comments you might have about this component diagram:

Would be better w/out the disconnected external service and database and the duplicated repositories.

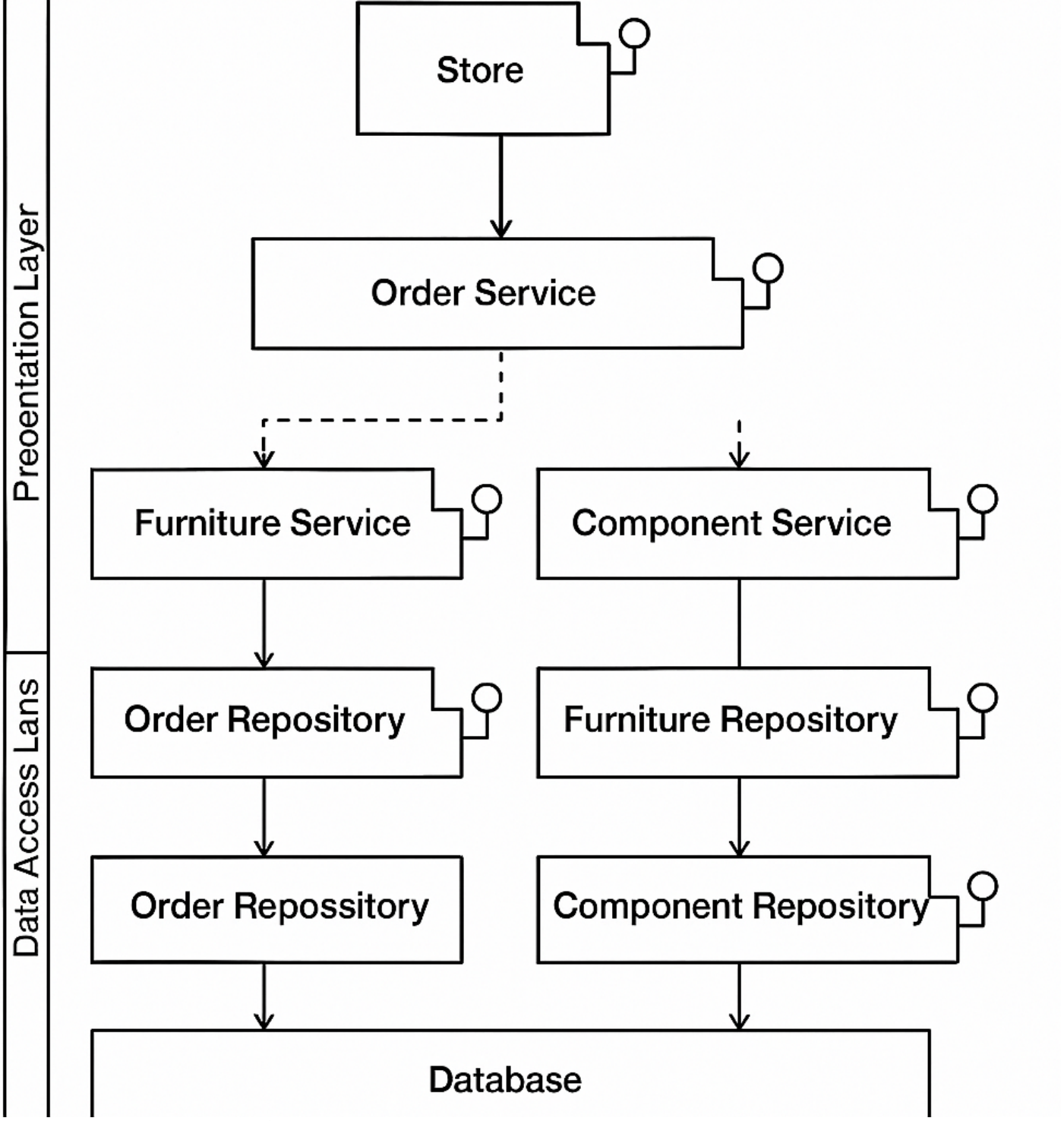
REQUIREMENTS.

Requirements:

```
Furniture

The known furniture factory Hi-Key-Ah, intends to implement an information system to store all
data on the different types of furniture and components it produces:
● The factory produces several lines of furniture, each with a different name and consisting of
several pieces of furniture of different types (beds, tables, chairs, …).
● All furniture pieces have a type, a single reference (eg CC6578) and a selling price.
● The major competitive advantage of this innovative plant is the fact that each component
produced can be used in more than one piece of furniture.
● Each piece of furniture is thus composed of several components. The same component can be used
more than once in the same piece.
● Every type of component produced is assigned a unique numerical code, a manufacturing price and
a type (screw, hinge, shelf …).
● The furniture is then sold in various stores throughout the world. Each store has a different
address and a fax number.
● To make the manufacturing process more efficient, stores have to place orders everytime they
need to replenish their stock. These orders must also be stored in the database.
● Each order has a order number, a date, the store that placed the order as well as a list of all
the ordered furniture and their quantities.
```

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q2. Please provide any other comments you might have about this component diagram:

Layers generally present and clear. Services in one, repositories in another. Use of the component diagram symbols seems strange.

REQUIREMENTS.

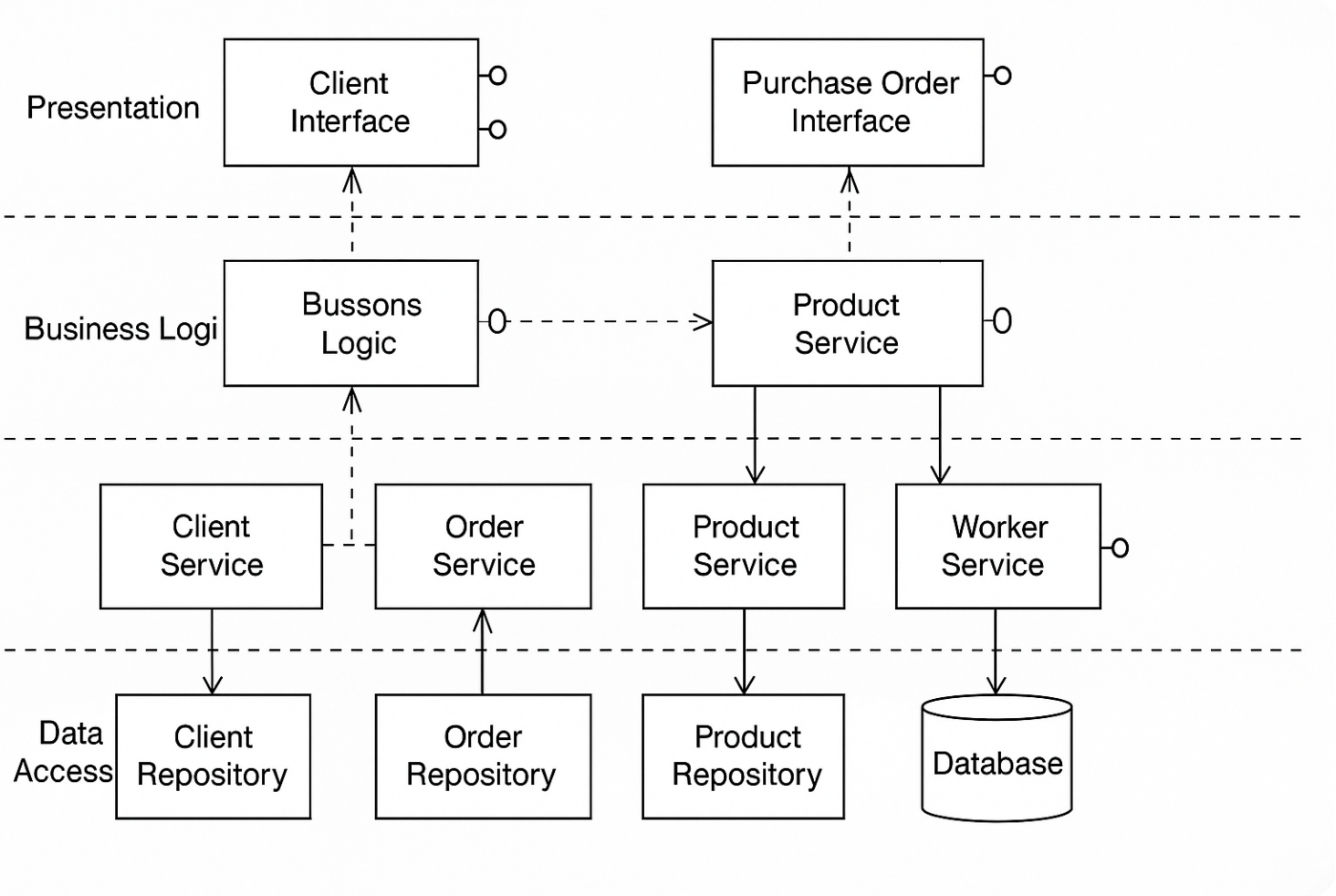
Requirements:

Factory

Create a database for a factory with the following requirements. Don’t forget to add unique identifiers for each one of the entities if needed.

- A factory has several machines. Each one of them is operated by several workers.
- A worker might work in more than one machine.
- In this factory, several products of different types, are produced. Each different type of product is produced in a single machine. But, the same machine can produce more than one type of product.
- Products from the same type are all produced from the same single material and have the same weigth.
- Clients can issue purchase orders. Each order has a list of the desired products and their quantity.
- For each worker, the following data should be stored in the database: name (first and last), birth date, address and a list of his skills.
- For each machine, the following data should be stored: serial number, make, model and purchase date.
- For each client, the followig data should be stored: name, address, phone number and name of the contact person (if any).

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○

Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	◉	○	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q2. Please provide any other comments you might have about this component diagram:

Clear layers, but the components are all weird.

REQUIREMENTS.

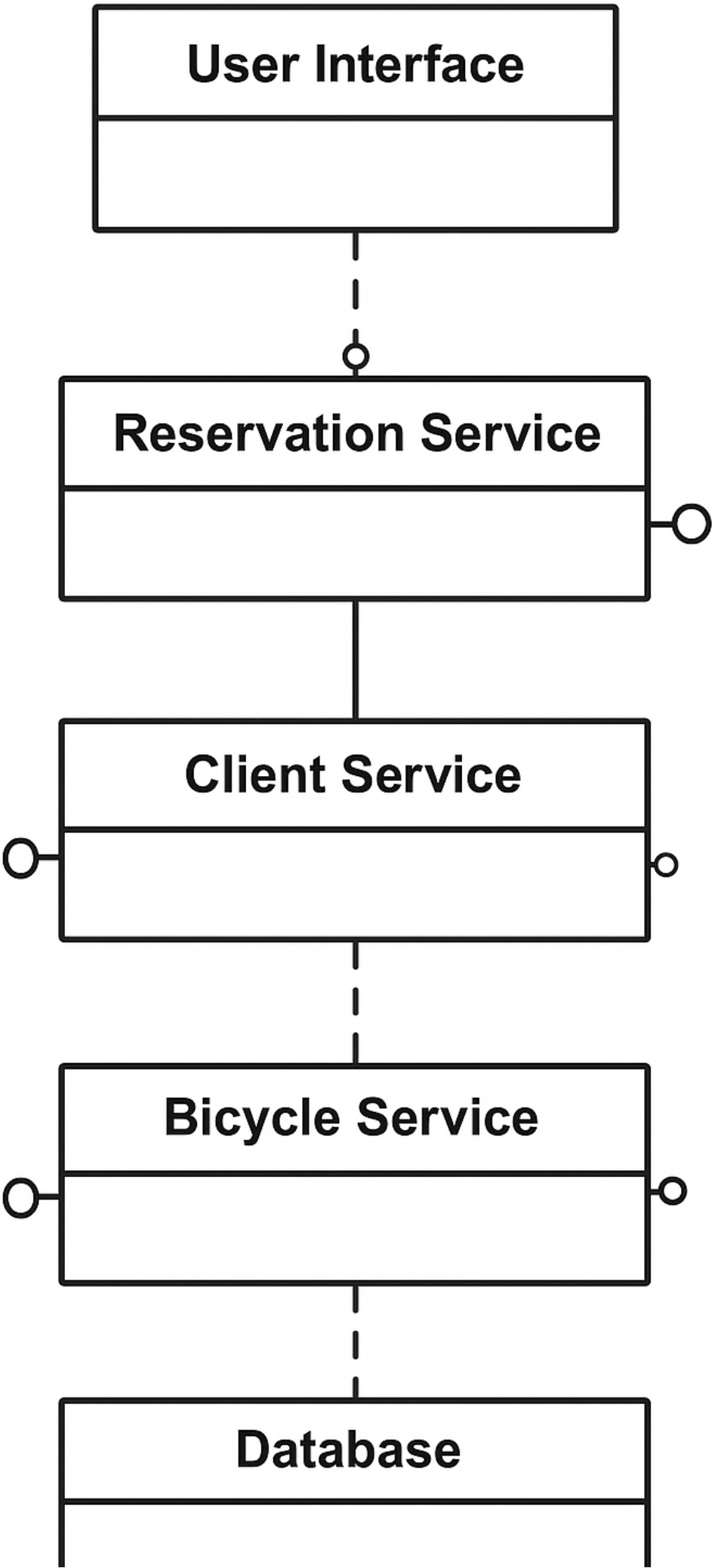
Requirements:

Bycicle Rental

A bicycle renting company wants to create an information system that allows it to store the data regarding all their reservations and rentals. The system should follow these requirements:

- It should be possible to store the national id number (NIN), tax identification number (TIN), name and address for every client. The NIN and TIN must be different for every client and all clients should have at least a TIN and a name.
- The database should also contain information about the bicycle models that can be rented- Each model has an unique name, a type (that can only be road, mountain, bmx or hybrid) and the number of gears.
- Each bicycle has a unique identifying number and a model.
- The company has several different stores where bicycles can be picked up and returned. Each one of these stores is identified by an unique name and has an address (both mandatory).
- When a reservation is made, the following data must be known: which client did the reservation, when will he pick up the bike (day), which bike model he wants and where will he pick up the bike (store).
- When a bike is picked up, the actual bike that was picked up must be stored in the database.
- When a bike is returned, the return date should be stored in the database.

Component Diagram:


User Interface
Reservation Service
Client Service
Bicycle Service
Database

Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

No real layers, just a chain of services

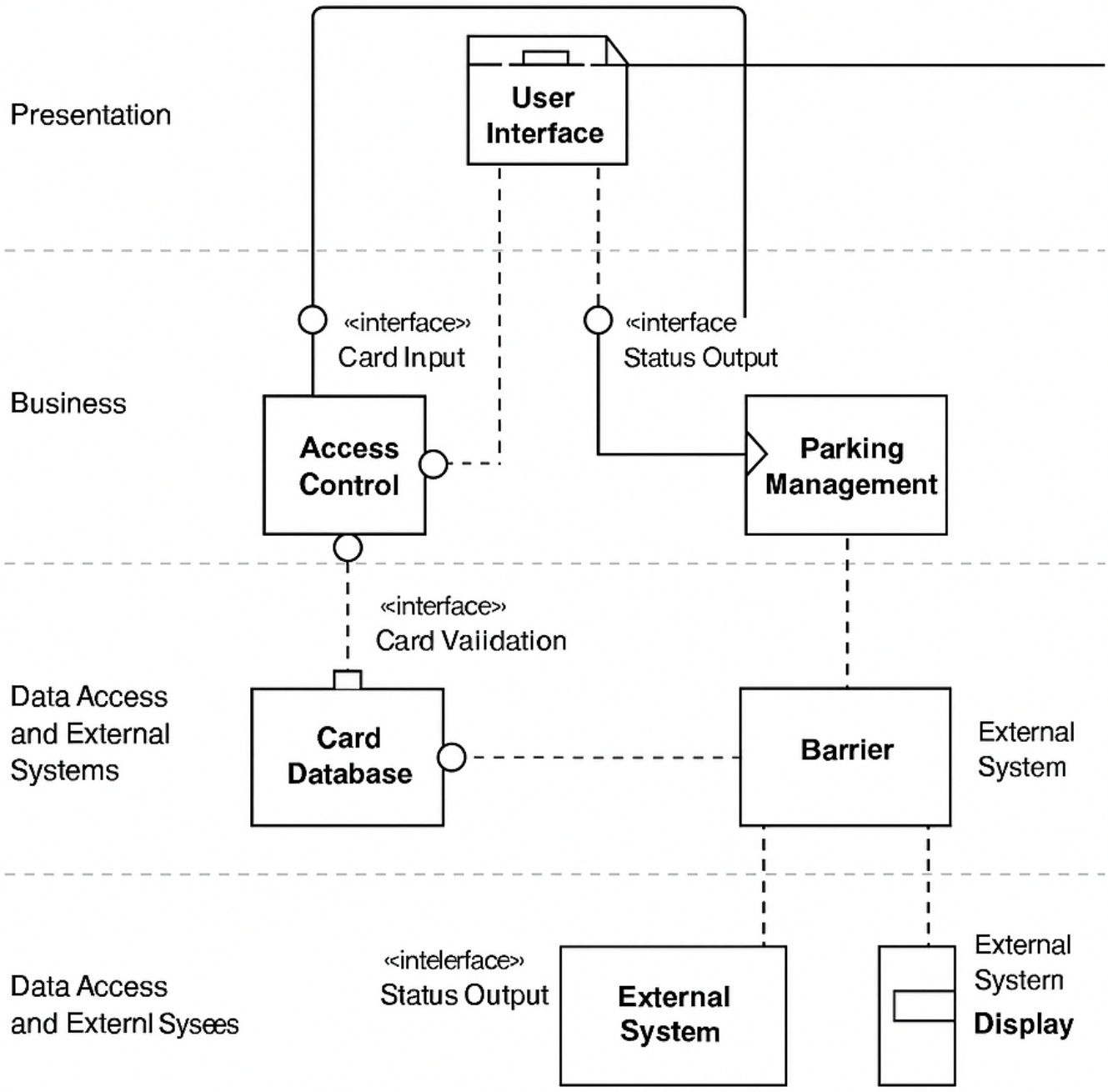
REQUIREMENTS.

Requirements:

Saturn Int. Management

Saturn Int. management wants to improve their security measures, both for their building and on site. They would like to prevent people who are not part of the company to use their car park. Saturn Int. has decided to issue identity cards to all employees. Each card records the name, department and number of a company staff, and give them access to the company car park. Employees are asked to wear the cards while on the site.
There is a barrier and a card reader placed at the entrance to the car park. When a driver drives his car into the car park, he/she inserts his or her identity card into the card reader. The card reader then verify the card number to see if it is known to the system. If the number is recognized, the reader sends a signal to trigger the barrier to rise. The driver can then drive his/her car into the car park.
There is another barrier at the exit of the car park, which is automatically raised when a car wishes to leave the car park.
A sign at the entrance display "Full" when there are no spaces in the car park. It is only switched off when a car leaves.
There is another type of card for guests, which also permits access to the car park. The card records a number and the current date. Such cards may be sent out in advance, or collected from reception. All guest cards must be returned to reception when the visitor leaves Saturn Int.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

Layers are listed but don't seem to mean anything. Strange lines going nowhere

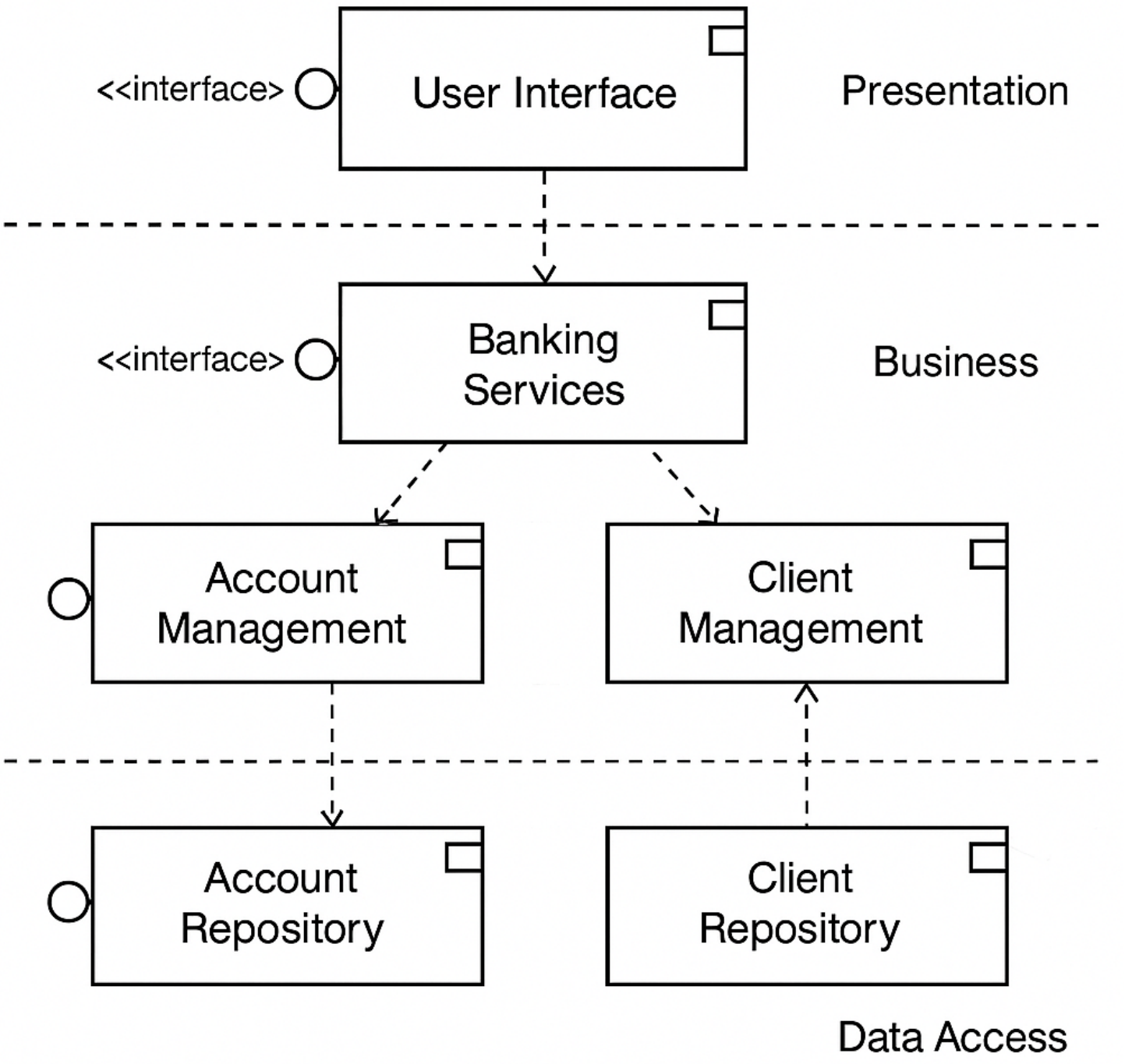
REQUIREMENTS.

Requirements:

OOBank

This system provides the basic services to manage bank accounts at a bank called OOBank. OOBank has many branches, each of which has an address and branch number. A client opens accounts at a branch. Each account is uniquely identified by an account number; it has a balance and a credit or overdraft limit. There are many types of accounts, including: a mortgage account (which has a property as collateral), a checking account, and a credit card account (which has an expiry date and can have secondary cards attached to it). It is possible to have a joint account (e.g. for a husband and wife). Each type of account has a particular interest rate, a monthly fee and a specific set of privileges (e.g. ability to write checks, insurance for purchases etc.). OOBank is divided into divisions and subdivisions (such as Planning, Investments and Consumer); the branches are considered subdivisions of the Consumer Division. Each division has a manager and a set of other employees. Each customer is assigned a particular employee as his or her 'personal banker'.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q2. Please provide any other comments you might have about this component diagram:

Nothing about divisions or branches? Pretty clear, though. Just a bit incomplete. Clear layers

REQUIREMENTS.

Requirements:

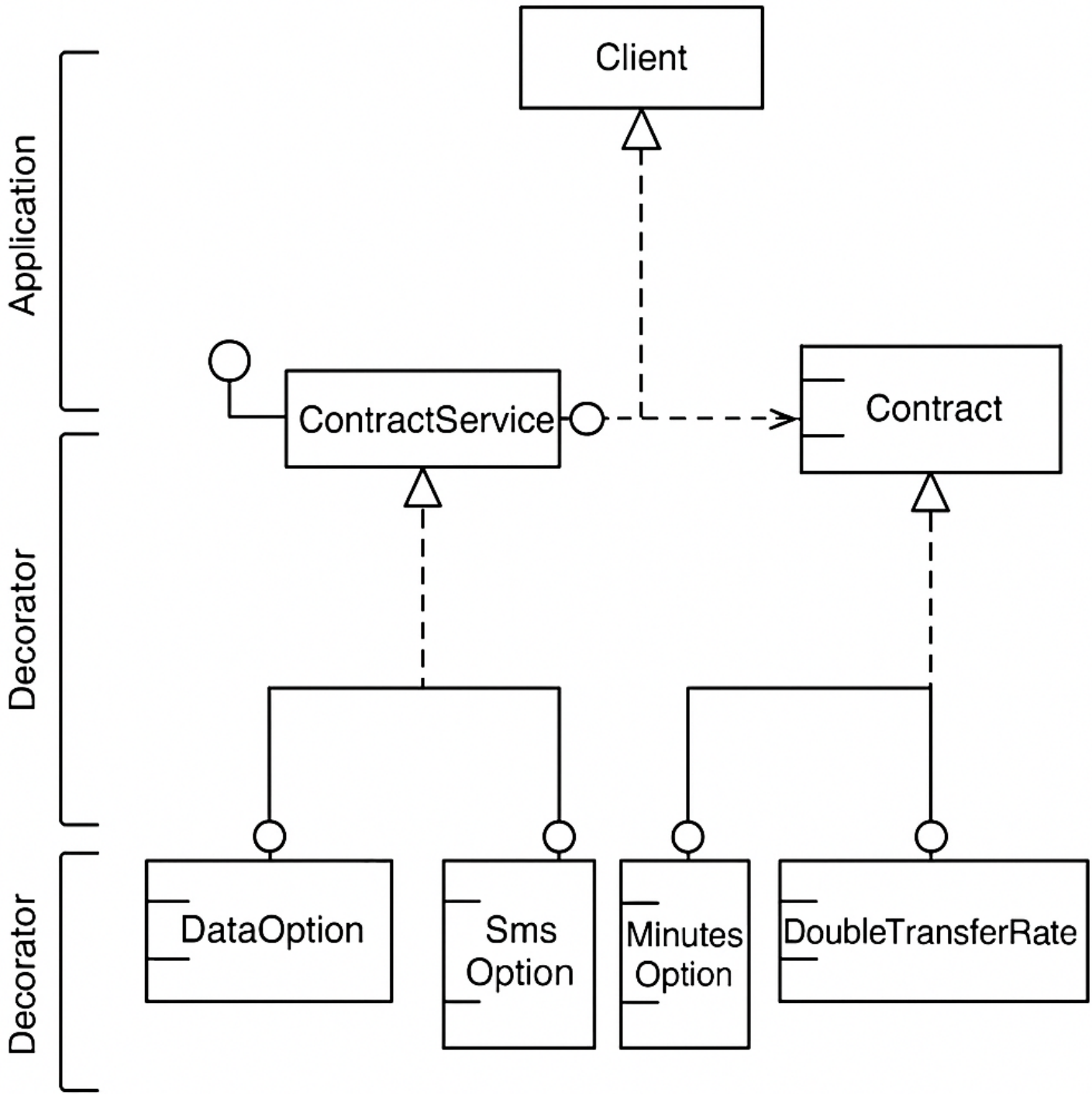
Prepaid Cell Phone

The contract of a prepaid cell phone should be modelled and implemented. A basic contract has a contract number (of type int) and a balance (of type double), but no monthly charges. The contract number is not automatically generated, but is to be set as a parameter by the constructor as well as the initial balance. The balance has a getter and a setter. The following options can be added to a contract (if needed also several times):

- 100 MB of data (monthly charge 1.00€)
- 50 SMS (monthly charge 0.50€)
- 50 minutes (monthly charge 1.50€)
- Double Transfer Rate (monthly charge 2.00€) implement this requirement with the help of the decorator pattern. All contract elements should be able to understand the methods getCharges():double, getBalance():double and setBalance(double).

The method getCharges() should provide the monthly charge of a contract with all its options selected. The methods getBalance() and setBalance() should be passed through and access the basic contract.

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

Looks like inheritance relationships across layers, not interfaces. Strange lines throughout

REQUIREMENTS.

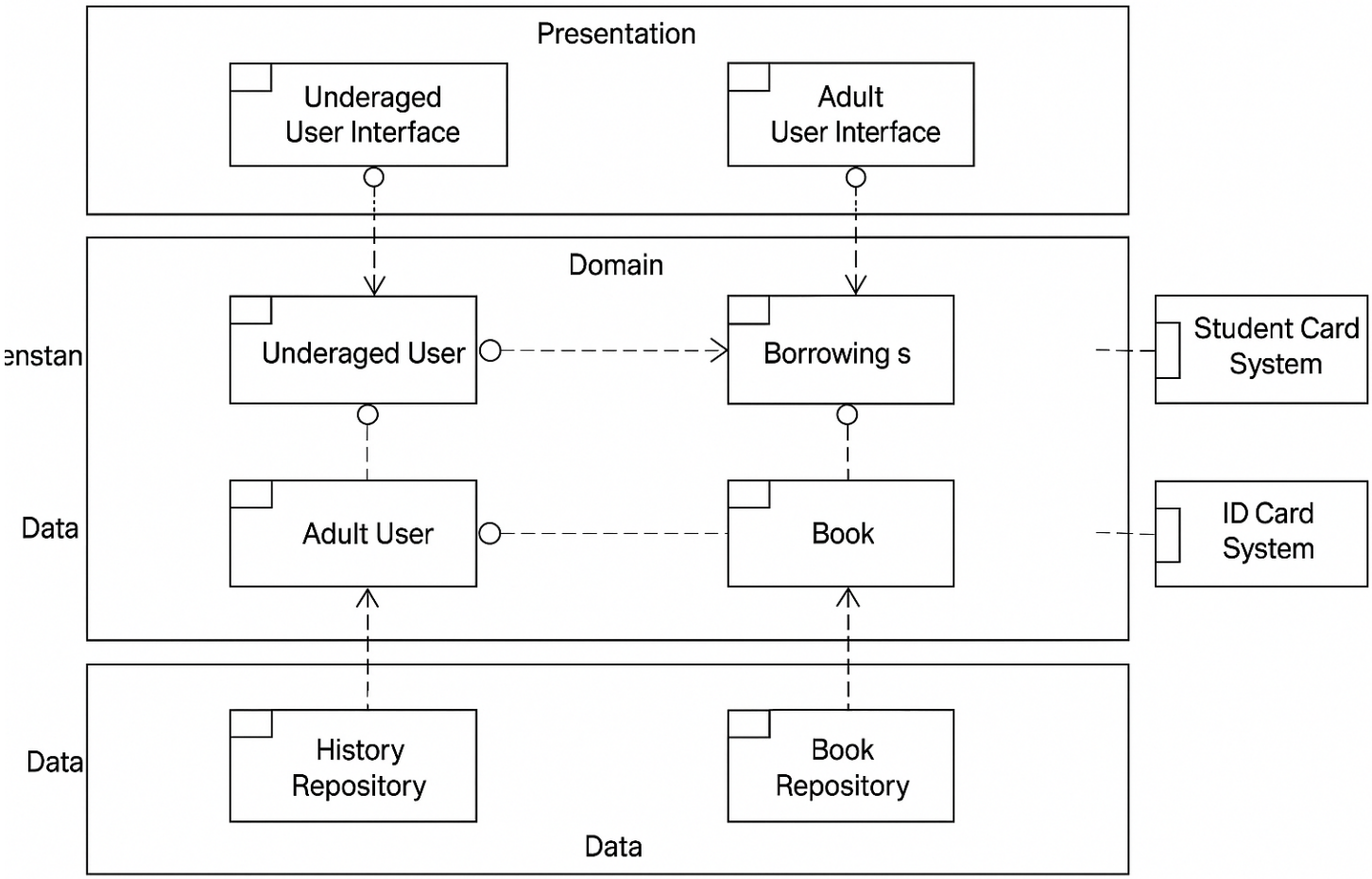
Requirements:

Library System

The exercise id to design a class structure for a library system. It should fulfil those requirements:

- There are two type of users – under-aged and adults.
- Under-aged users are identified with usage of their full name and student card.
- Adult users are identified with usage of their full name and ID card.
- The library contains books.
- There is basic information about every book (title, author, etc).
- The user can borrow at most 4 books at the same time.
- There is a history of previously borrowed books for every user (along with all the dates).

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q2. Please provide any other comments you might have about this component diagram:

Layers generally clear, cards on the right side are odd

REQUIREMENTS.

Requirements:

MyDoctor

The MyDoctor application aims to be a management tool for the appointments of a doctor.
A hospital has multiple offices.
The users of the application can be doctors and patients.
The doctors can apply to practice in offices and create a schedule for an office. The schedules in different offices can't overlay.

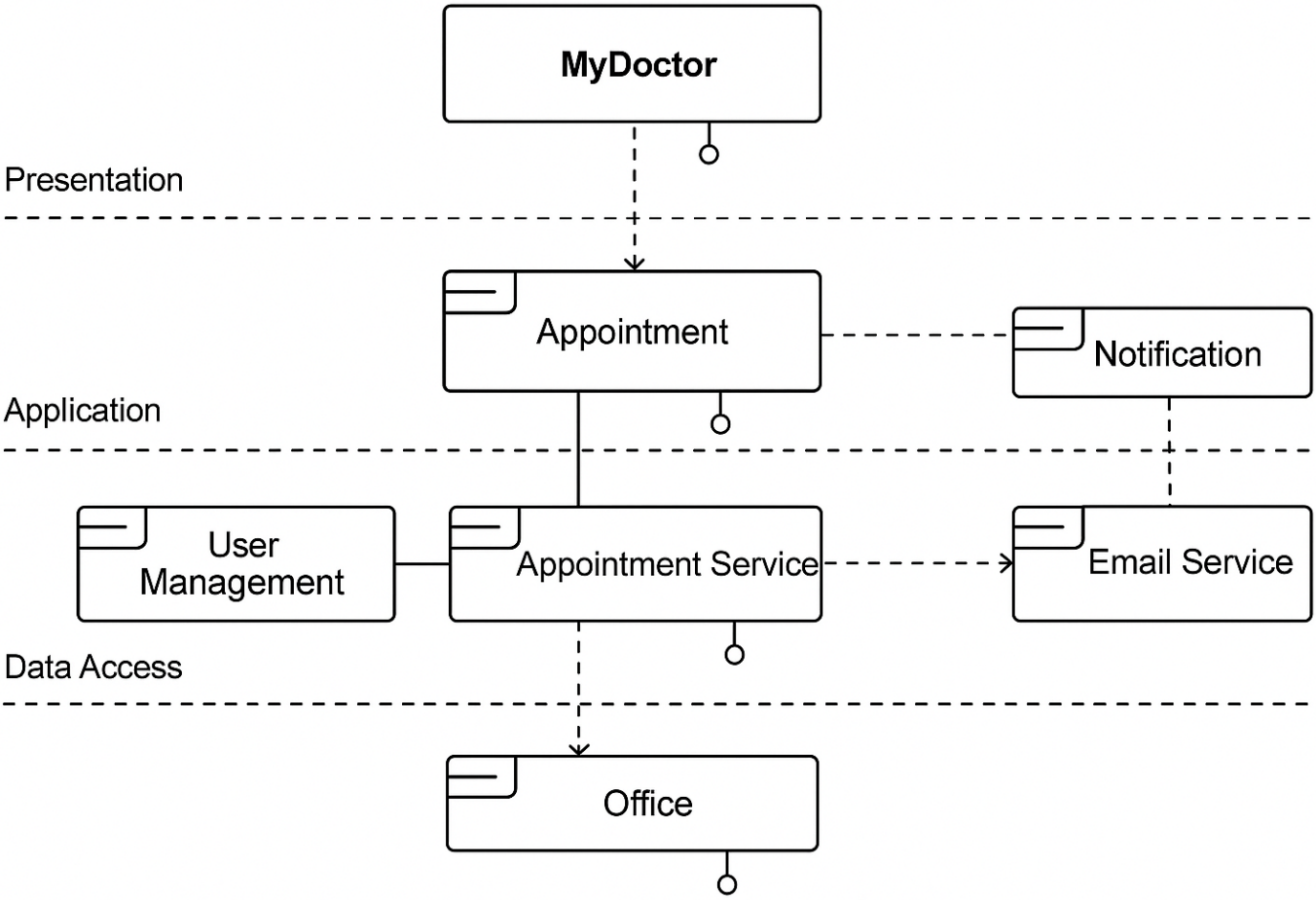
📝 Example:
Doctor Ana is available in Office 4 on the 4th of September during 1 PM - 5PM.
Doctor Ana can't practice in Office 5 on the 4th of September during 3PM - 8 PM, but she can practice in Office 5 on the 4th of September during 5:30PM - 8 PM.
The patients can see the existing doctors in the system, the schedule of the offices and can book appointments for specific doctors and for specific schedules. The appointments can be of 3 types:

- Blood Test - 15 mins
- Consultation - 30 mins
- Surgery - 60 mins

The booking of an appointment will not be possible if another appointment is already booked at the same time frame. An email is sent to the patient with the confirmation of the appointment.

📝 Example:

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○

Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q2. Please provide any other comments you might have about this component diagram:

Clear layers, the "office" at the bottom is strange

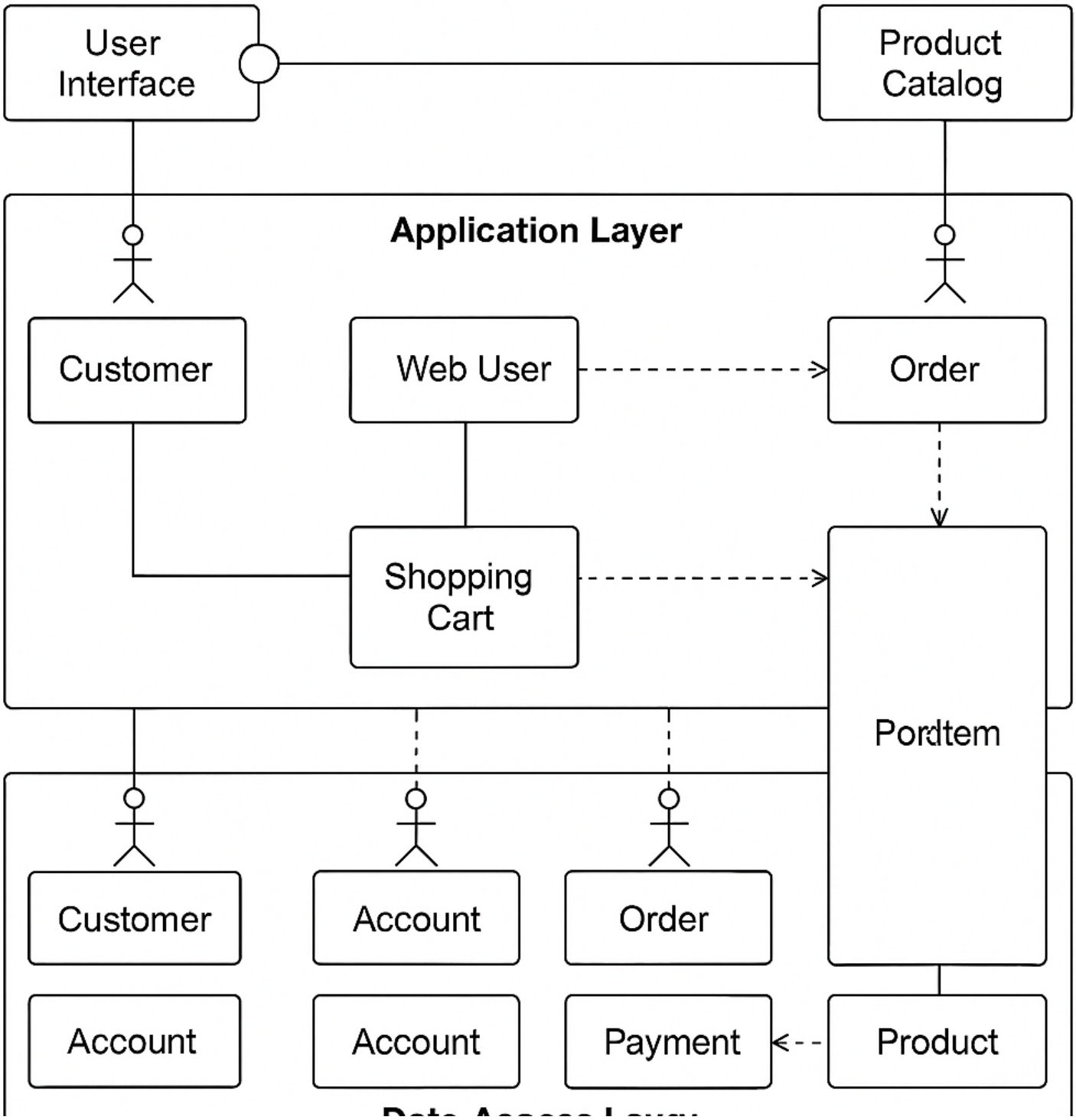
REQUIREMENTS.

Requirements:

```
Online Shopping

Each customer has unique id and is linked to exactly one account. Account owns shopping cart and
orders. Customer could register as a web user to be able to buy items online. Customer is not
required to be a web user because purchases could also be made by phone or by ordering from
catalogues. Web user has login name which also serves as unique id. Web user could be in several
states - new, active, temporary blocked, or banned, and be linked to a shopping cart. Shopping
cart belongs to account.
Account owns customer orders. Customer may have no orders. Customer orders are sorted and unique.
Each order could refer to several payments, possibly none. Every payment has unique id and is
related to exactly one account.
Each order has current order status. Both order and shopping cart have line items linked to a
specific product. Each line item is related to exactly one product. A product could be associated
to many line items or no item at all.
```

Component Diagram:



Requested Architectural Style:

Layered architecture: A layered / tiered architecture primarily comprises a set of well-defined "layers" with each component / service assigned to a specific layer. In a "closed" layered architecture, components are constrained to only require services provided by components in "lower" layers and expose interfaces for use by services in the "higher" layers.

- Components should be arranged in well-defined layers
- Components in each layer should only require interfaces from components in lower layers
- Components in each layer should expose interfaces used by components in the higher layers

Q1. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	◉	○	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	◉	○	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q2. Please provide any other comments you might have about this component diagram:

This one is just a mess. Boxes across multiple layers? Users in there like a use case? Bad.

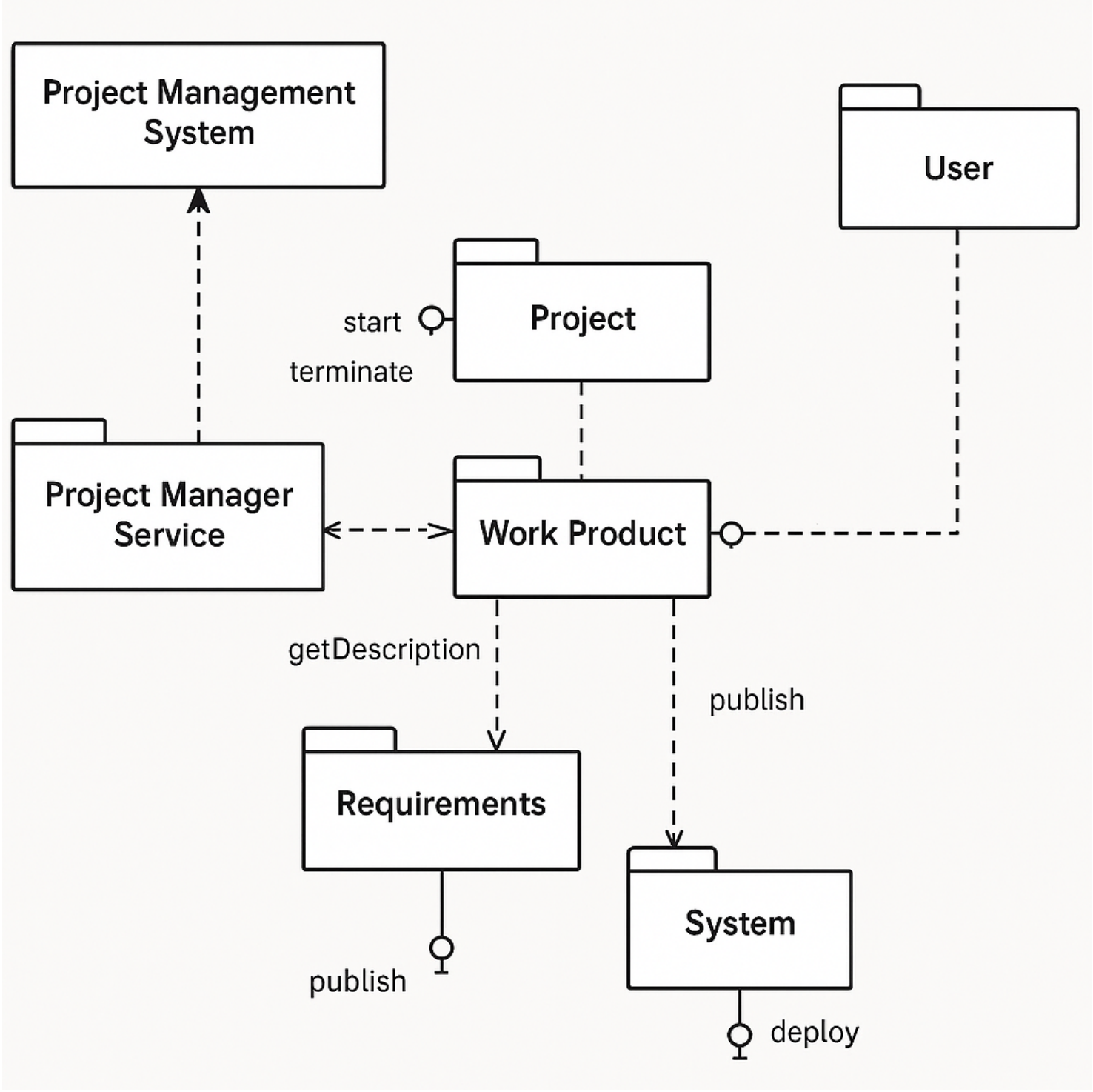
Q6.

Requirements:

```
Project Management System

A project manager uses the project management system to manage a project. The project manager
leads a team to execute the project within the project's start and end dates. Once a project is
created in the project management system, a manager may initiate and later terminate the project
due to its completion or for some other reason. As input, a project uses requirements. As output,
a project produces a system (or part of a system). The requirements and system are work products:
things that are created, used, updated, and elaborated on throughout a project. Every work
product has a description, is of some percent complete throughout the effort, and may be
validated. However, validation is dependent on the type of work product. For example, the
requirements are validated with users in workshops, and the system is validated by being tested
against the requirements. Furthermore, requirements may be published using various types of
media, including on an intranet or in paper form; and systems may be deployed onto specific
platforms.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Looks like microservices. Connections between components do not match requirements

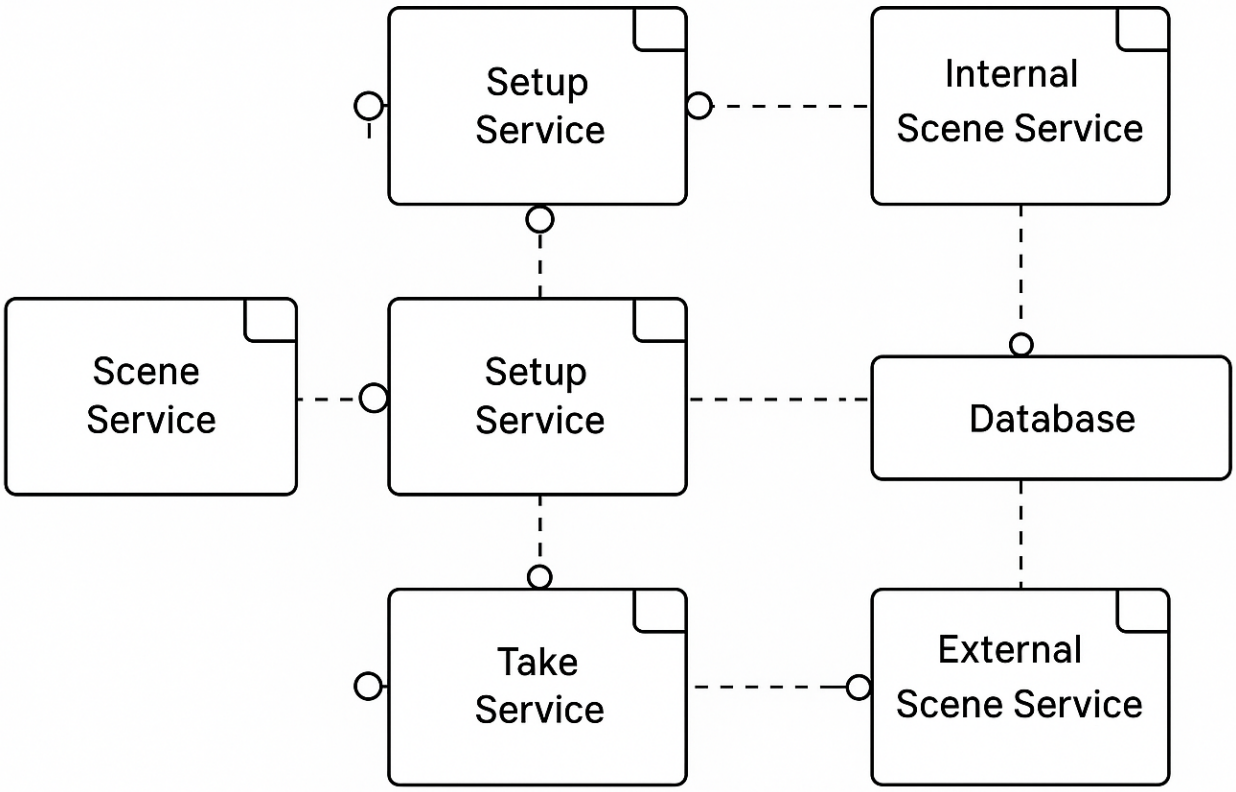
Q6.

Requirements:

```
Hollywood Approach

We are interested in building a software application to manage filmed scenes for realizing a
movie, by following the so-called “Hollywood Approach”. Every scene is identified by a code (a
string) and it is described by a text in natural language. Every scene is filmed from different
positions (at least one), each of this is called a setup. Every setup is characterized by a code
(a string) and a text in natural language where the photographic parameters are noted (e.g.,
aperture, exposure, focal length, filters, etc.). Note that a setup is related to a single scene.
For every setup, several takes may be filmed (at least one). Every take is characterized by a
(positive) natural number, a real number representing the number of meters of film that have been
used for shooting the take, and the code (a string) of the reel where the film is stored. Note
that a take is associated to a single setup. Scenes are divided into internals that are filmed in
a theater, and externals that are filmed in a location and can either be “day scene” or “night
scene”. Locations are characterized by a code (a string) and the address of the location, and a
text describing them in natural language.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Duplicate "Setup Service", otherwise not too bad. Few connections among services, each a single responsibility

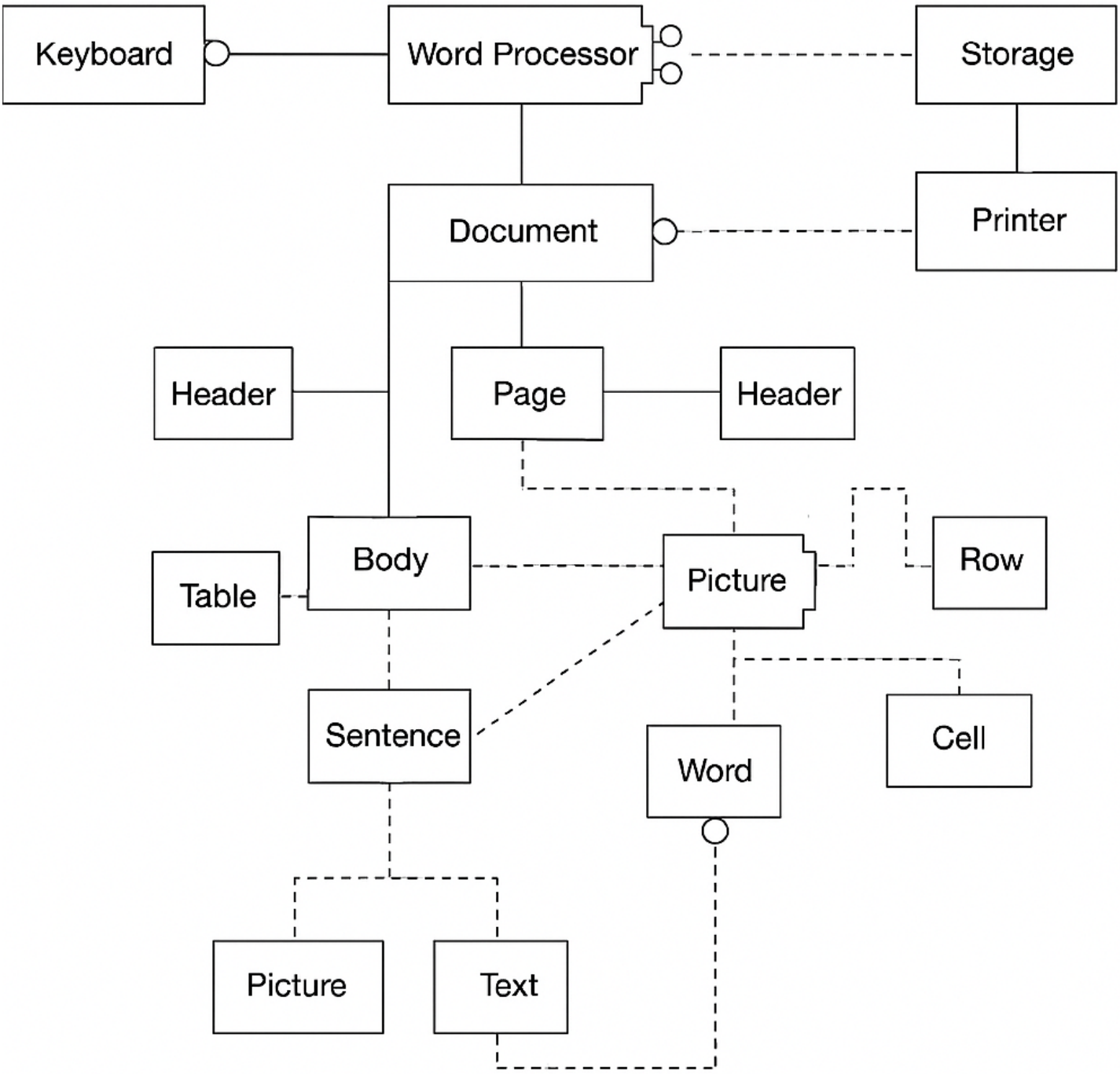
Q6.

Requirements:

```
Word Processor

A user can open a new or existing document. Text is entered through a keyboard. A document is
made up of several pages and each page is made up of a header, body and footer. Date, time and
page number may be added to header or footer. Document body is made up of sentences, which are
themselves made up of words and punctuation characters. Words are made up of letters, digits
and/or special characters. Pictures and tables may be inserted into the document body. Tables are
made up of rows and columns and every cell in a table can contain both text and pictures. Users
can save or print documents.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q8. Please provide any other comments you might have about this component diagram:

Way too many services for a high-level architecture diagram. Connections between incorrect services (e.g., sentence and picture)

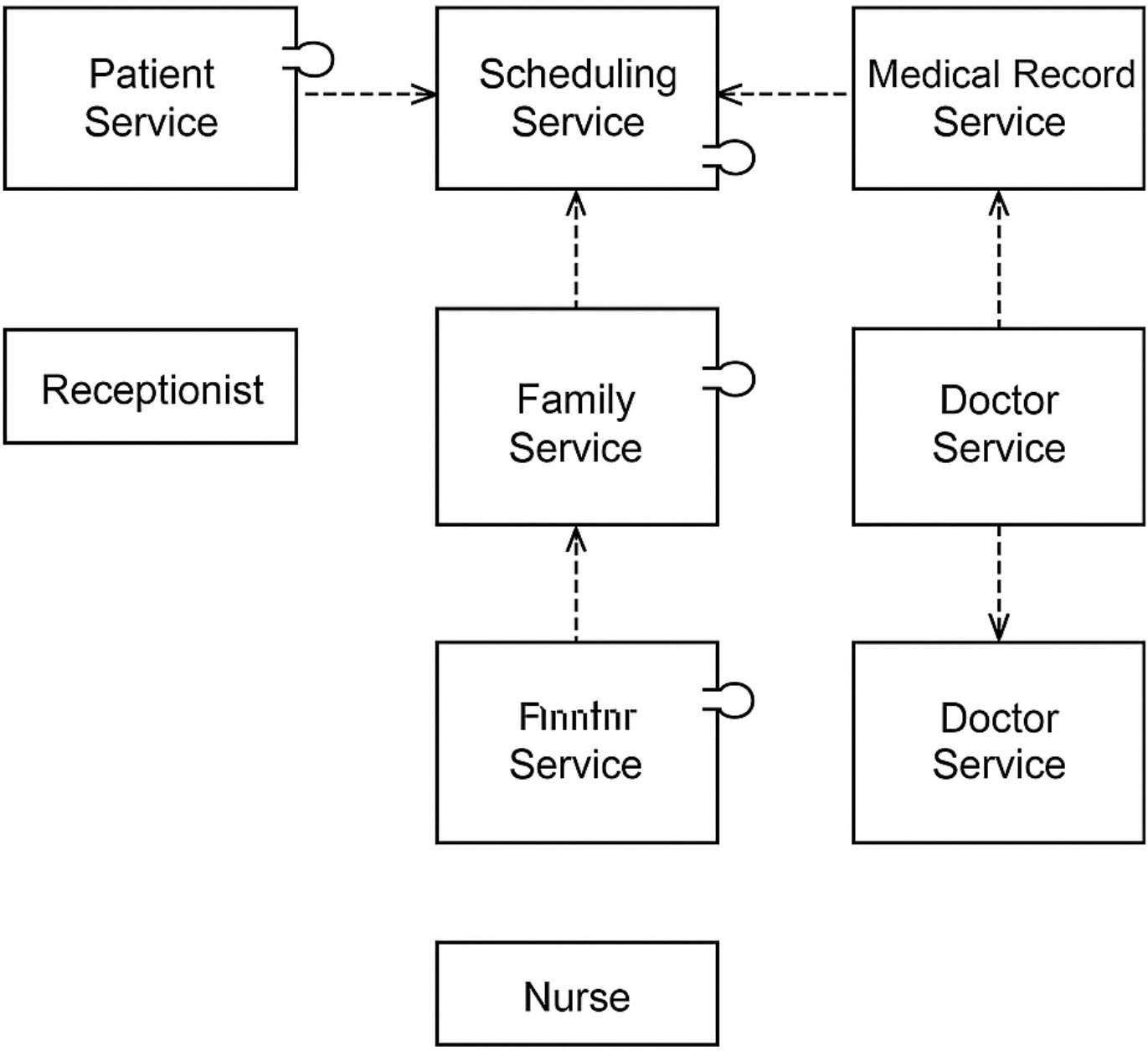
Q6.

Requirements:

```
Patient Record and Scheduling System

A patient record and scheduling system in a doctor's office is used by the receptionists, nurses,
and doctors. The receptionists use the system to enter new patient information when first time
patients visit the doctor. They also schedule all appointments. The nurses use the system to keep
track of the results of each visit including diagnosis and medications. For each visit, free form
text fields are used captures information on diagnosis and treatment. Multiple medications may be
prescribed during each visit. The nurses can also access the information to print out a history
of patient visits. The doctors primarily use the system to view patient history. The doctors may
enter some patient treatment information and prescriptions occasionally, but most frequently they
let the nurses enter this information. -- Each patient is assigned to a family. The head of
family is responsible for the person with the primary medical coverage. Information about doctors
is maintained since a family has a primary care physician, but different doctors may be the ones
seeing the patient during the visit.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q8. Please provide any other comments you might have about this component diagram:

Not sure what this does at all. Duplicate doctor service. Unconnected receptionist service. Some longer chains of association that could indicate failures.

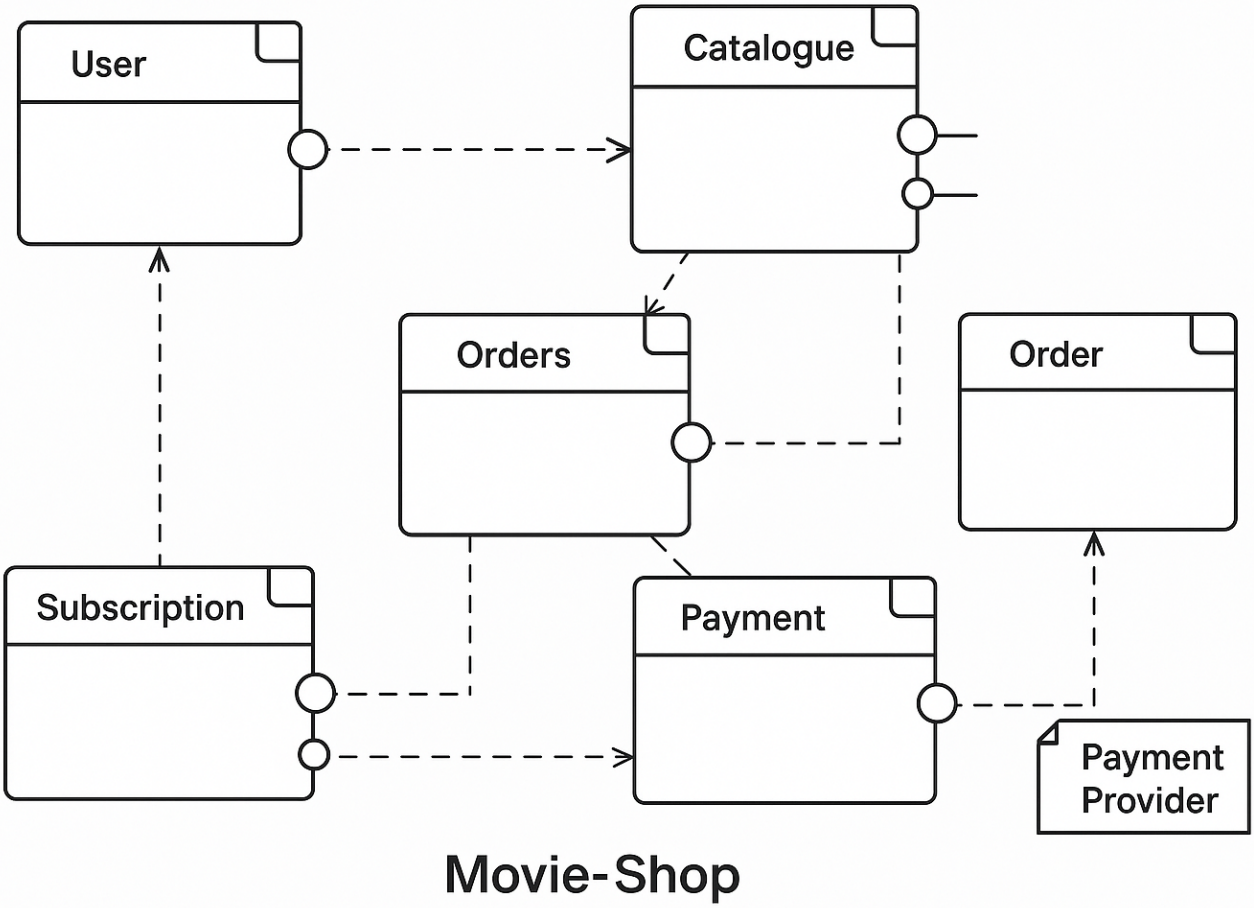
Q6.

Requirements:

```
Movie-Shop

♣ Design a system for a movie-shop, in order to handle ordering of movies and browsing of the
catalogue of the store, and user subscriptions with rechargeable cards. ♣ Only subscribers are
allowed hiring movies with their own card. ♣ Credit is updated on the card during rent
operations. ♣ Both users and subscribers can buy a movie and their data are saved in the related
order. ♣ When a movie is not available it is ordered .
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q8. Please provide any other comments you might have about this component diagram:

Not too bad. Order should be connected with Orders?

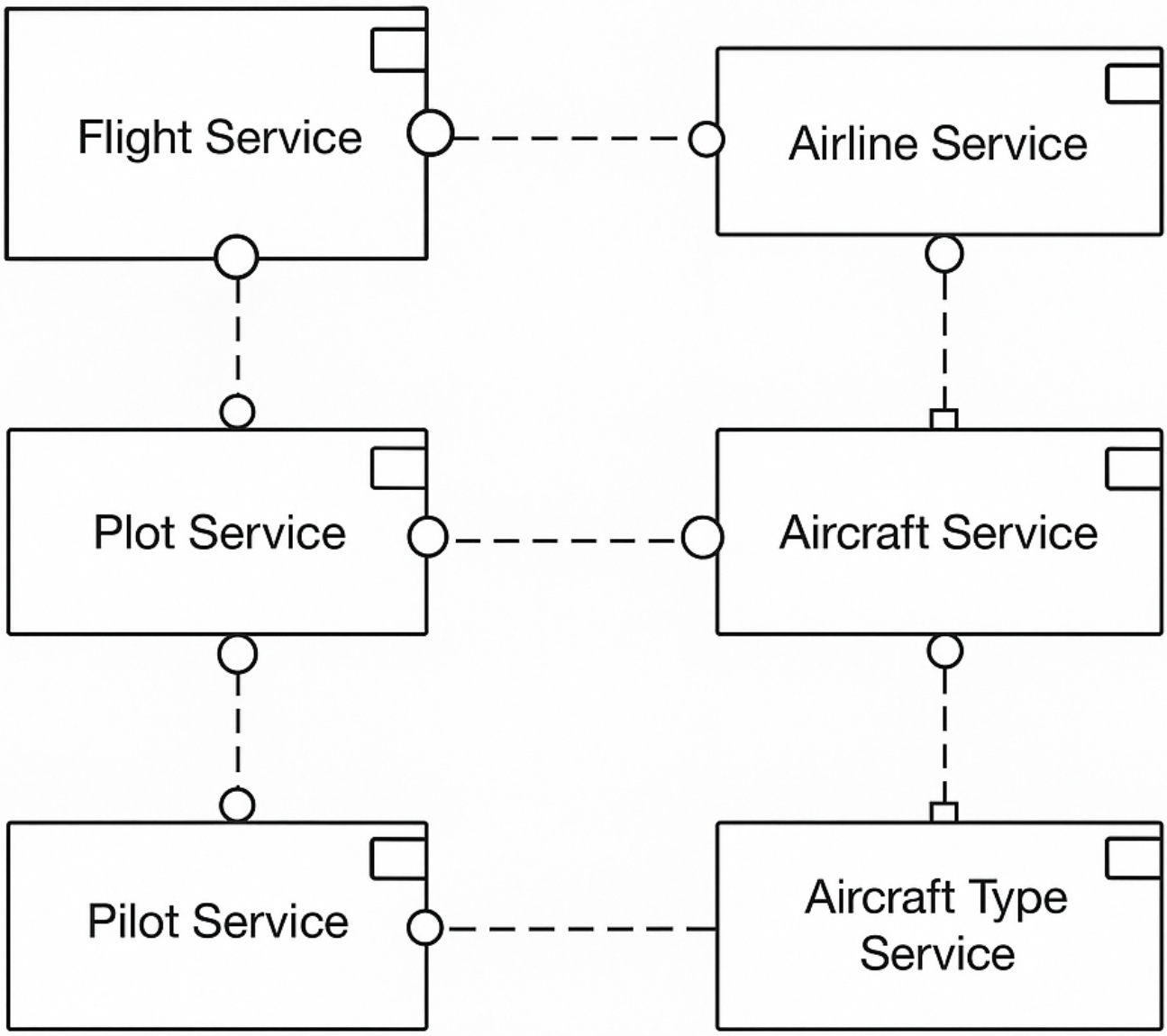
Q6.

Requirements:

```
Flights

We want to model a system for management of flights and pilots. An airline operates flights. Each
airline has an ID. Each flight has an ID a departure airport and an arrival airport: an airport
as a unique identifier. Each flight has a pilot and a co-pilot, and it uses an aircraft of a
certain type; a flight has also a departure time and an arrival time. An airline owns a set of
aircrafts of different types. An aircraft can be in a working state or it can be under repair. In
a particular moment an aircraft can be landed or airborne. A company has a set of pilots: each
pilot has an experience level: 1 is minimum, 3 is maximum. A type of aeroplane may need a
particular number of pilots, with a different role (e.g.: captain, co-pilot, navigator): there
must be at least one captain and one co-pilot, and a captain must have a level 3.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	◉	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Generally OK. Plot service should be removed.

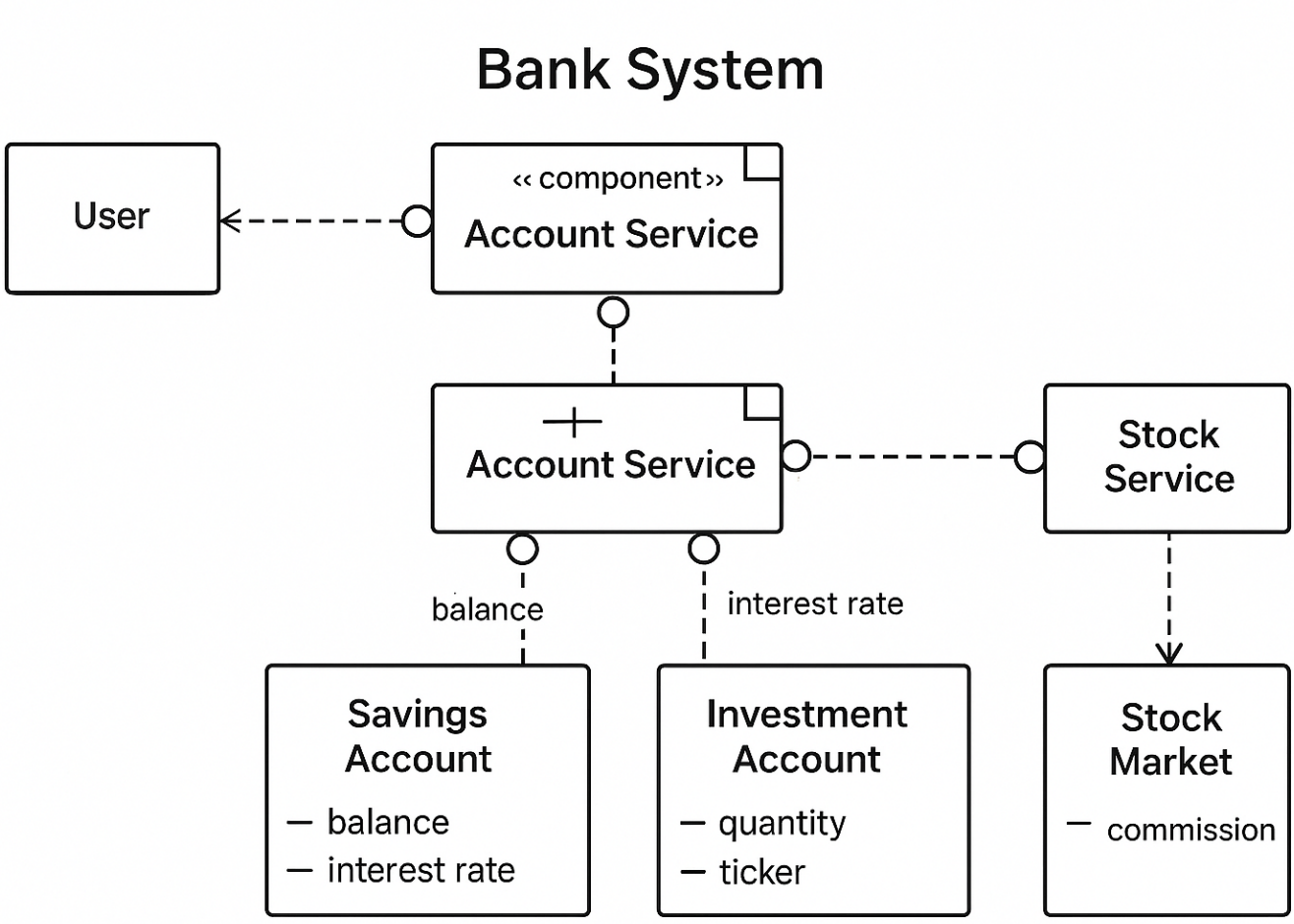
Q6.

Requirements:

```
Bank System

A bank system contains data on customers (identified by name and address) and their accounts.
Each account has a balance and there are 2 type of accounts: one for savings which offers an
interest rate, the other for investments, used to buy stocks. Stocks are bought at a certain
quantity for a certain price (ticker) and the bank applies commission on stock orders.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q8. Please provide any other comments you might have about this component diagram:

Seems like a data design again, not a microservice design. Duplicate account service

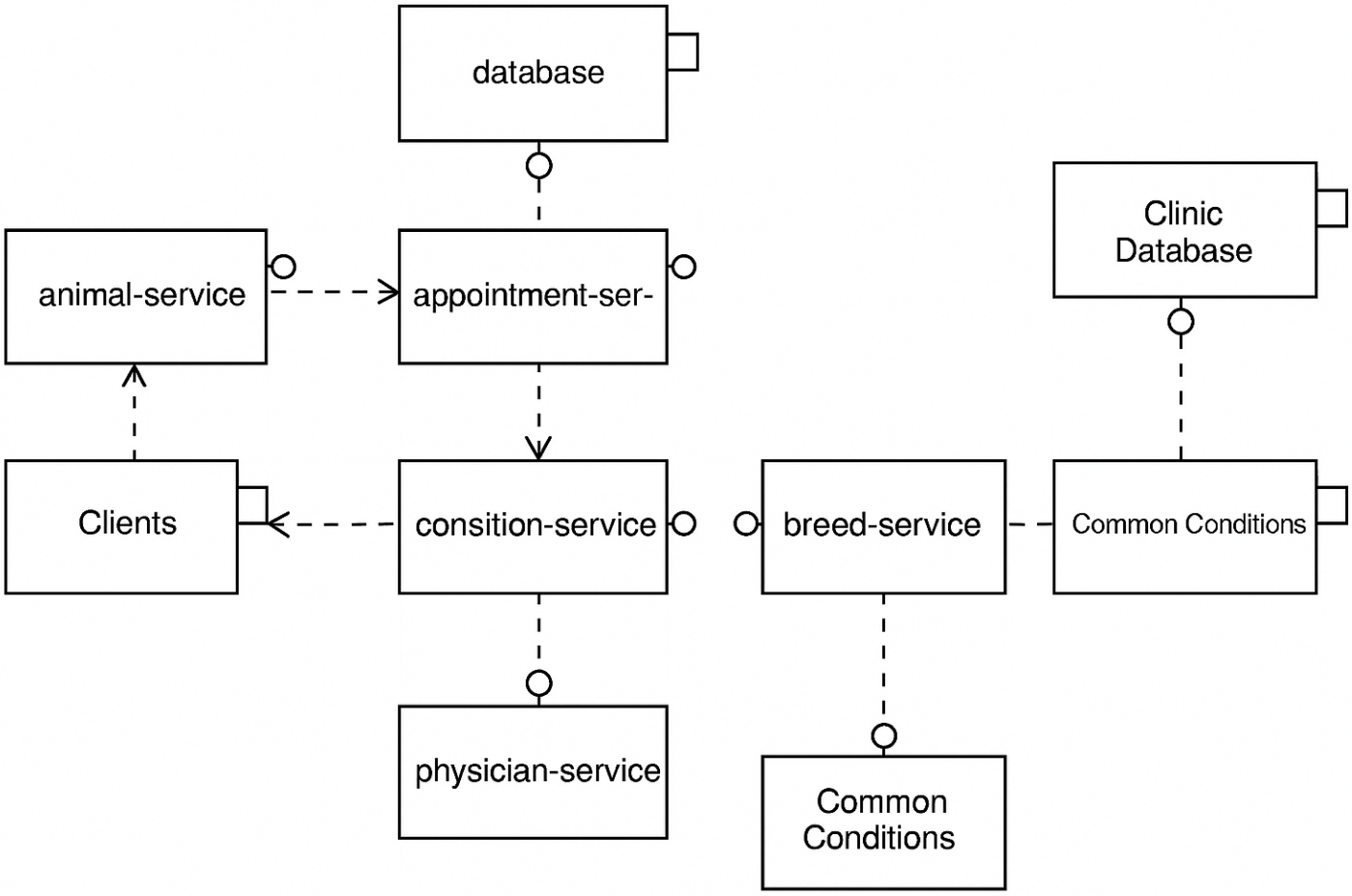
Q6.

Requirements:

```
Veterinary Clinic

The owner of a veterinary clinic wants to create a database to store information about all
veterinary services performed. After some research he came up with the following requirements:
● For each admitted animal, its name, breed (if any) and owner must be stored. Each animal should
be given an unique numeric identifier.
● For each owner, its name, address and phone number should be stored. An unique numeric
identifier should also be generated for each one of them.
● An animal might be owner-less. This happens frequently as the clinic often rescues abandoned
dogs from the streets in order to treat them and get them new owners.
● It should be possible to store information about a specific breed even if no animals of that
breed have been treated at the clinic.
● Each appointement always has a responsible physician. All appointements start at a certain date
and time; and are attended by an animal (and of course its owner).
● For each physician, his name, address and phone number should be stored. An unique numeric
identifier should also be generated for each one of them.
● In an appointement, several medical conditions might be detected. Each condition has a common
name and a scientific name. No two conditions have the same scientific name.
● It should be possible to store information about the most common conditions for each different
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q8. Please provide any other comments you might have about this component diagram:

Two disconnected graphs, that's odd. Some weird words in there ("consition"). Unclear why some of the connections are the way they are.

Q6.

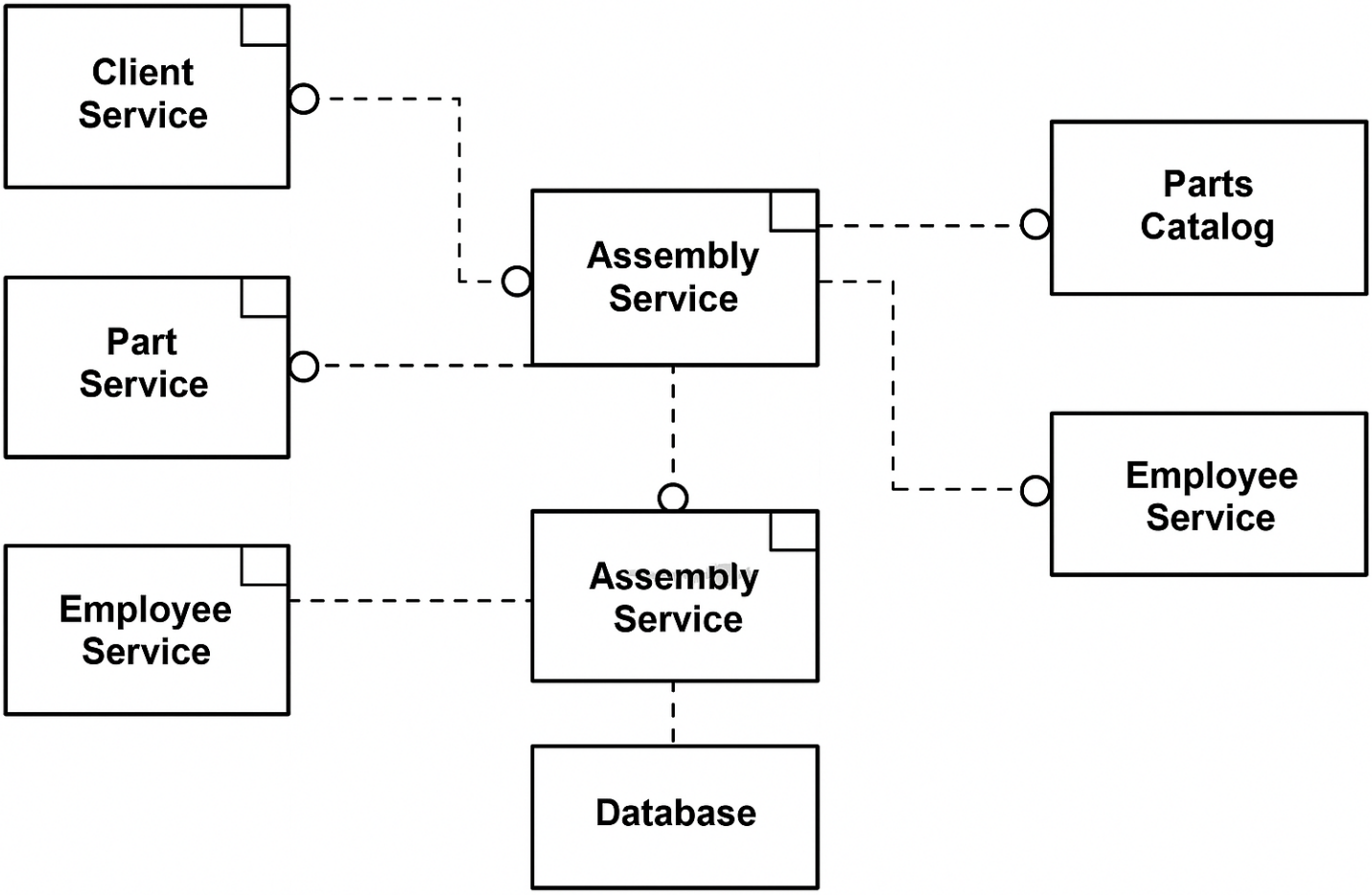
Requirements:

Auto Repair

An auto repair shop, that sells and mounts parts and accessories for all kinds of vehicles, wants a new information system to manage their clients, parts, accessories and assembly services:

- There are several employees. Each one of them has an unique identifying number, a name and an address.
- In this shop, assembly services, where parts and accessories are installed in a vehicle, are executed. For each one these services the following data must be stored: In which car the service was executed, how many kms had the car at the time, who was the responsible employee, which parts and accessories were fitted, how many work hours did it take and the admission and finish dates.
- Parts and accessories are only sold together with an assembly service.
- Each part/accessory only fits in some car models. Therefore, it is important to store that information.
- Each part/accessory has a category (radio, tyre, …), a serial number and a price.
- Each car has a license plate, a make, a model, a color and an owner. Each owner has a name, identifying number, address and a phone.
- One person can own more than one car but one car only has one owner.

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	◉	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	○	◉
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	○	◉
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Duplicate assembly service needs to be removed, but otherwise ok. Assembly service is a single point of failure, but the "manager" services frequently are.

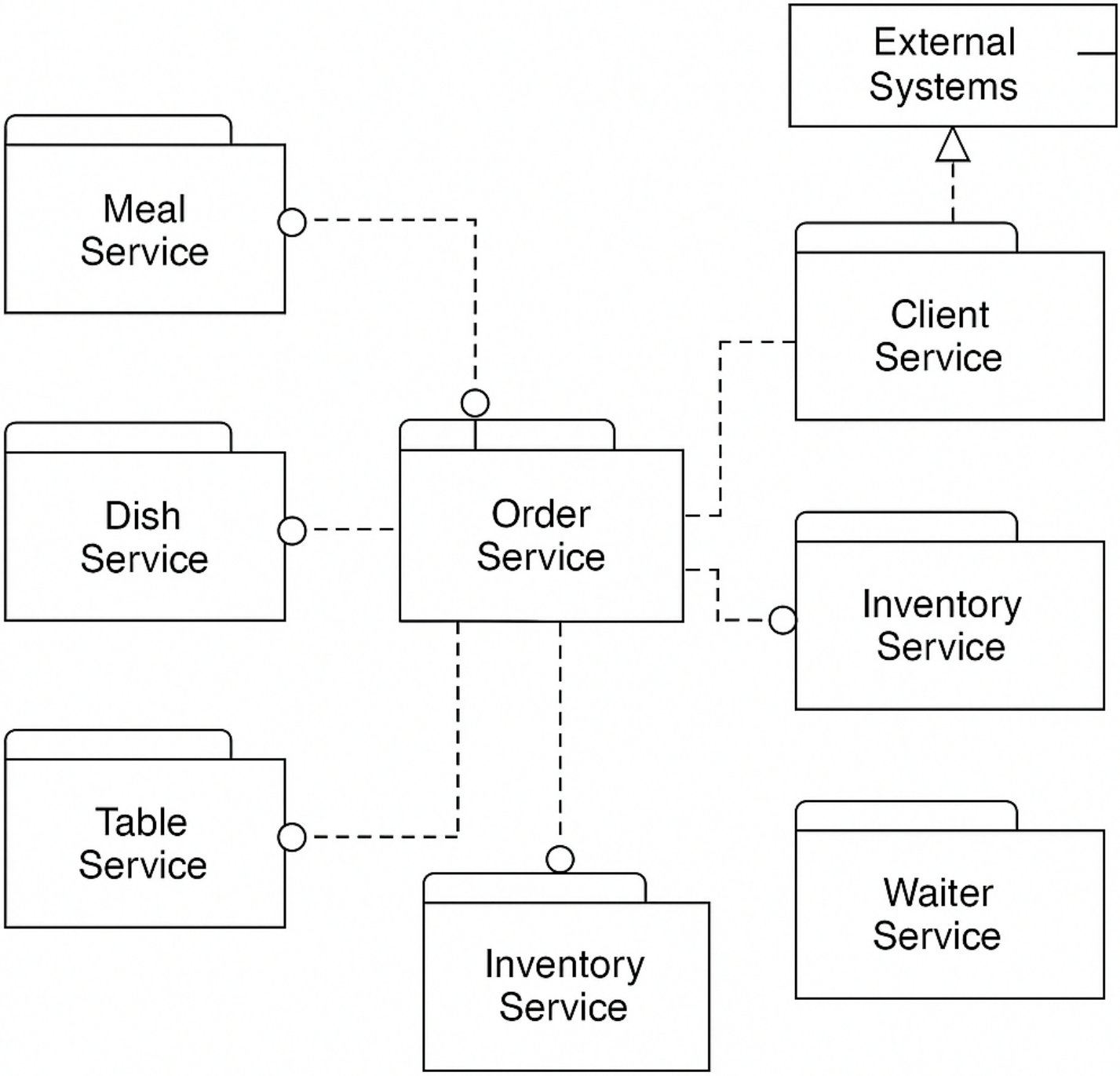
Q6.

Requirements:

```
10. Restaurant

The owner of a small restaurant wants a new information system to store data for all meals
consumed there and also to keep a record of ingredients kept in stock. After some research he
reached the following requirements list:
• Each ingredient has a name, a measuring unit (e.g. olive oil is measured in liters, while eggs
are unit based) and a quantity in stock. There are no two ingredients with the same name.
• Each dish is composed of several ingredients in a certain quantity. An ingredient can, of
course, be used in different dishes.
• A dish has an unique name and a numeric identifier.
• There are several tables at the restaurant. Each one of them has an unique numeric identifier
and a maximum ammount of people that can be seated there.
• In each meal, several dishes are consumed at a certain table. The same dish can be eaten more
than once in the same meal.
• A meal takes place in a certain date and has a start and end time. Each meal has a responsible
waiter.
• A waiter has an unique numerical identifier, a name, an address and a phone number.
• In some cases it is important to store information about the client that consumed the meal. A
client has a tax identification number, a name and an address.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q8. Please provide any other comments you might have about this component diagram:

Generally ok except for correctness. Should be more connections between services like order and waiter.

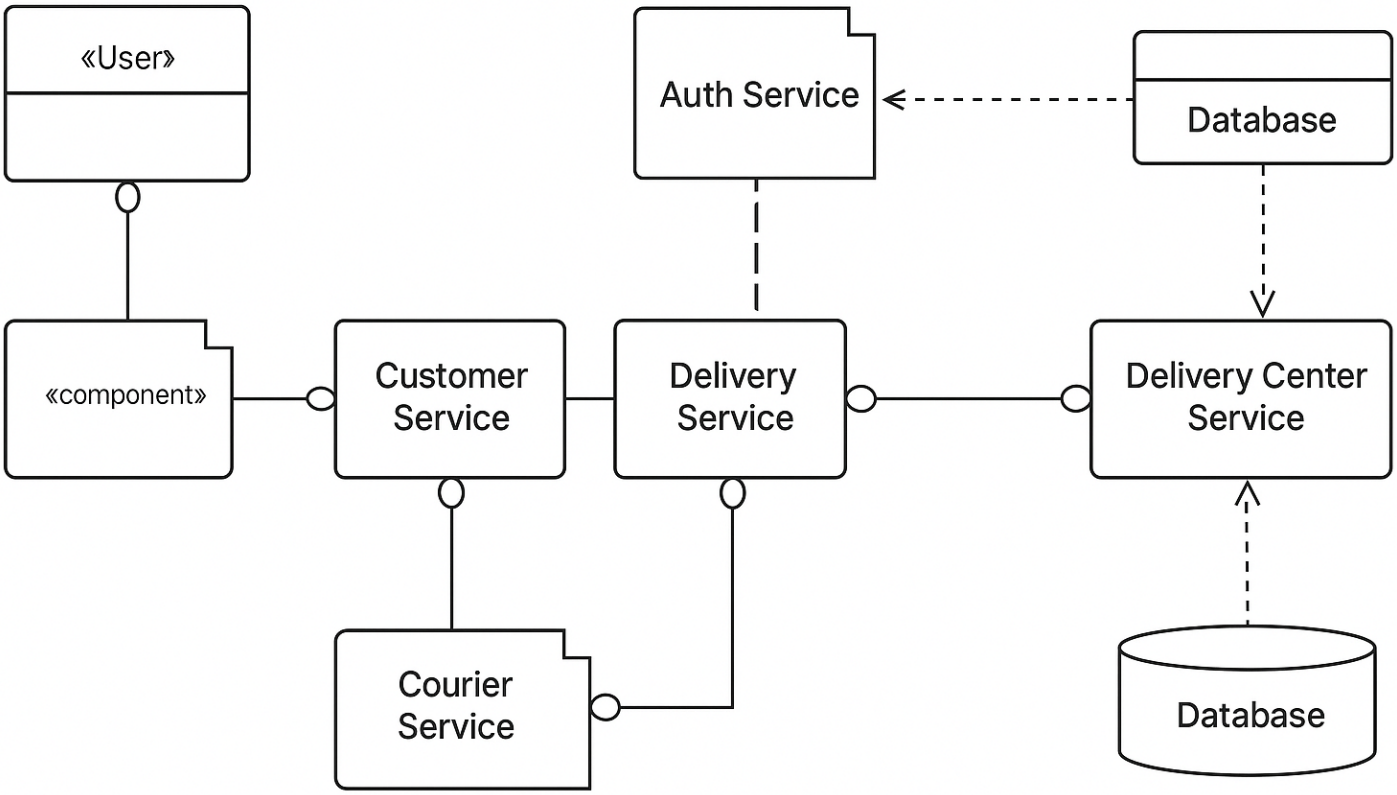
Q6.

Requirements:

```
Deliveries

The owner of a small delivery company plans to have an information system that allows him to save
data about his customers and deliveries. After some time studying the problem, he reached the
following requirements:
• Each customer has a VAT number, a name, a phone number and an address. There are no two clients
with the same VAT number.
• When a customer wants to send a package to another customer, he just has to login to the
company website, select the customer he wants to send the package to, enter the package's weight
and if the delivery is normal or urgent. He then receives an unique identifier code that he
writes on the package.
• The package is then delivered by the customer at the delivery center of his choosing. A
delivery center has a unique name and an address.
• Each client has an associated delivery center. This delivery center is chosen by the company
and it is normally the one closest to the customer's house.
• The package is them routed through an internal system until it reaches the delivery center of
the recipient.
• The package is then delivered by hand from that delivery center to the recipient by a courier.
• Couriers have a single VAT number, a name and a phone number. Each courier works in a single
```

Component Diagram:



Microservices actigumnet

Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q8. Please provide any other comments you might have about this component diagram:

Duplicate database. Generally not bad.

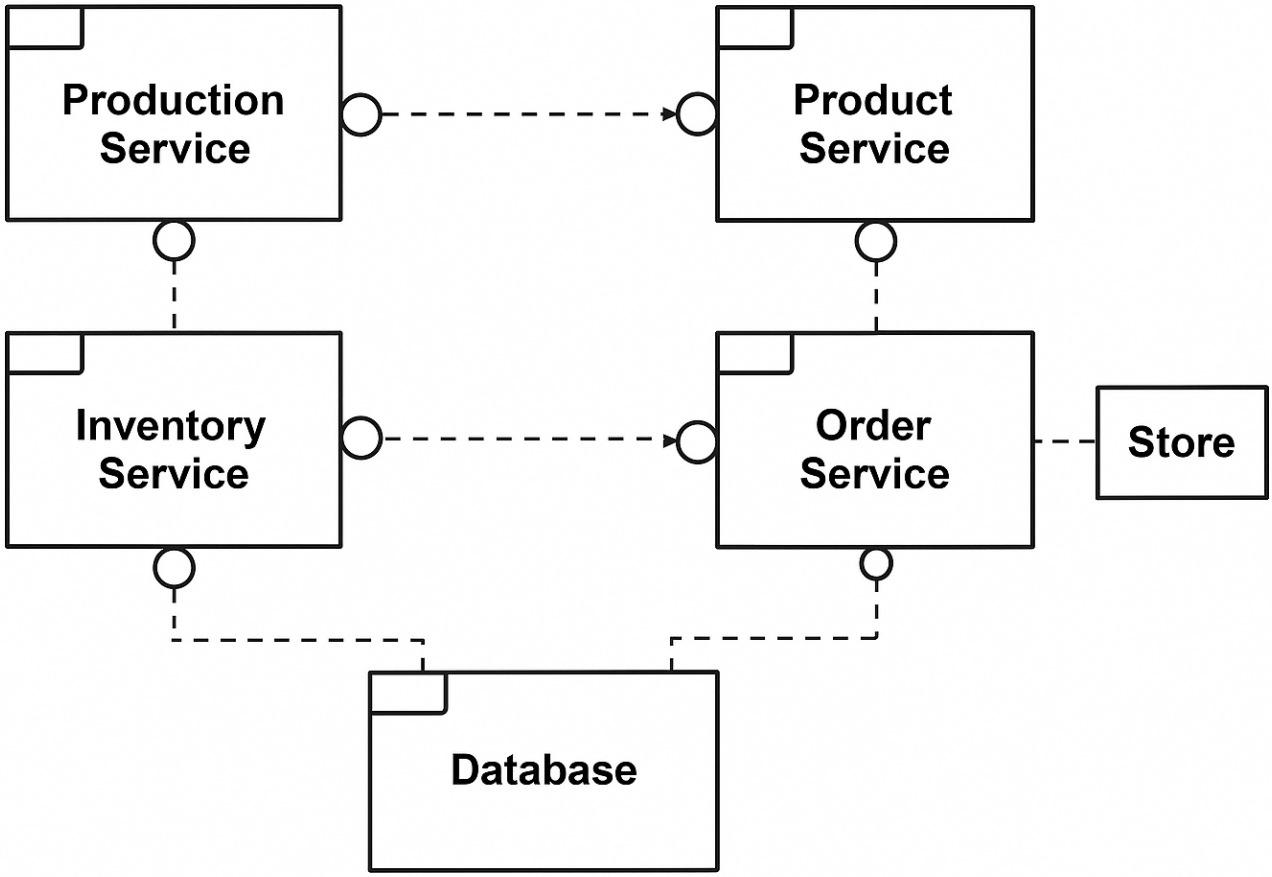
Q6.

Requirements:

```
Furniture

The known furniture factory Hi-Key-Ah, intends to implement an information system to store all
data on the different types of furniture and components it produces:
● The factory produces several lines of furniture, each with a different name and consisting of
several pieces of furniture of different types (beds, tables, chairs, …).
● All furniture pieces have a type, a single reference (eg CC6578) and a selling price.
● The major competitive advantage of this innovative plant is the fact that each component
produced can be used in more than one piece of furniture.
● Each piece of furniture is thus composed of several components. The same component can be used
more than once in the same piece.
● Every type of component produced is assigned a unique numerical code, a manufacturing price and
a type (screw, hinge, shelf …).
● The furniture is then sold in various stores throughout the world. Each store has a different
address and a fax number.
● To make the manufacturing process more efficient, stores have to place orders everytime they
need to replenish their stock. These orders must also be stored in the database.
● Each order has a order number, a date, the store that placed the order as well as a list of all
the ordered furniture and their quantities.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	◉	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	◉	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Generally ok.

Q6.

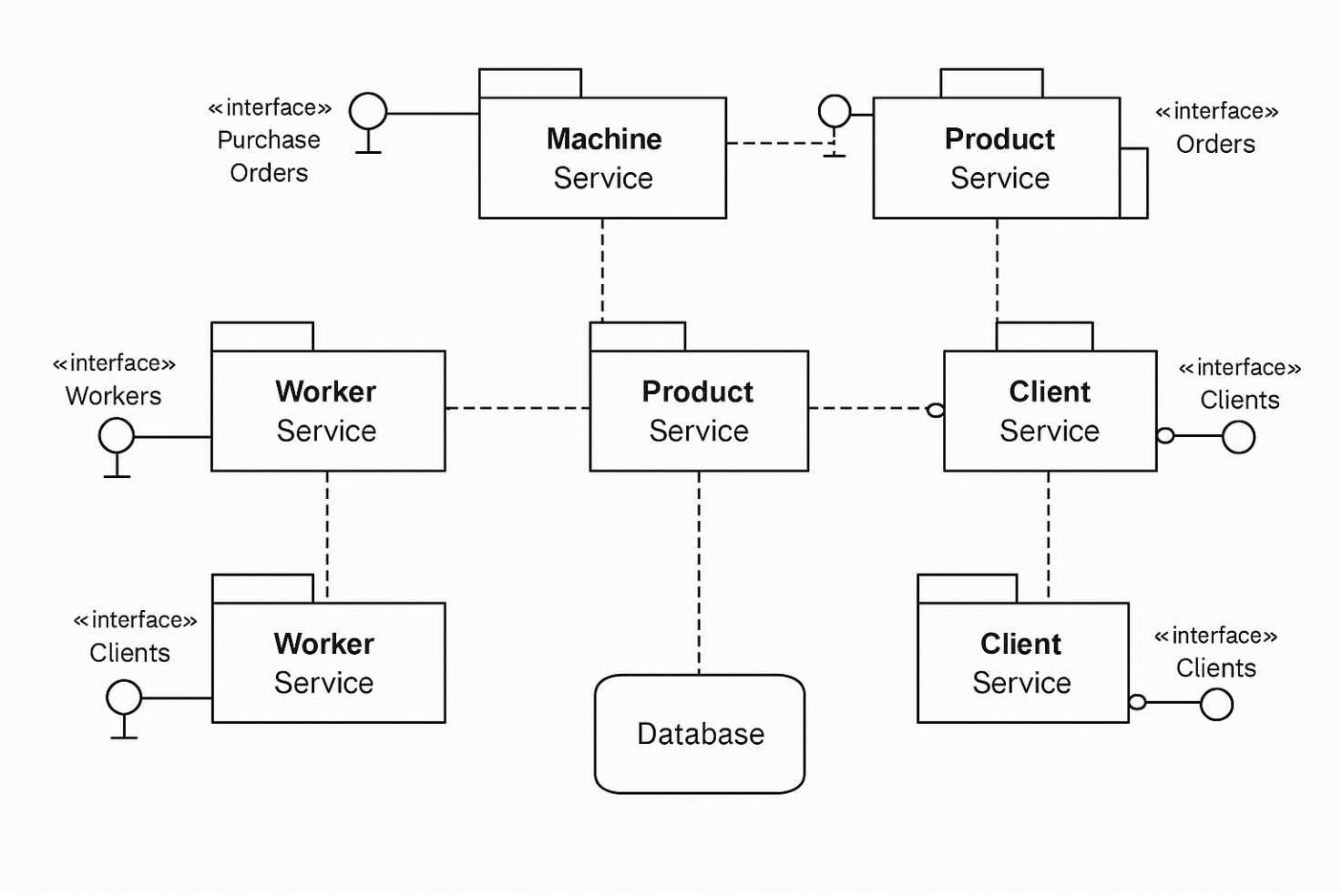
Requirements:

Factory

Create a database for a factory with the following requirements. Don’t forget to add unique identifiers for each one of the entities if needed.

- A factory has several machines. Each one of them is operated by several workers.
- A worker might work in more than one machine.
- In this factory, several products of different types, are produced. Each different type of product is produced in a single machine. But, the same machine can produce more than one type of product.
- Products from the same type are all produced from the same single material and have the same weigth.
- Clients can issue purchase orders. Each order has a list of the desired products and their quantity.
- For each worker, the following data should be stored in the database: name (first and last), birth date, address and a list of his skills.
- For each machine, the following data should be stored: serial number, make, model and purchase date.
- For each client, the followig data should be stored: name, address, phone number and name of the contact person (if any).

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	◉	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Pretty good microservices. Correctness bad, duplicates of mulitple components, missing connections between them.

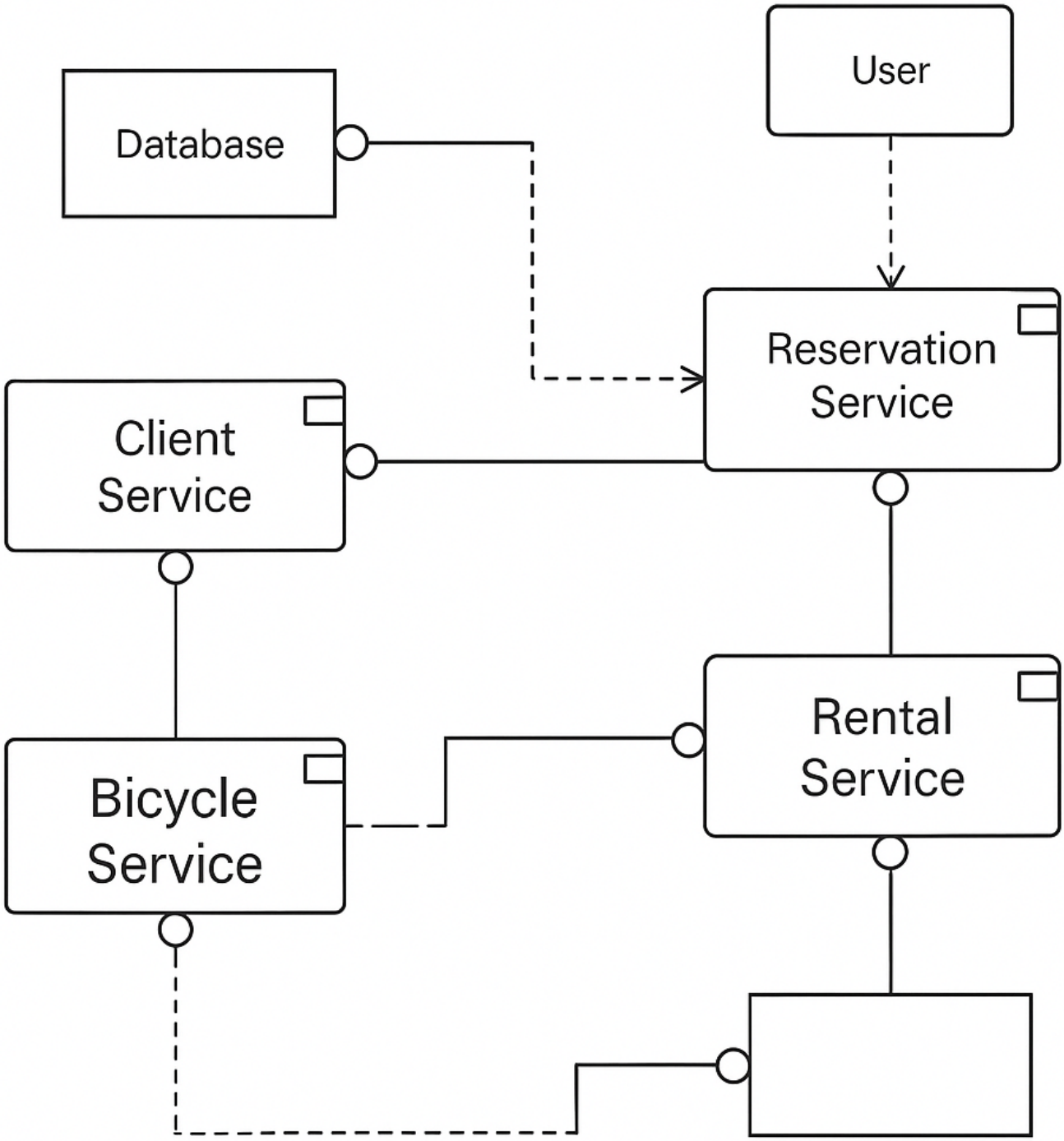
Q6.

Requirements:

```
Bycicle Rental

A bicycle renting company wants to create an information system that allows it to store the data
regarding all their reservations and rentals. The system should follow these requirements:
● It should be possible to store the national id number (NIN), tax identification number (TIN),
name and address for every client. The NIN and TIN must be different for every client and all
clients should have at least a TIN and a name.
● The database should also contain information about the bicycle models that can be rented- Each
model has an unique name, a type (that can only be road, mountain, bmx or hybrid) and the number
of gears.
● Each bicycle has a unique identifying number and a model.
● The company has several different stores where bicycles can be picked up and returned. Each one
of these stores is identified by an unique name and has an address (both mandatory).
● When a reservation is made, the following data must be known: which client did the reservation,
when will he pick up the bike (day), which bike model he wants and where will he pick up the bike
(store).
● When a bike is picked up, the actual bike that was picked up must be stored in the database.
● When a bike is returned, the return date should be stored in the database.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	◉	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Some empty components in the diagram. Should be more connections to the database

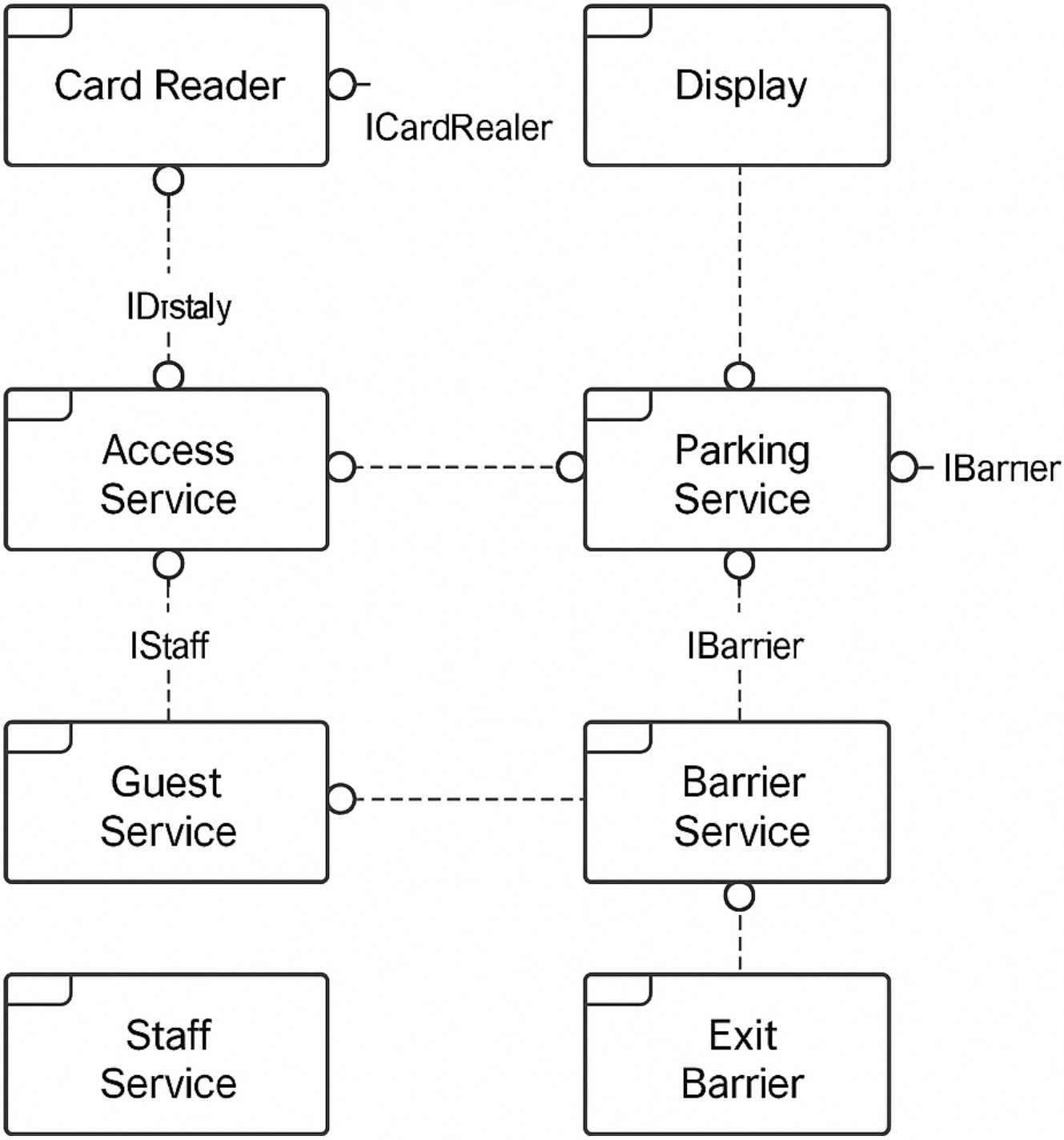
Q6.

Requirements:

```
Saturn Int. Management

Saturn Int. management wants to improve their security measures, both for their building and on
site. They would like to prevent people who are not part of the company to use their car park.
Saturn Int. has decided to issue identity cards to all employees. Each card records the name,
department and number of a company staff, and give them access to the company car park. Employees
are asked to wear the cards while on the site.
There is a barrier and a card reader placed at the entrance to the car park. When a driver drives
his car into the car park, he/she inserts his or her identity card into the card reader. The card
reader then verify the card number to see if it is known to the system. If the number is
recognized, the reader sends a signal to trigger the barrier to rise. The driver can then drive
his/her car into the car park.
There is another barrier at the exit of the car park, which is automatically raised when a car
wishes to leave the car park.
A sign at the entrance display “Full” when there are no spaces in the car park. It is only
switched off when a car leaves.
There is another type of card for guests, which also permits access to the car park. The card
records a number and the current date. Such cards may be sent out in advance, or collected from
reception. All guest cards must be returned to reception when the visitor leaves Saturn Int.
```

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q8. Please provide any other comments you might have about this component diagram:

Some services there, but not great. Barrier service functionality spread among multiple services.

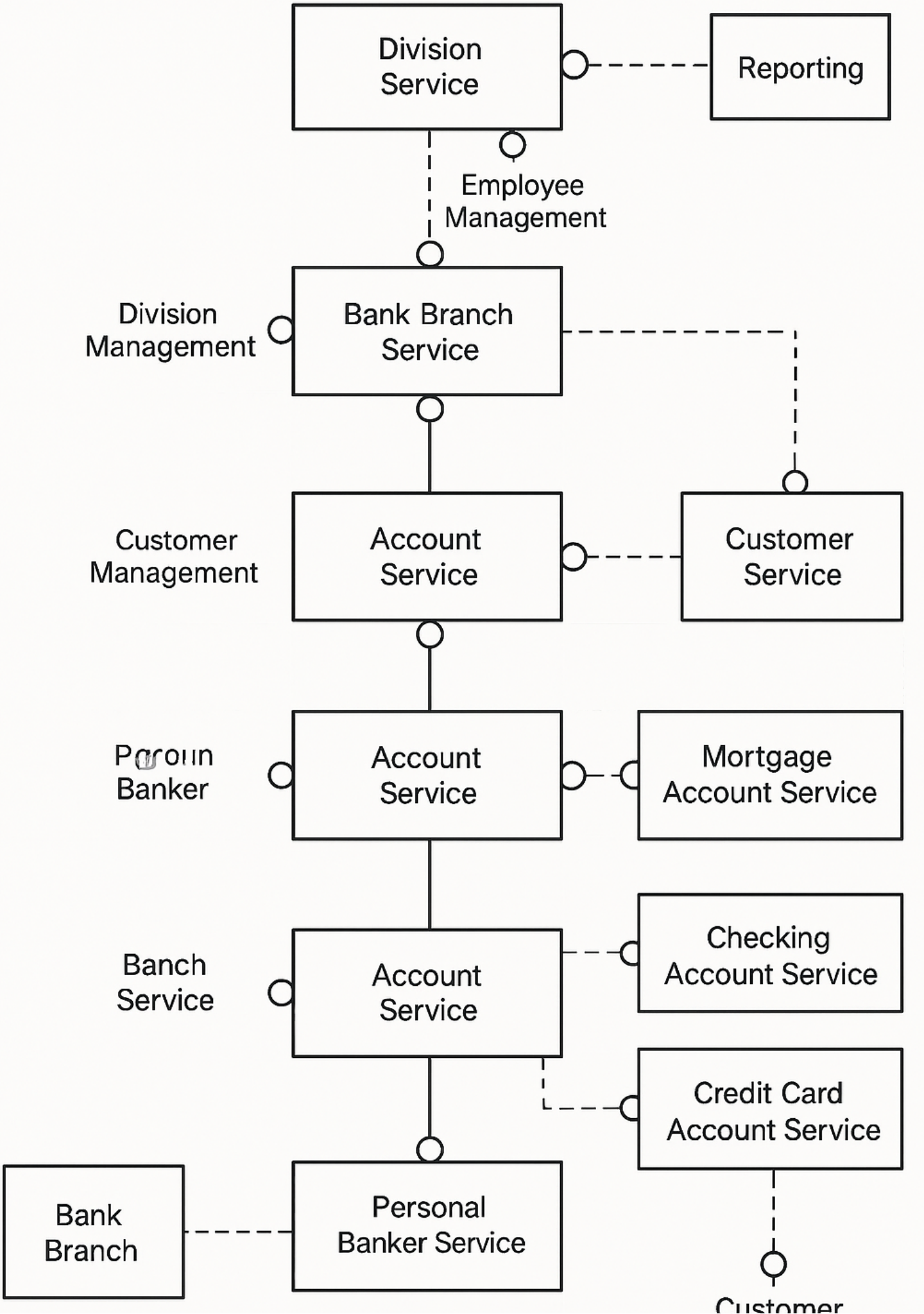
Q6.

Requirements:

```
OOBank

This system provides the basic services to manage bank accounts at a bank called OOBank. OOBank
has many branches, each of which has an address and branch number. A client opens accounts at a
branch. Each account is uniquely identified by an account number; it has a balance and a credit
or overdraft limit. There are many types of accounts, including: a mortgage account (which has a
property as collateral), a checking account, and a credit card account (which has an expiry date
and can have secondary cards attached to it). It is possible to have a joint account (e.g. for a
husband and wife). Each type of account has a particular interest rate, a monthly fee and a
specific set of privileges (e.g. ability to write checks, insurance for purchases etc.). OOBank
is divided into divisions and subdivisions (such as Planning, Investments and Consumer); the
branches are considered subdivisions of the Consumer Division. Each division has a manager and a
set of other employees. Each customer is assigned a particular employee as his or her 'personal
banker'.
```

Component Diagram:


Division Service
Reporting
Employee Management
Division Management
Bank Branch Service
Customer Management
Account Service
Customer Service
Account Service
Mortgage Account Service
Banch Service
Account Service
Checking Account Service
Credit Card Account Service
Bank Branch
Personal Banker Service

Customer

Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q8. Please provide any other comments you might have about this component diagram:

Duplicate services. Branch and personal banker should connect

Q6.

Requirements:

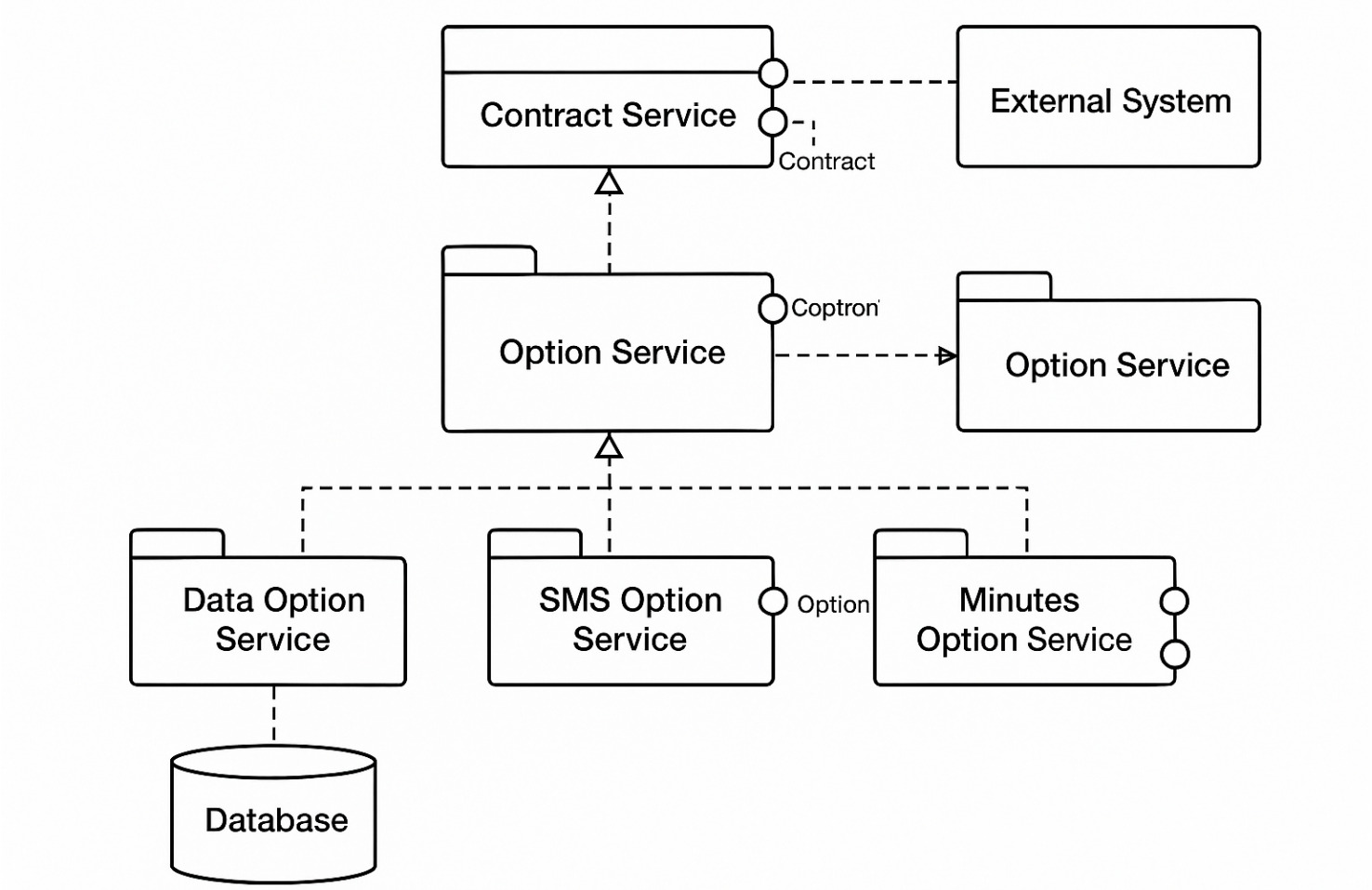
Prepaid Cell Phone

The contract of a prepaid cell phone should be modelled and implemented. A basic contract has a contract number (of type int) and a balance (of type double), but no monthly charges. The contract number is not automatically generated, but is to be set as a parameter by the constructor as well as the initial balance. The balance has a getter and a setter. The following options can be added to a contract (if needed also several times):

- 100 MB of data (monthly charge 1.00€)
- 50 SMS (monthly charge 0.50€)
- 50 minutes (monthly charge 1.50€)
- Double Transfer Rate (monthly charge 2.00€) implement this requirement with the help of the decorator pattern. All contract elements should be able to understand the methods getCharges():double, getBalance():double and setBalance(double).

The method getCharges() should provide the monthly charge of a contract with all its options selected. The methods getBalance() and setBalance() should be passed through and access the basic contract.

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q8. Please provide any other comments you might have about this component diagram:

Shows inheritance, not microservices. Duplicate option service

Q6.

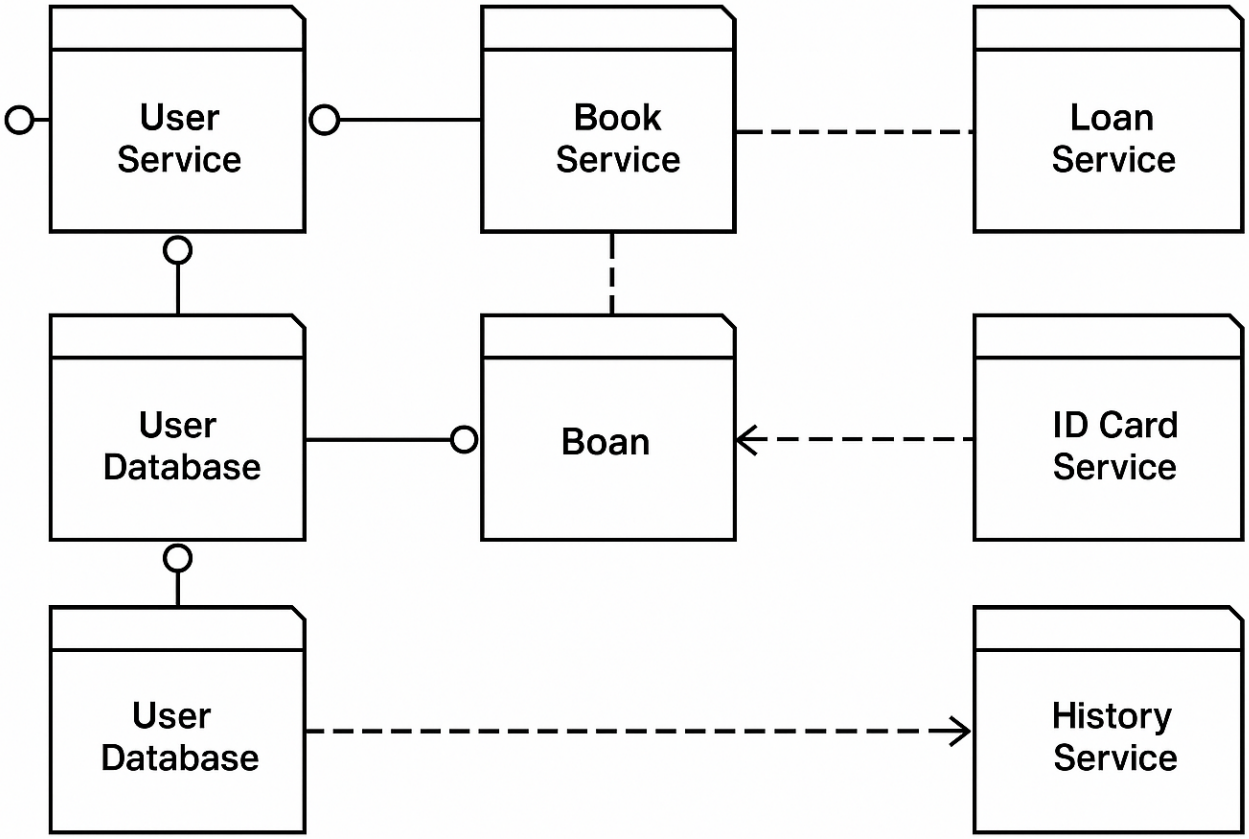
Requirements:

```
Library System

The exercise id to design a class structure for a library system. It should fulfil those
requirements:
```

- There are two type of users - under-aged and adults.
- Under-aged users are identified with usage of their full name and student card.
- Adult users are identified with usage of their full name and ID card.
- The library contains books.
- There is basic information about every book (title, author, etc).
- The user can borrow at most 4 books at the same time.
- There is a history of previously borrowed books for every user (along with all the dates).

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○

Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q8. Please provide any other comments you might have about this component diagram:

Book database should be inside book service. "Boan"? Oddly connected history service

Q6.

Requirements:

MyDoctor

The MyDoctor application aims to be a management tool for the appointments of a doctor.
A hospital has multiple offices.
The users of the application can be doctors and patients.
The doctors can apply to practice in offices and create a schedule for an office. The schedules in different offices can't overlay.

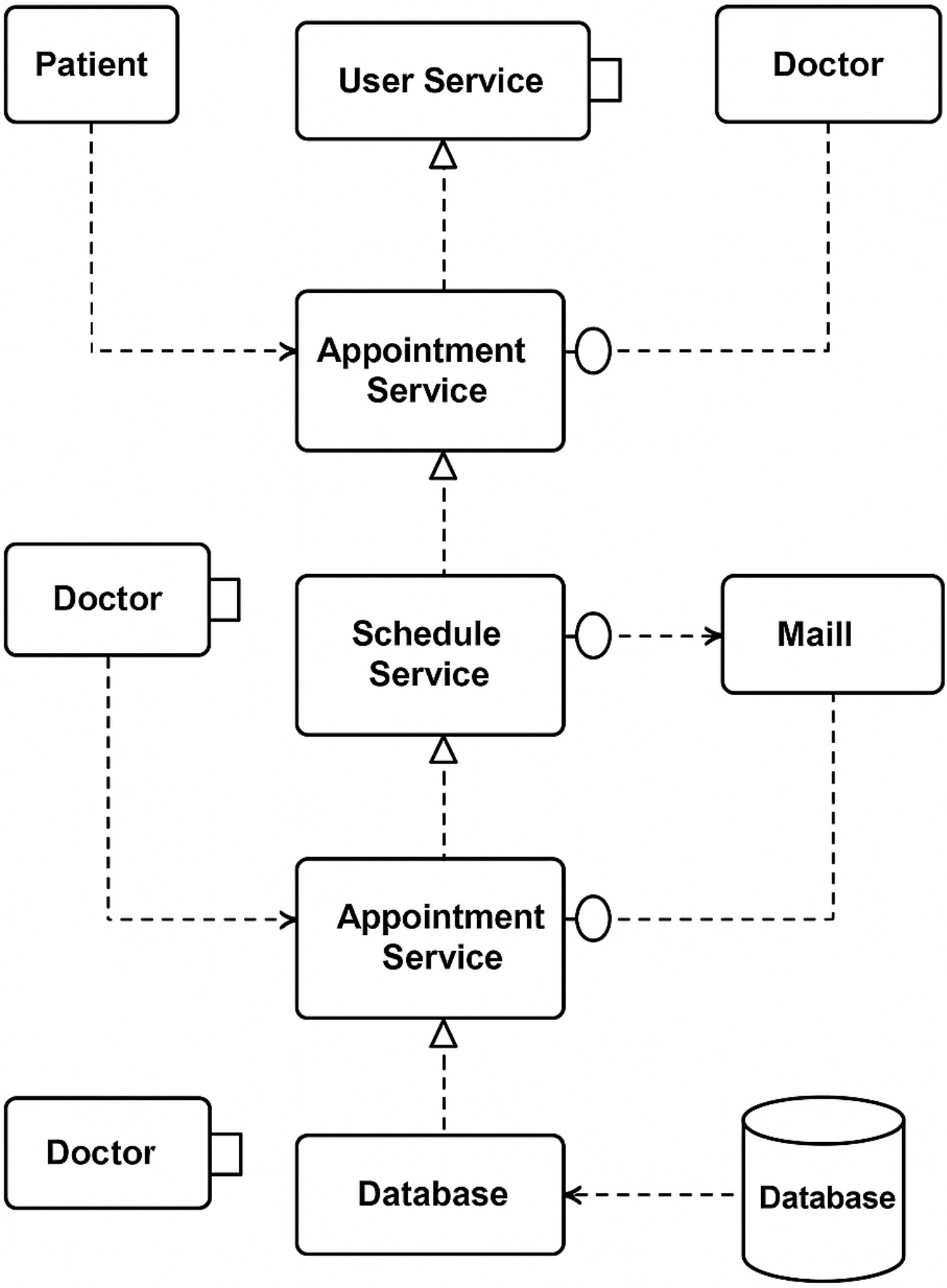
📝 Example:
Doctor Ana is available in Office 4 on the 4th of September during 1 PM - 5PM.
Doctor Ana can't practice in Office 5 on the 4th of September during 3PM - 8 PM, but she can practice in Office 5 on the 4th of September during 5:30PM - 8 PM.
The patients can see the existing doctors in the system, the schedule of the offices and can book appointments for specific doctors and for specific schedules. The appointments can be of 3 types:

- Blood Test - 15 mins
- Consultation - 30 mins
- Surgery - 60 mins

The booking of an appointment will not be possible if another appointment is already booked at the same time frame. An email is sent to the patient with the confirmation of the appointment.

📝 Example:

Component Diagram:


Patient
User Service
Doctor
Appointment Service
Doctor
Schedule Service
Maill
Appointment Service
Doctor
Database
Database

Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q8. Please provide any other comments you might have about this component diagram:

Overall clearly interacting microservices. Multiple doctors, data should be managed in service.

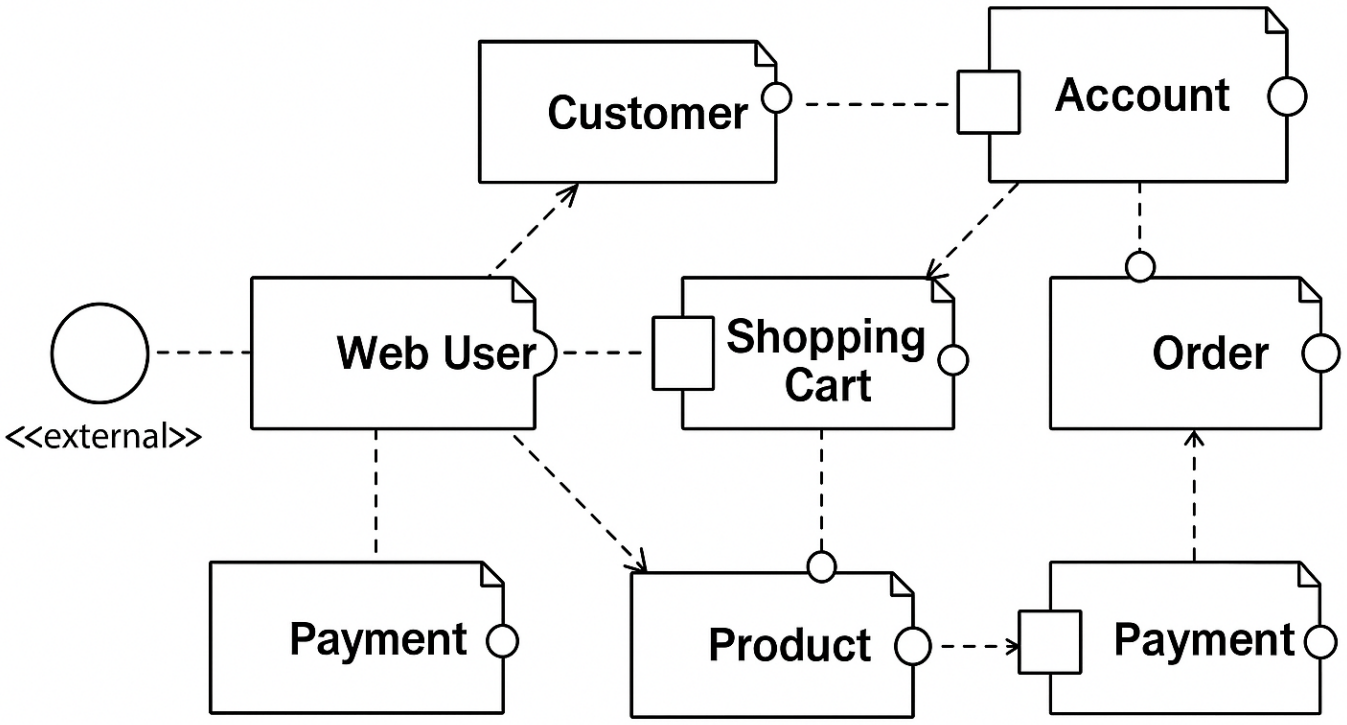
Q6.

Requirements:

Online Shopping

Each customer has unique id and is linked to exactly one account. Account owns shopping cart and orders. Customer could register as a web user to be able to buy items online. Customer is not required to be a web user because purchases could also be made by phone or by ordering from catalogues. Web user has login name which also serves as unique id. Web user could be in several states - new, active, temporary blocked, or banned, and be linked to a shopping cart. Shopping cart belongs to account.
Account owns customer orders. Customer may have no orders. Customer orders are sorted and unique. Each order could refer to several payments, possibly none. Every payment has unique id and is related to exactly one account.
Each order has current order status. Both order and shopping cart have line items linked to a specific product. Each line item is related to exactly one product. A product could be associated to many line items or no item at all.

Component Diagram:



Requested Architectural Style:

Microservice architecture: A microservice architecture is characterized by a series of small, single responsibility components designed to deploy independently as needed to maximize resilience and scalability. Microservices are ideally stateless, but even in cases where state is needed, each service is expected to manage its own state. Loose coupling is required such that the failure of one service will not cascade to produce a failure in another service.

- Each component should expose only one / very few interfaces (to indicate their single responsibility).
- There should not be long "chains" of interfaces, which may indicate a failure in one component could cascade to others.
- There should be no shared state between components.

Q7. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q8. Please provide any other comments you might have about this component diagram:

Not too bad. Multiple payment. Order should connect to product.

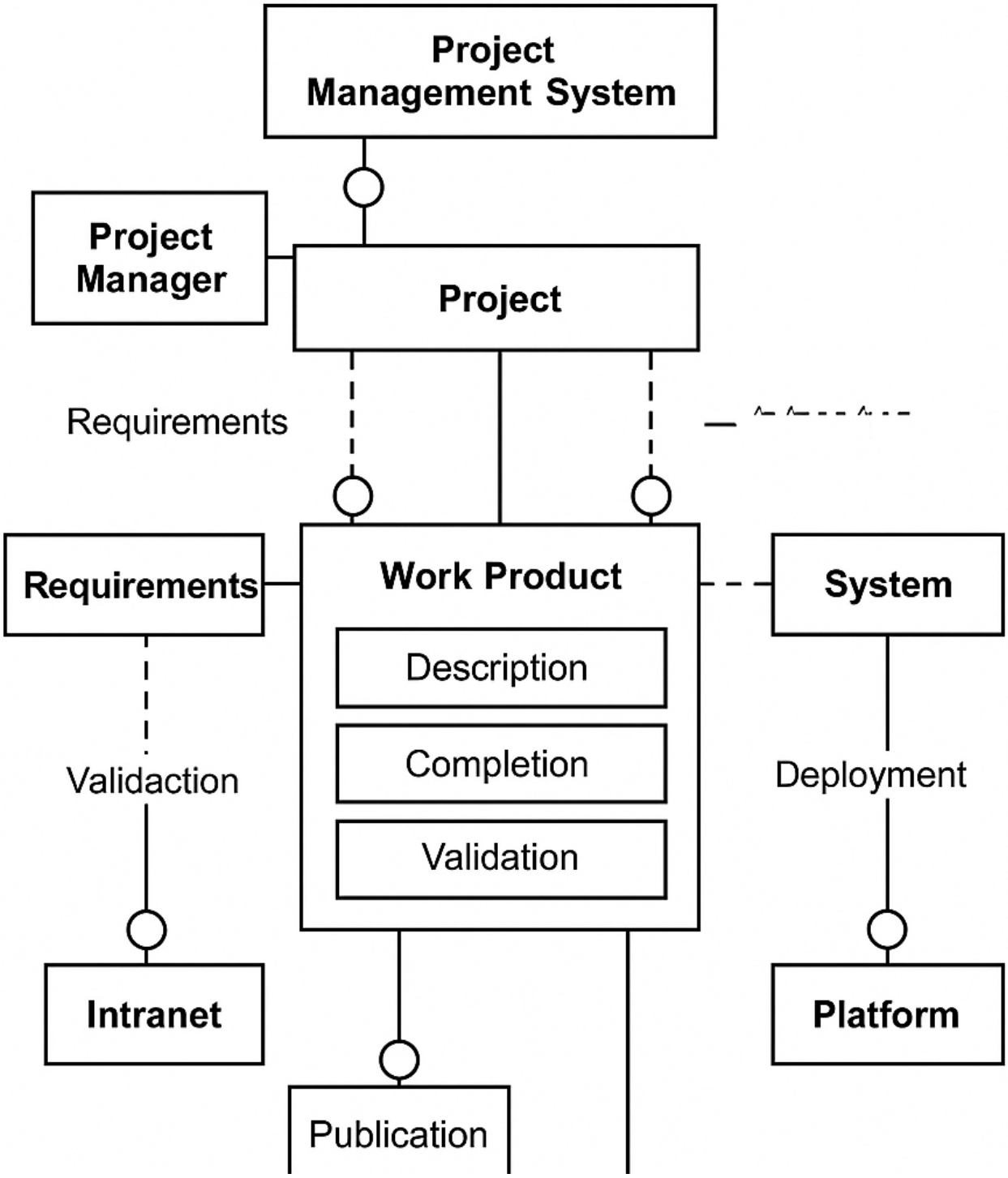
Q9.

Requirements:

```
Project Management System

A project manager uses the project management system to manage a project. The project manager
leads a team to execute the project within the project's start and end dates. Once a project is
created in the project management system, a manager may initiate and later terminate the project
due to its completion or for some other reason. As input, a project uses requirements. As output,
a project produces a system (or part of a system). The requirements and system are work products:
things that are created, used, updated, and elaborated on throughout a project. Every work
product has a description, is of some percent complete throughout the effort, and may be
validated. However, validation is dependent on the type of work product. For example, the
requirements are validated with users in workshops, and the system is validated by being tested
against the requirements. Furthermore, requirements may be published using various types of
media, including on an intranet or in paper form; and systems may be deployed onto specific
platforms.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

No event bus listed or representation of events. Strange text artifact on the right.

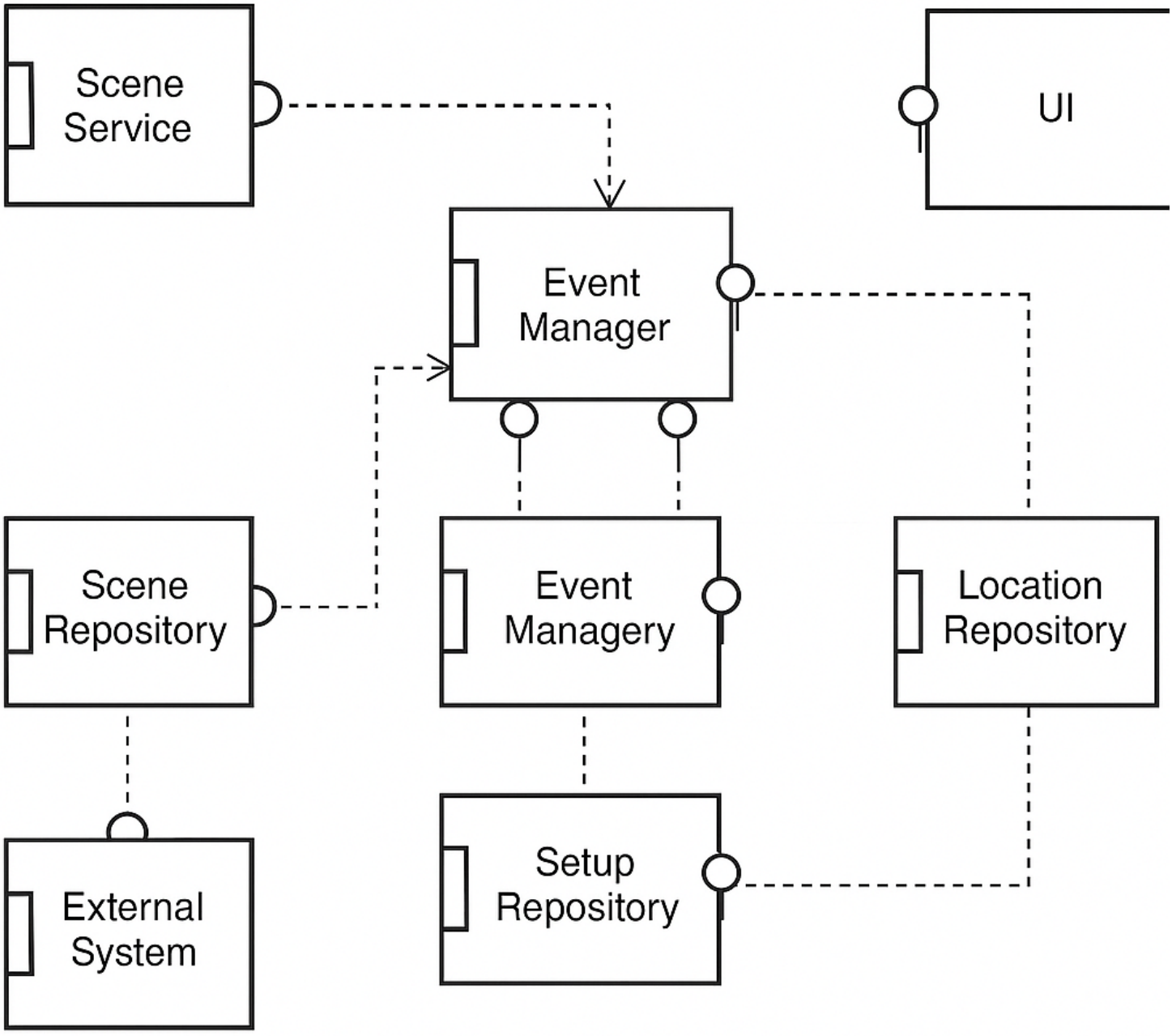
Q9.

Requirements:

```
Hollywood Approach

We are interested in building a software application to manage filmed scenes for realizing a
movie, by following the so-called “Hollywood Approach”. Every scene is identified by a code (a
string) and it is described by a text in natural language. Every scene is filmed from different
positions (at least one), each of this is called a setup. Every setup is characterized by a code
(a string) and a text in natural language where the photographic parameters are noted (e.g.,
aperture, exposure, focal length, filters, etc.). Note that a setup is related to a single scene.
For every setup, several takes may be filmed (at least one). Every take is characterized by a
(positive) natural number, a real number representing the number of meters of film that have been
used for shooting the take, and the code (a string) of the reel where the film is stored. Note
that a take is associated to a single setup. Scenes are divided into internals that are filmed in
a theater, and externals that are filmed in a location and can either be “day scene” or “night
scene”. Locations are characterized by a code (a string) and the address of the location, and a
text describing them in natural language.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q11. Please provide any other comments you might have about this component diagram:

Does have a central event hub. Missing some components related to scenes, setups, etc. Strange connections between "repositories"

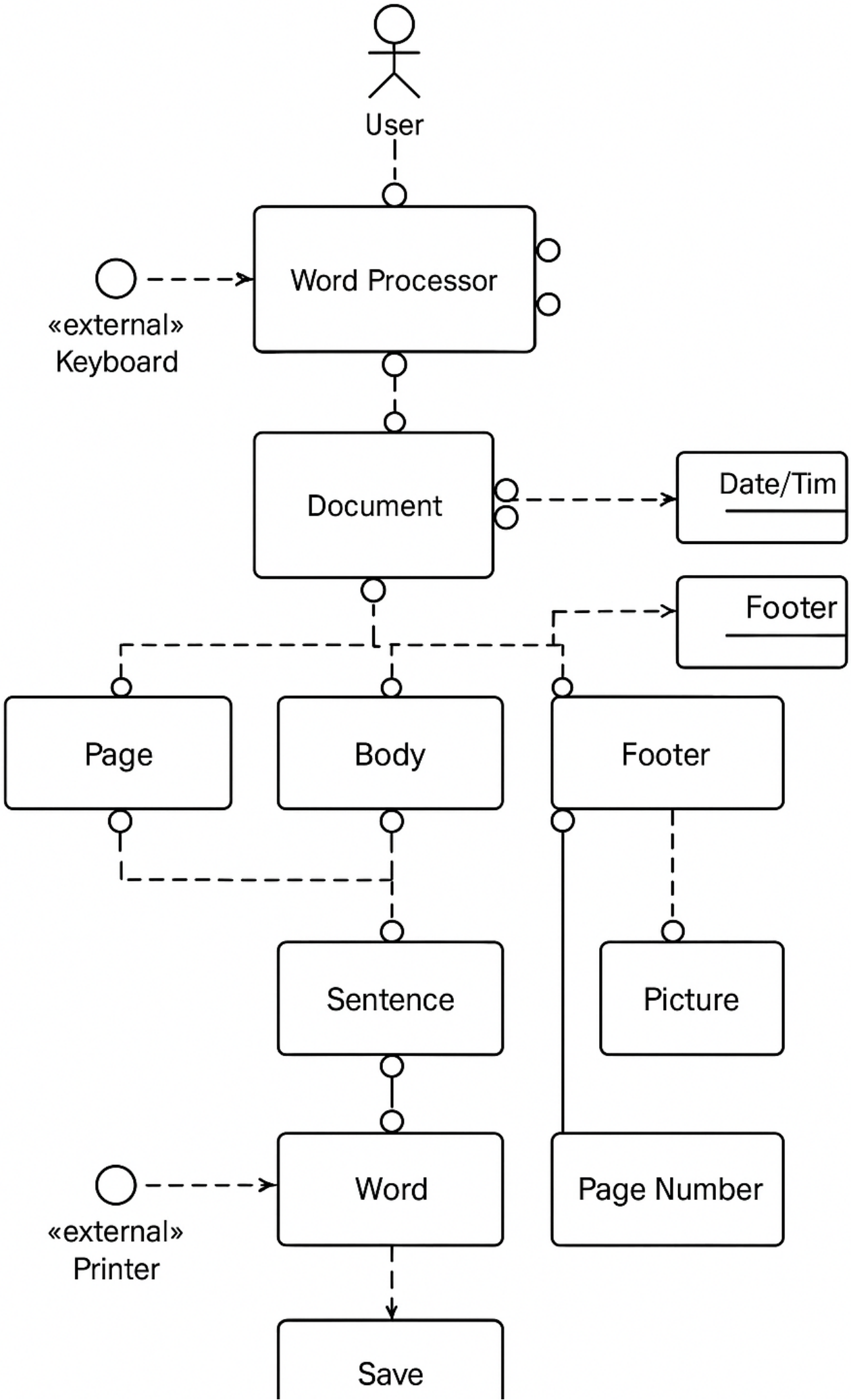
Q9.

Requirements:

```
Word Processor

A user can open a new or existing document. Text is entered through a keyboard. A document is
made up of several pages and each page is made up of a header, body and footer. Date, time and
page number may be added to header or footer. Document body is made up of sentences, which are
themselves made up of words and punctuation characters. Words are made up of letters, digits
and/or special characters. Pictures and tables may be inserted into the document body. Tables are
made up of rows and columns and every cell in a table can contain both text and pictures. Users
can save or print documents.
```

Component Diagram:


User
Word Processor
«external»
Keyboard
Document
Date/Tim
Footer
Page
Body
Footer
Sentence
Picture
Word
Page Number
«external»
Printer
Save

Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

No events mentioned at all, still just a class diagram.

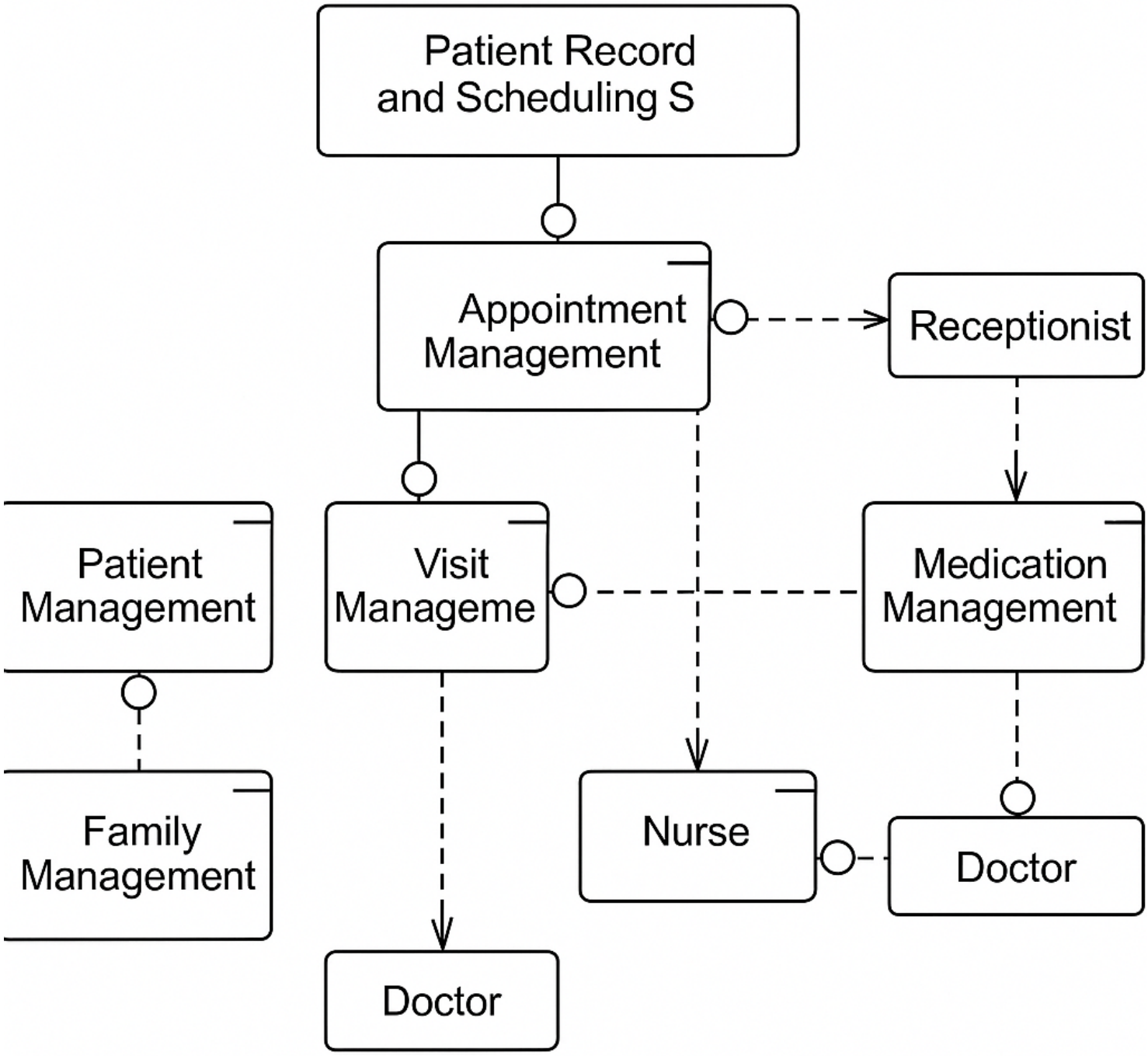
Q9.

Requirements:

Patient Record and Scheduling System

A patient record and scheduling system in a doctor’s office is used by the receptionists, nurses, and doctors. The receptionists use the system to enter new patient information when first time patients visit the doctor. They also schedule all appointments. The nurses use the system to keep track of the results of each visit including diagnosis and medications. For each visit, free form text fields are used captures information on diagnosis and treatment. Multiple medications may be prescribed during each visit. The nurses can also access the information to print out a history of patient visits. The doctors primarily use the system to view patient history. The doctors may enter some patient treatment information and prescriptions occasionally, but most frequently they let the nurses enter this information. -- Each patient is assigned to a family. The head of family is responsible for the person with the primary medical coverage. Information about doctors is maintained since a family has a primary care physician, but different doctors may be the ones seeing the patient during the visit.

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

No event bus

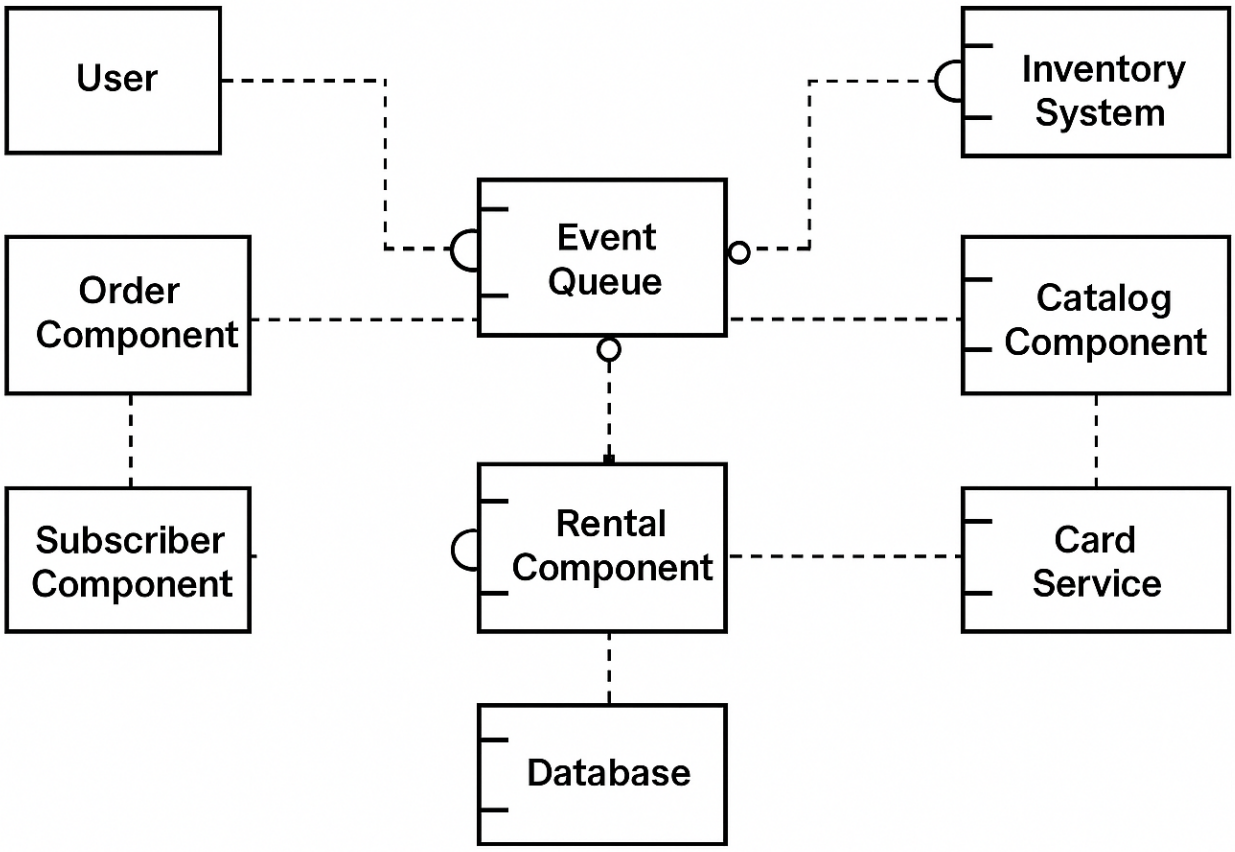
Q9.

Requirements:

Movie-Shop

♣ Design a system for a movie-shop, in order to handle ordering of movies and browsing of the catalogue of the store, and user subscriptions with rechargeable cards. ♣ Only subscribers are allowed hiring movies with their own card. ♣ Credit is updated on the card during rent operations. ♣ Both users and subscribers can buy a movie and their data are saved in the related order. ♣ When a movie is not available it is ordered .

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	◉	○

Q11. Please provide any other comments you might have about this component diagram:

Central event broker. Some odd connections among other components

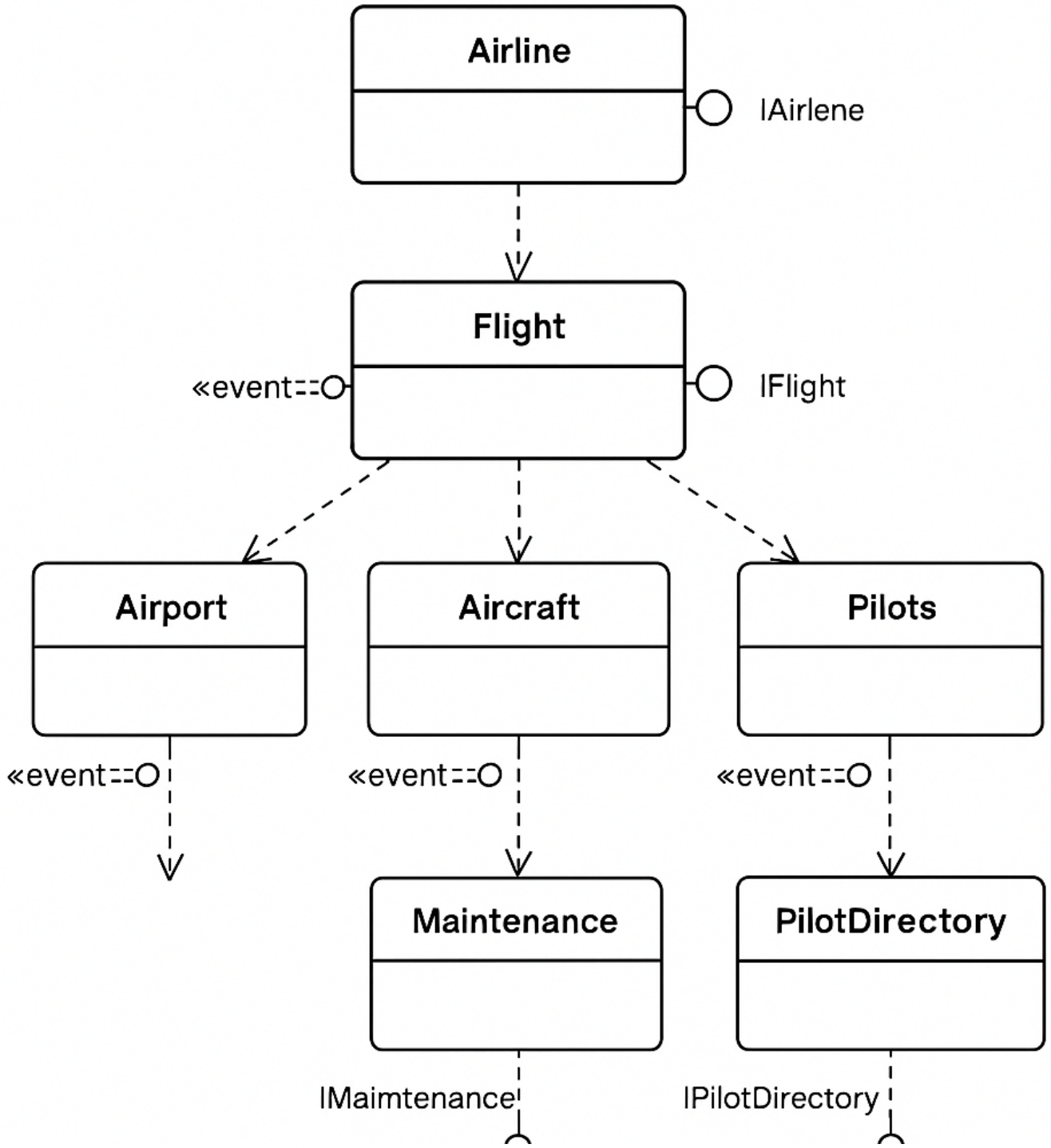
Q9.

Requirements:

```
Flights

We want to model a system for management of flights and pilots. An airline operates flights. Each
airline has an ID. Each flight has an ID a departure airport and an arrival airport: an airport
as a unique identifier. Each flight has a pilot and a co-pilot, and it uses an aircraft of a
certain type; a flight has also a departure time and an arrival time. An airline owns a set of
aircrafts of different types. An aircraft can be in a working state or it can be under repair. In
a particular moment an aircraft can be landed or airborne. A company has a set of pilots: each
pilot has an experience level: 1 is minimum, 3 is maximum. A type of aeroplane may need a
particular number of pilots, with a different role (e.g.: captain, co-pilot, navigator): there
must be at least one captain and one co-pilot, and a captain must have a level 3.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Some lip service to events, but notation is strange. Really weird changes to the text here.

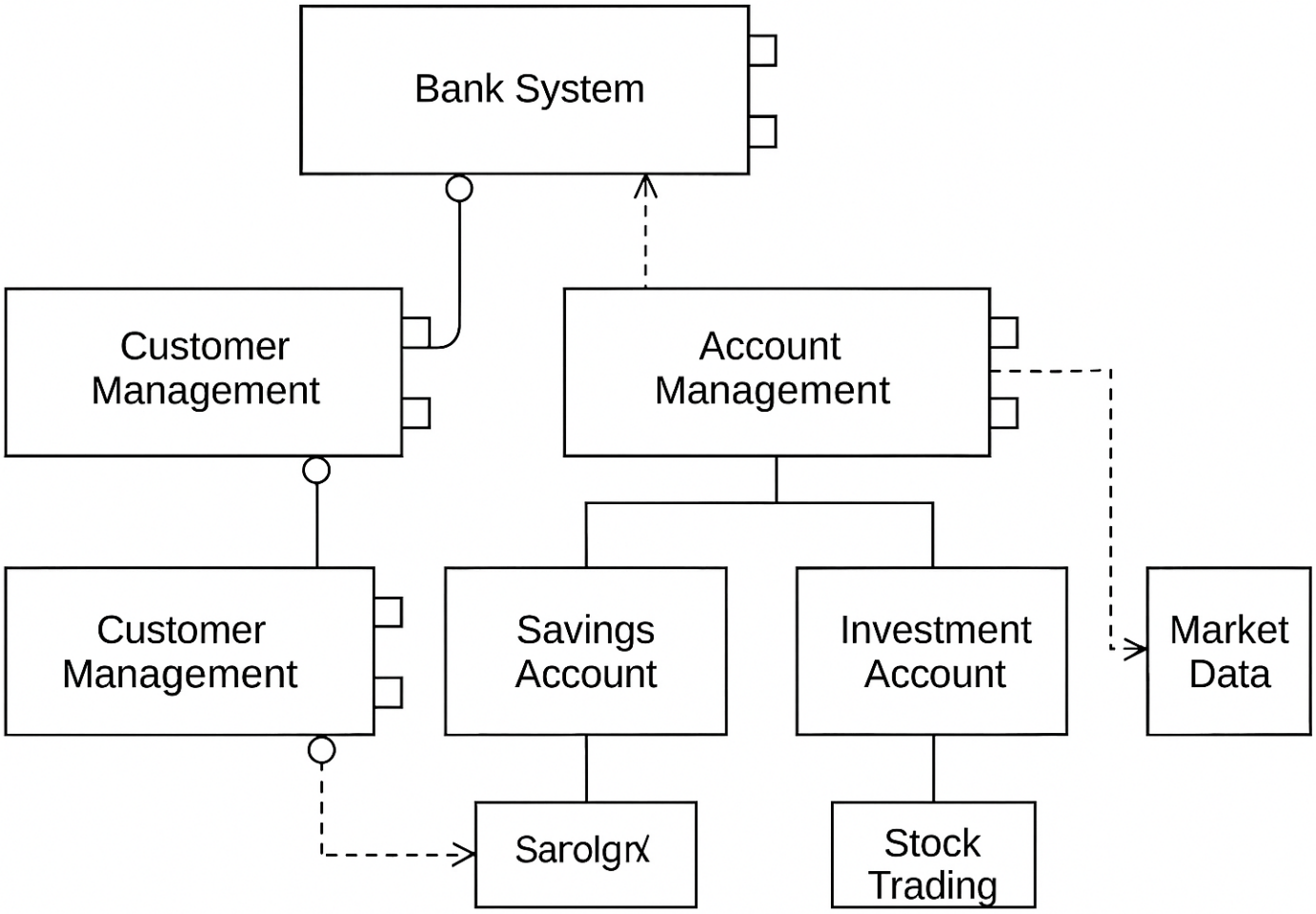
Q9.

Requirements:

```
Bank System

A bank system contains data on customers (identified by name and address) and their accounts.
Each account has a balance and there are 2 type of accounts: one for savings which offers an
interest rate, the other for investments, used to buy stocks. Stocks are bought at a certain
quantity for a certain price (ticker) and the bank applies commission on stock orders.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○

Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

No central event bus. Covers the other types of account ok, strange component at the bottom

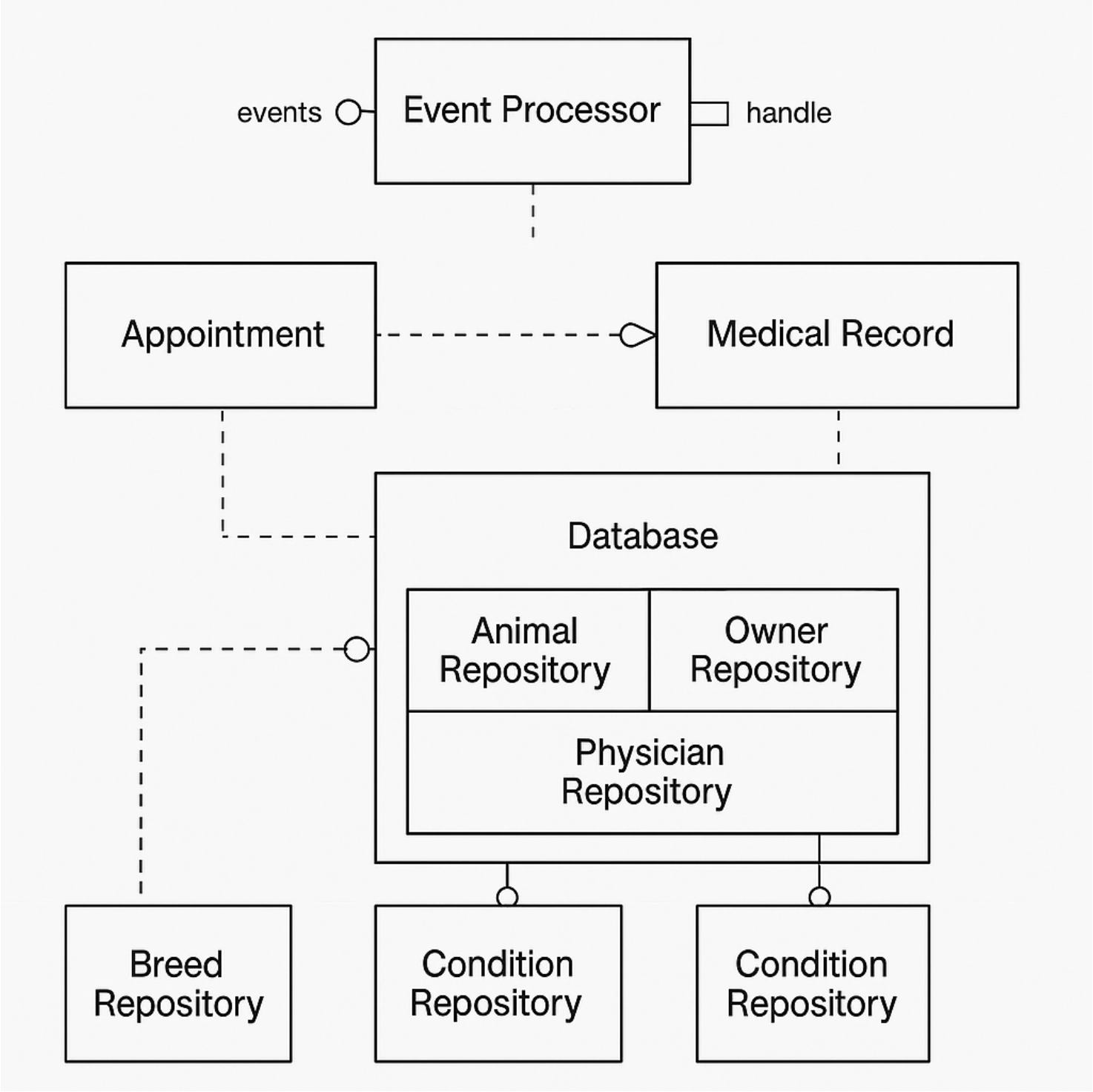
Q9.

Requirements:

```
Veterinary Clinic

The owner of a veterinary clinic wants to create a database to store information about all
veterinary services performed. After some research he came up with the following requirements:
● For each admitted animal, its name, breed (if any) and owner must be stored. Each animal should
be given an unique numeric identifier.
● For each owner, its name, address and phone number should be stored. An unique numeric
identifier should also be generated for each one of them.
● An animal might be owner-less. This happens frequently as the clinic often rescues abandoned
dogs from the streets in order to treat them and get them new owners.
● It should be possible to store information about a specific breed even if no animals of that
breed have been treated at the clinic.
● Each appointement always has a responsible physician. All appointements start at a certain date
and time; and are attended by an animal (and of course its owner).
● For each physician, his name, address and phone number should be stored. An unique numeric
identifier should also be generated for each one of them.
● In an appointement, several medical conditions might be detected. Each condition has a common
name and a scientific name. No two conditions have the same scientific name.
● It should be possible to store information about the most common conditions for each different
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Disconnected event bus at the top, not used by the rest

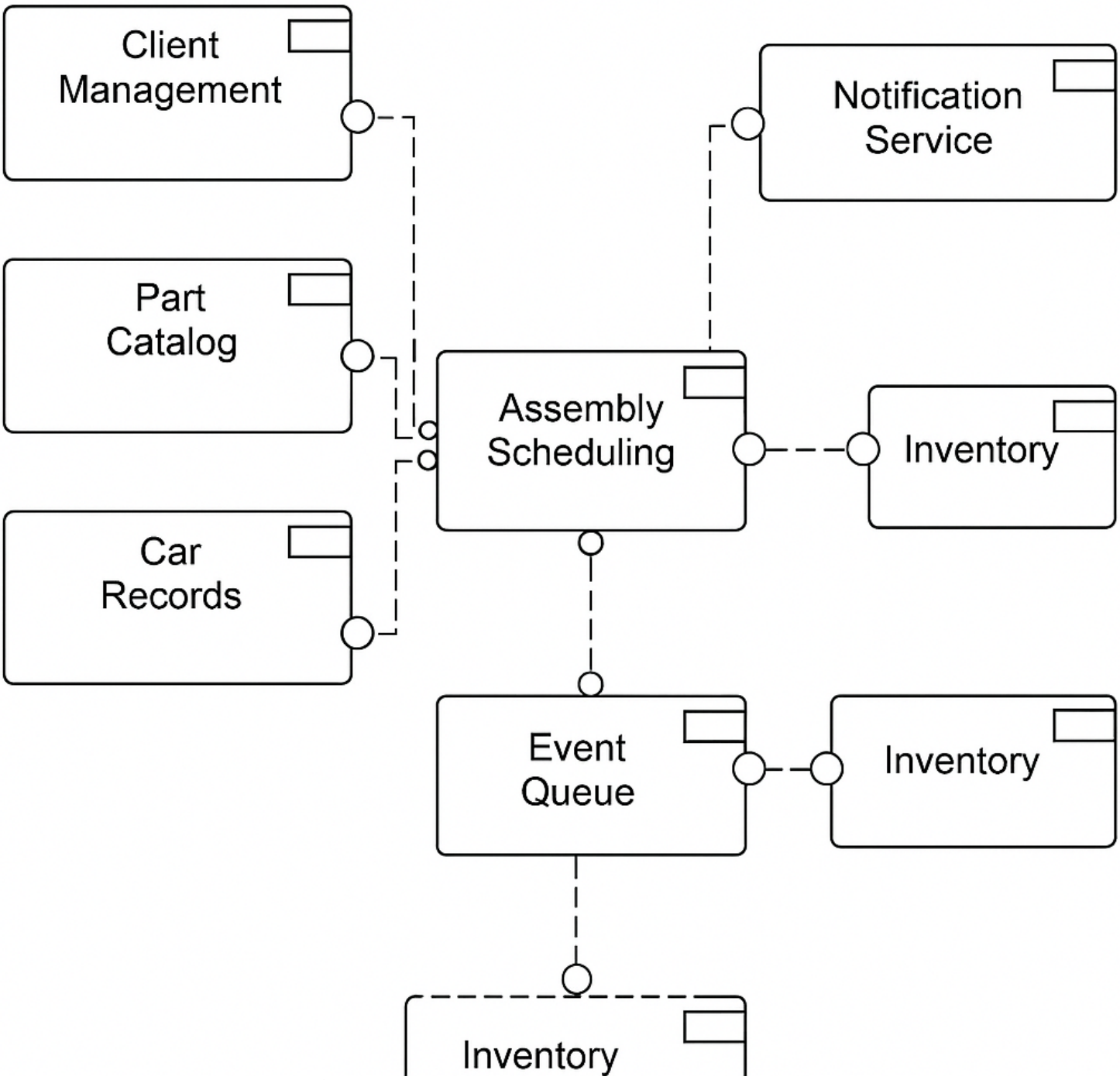
Q9.

Requirements:

```
Auto Repair

An auto repair shop, that sells and mounts parts and accessories for all kinds of vehicles, wants
a new information system to manage their clients, parts, accessories and assembly services:
● There are several employees. Each one of them has an unique identifying number, a name and an
address.
● In this shop, assembly services, where parts and accessories are installed in a vehicle, are
executed. For each one these services the following data must be stored: In which car the service
was executed, how many kms had the car at the time, who was the responsible employee, which parts
and accessories were fitted, how many work hours did it take and the admission and finish dates.
● Parts and accessories are only sold together with an assembly service.
● Each part/accessory only fits in some car models. Therefore, it is important to store that
information.
● Each part/accessory has a category (radio, tyre, …), a serial number and a price.
● Each car has a license plate, a make, a model, a color and an owner. Each owner has a name,
identifying number, address and a phone.
● One person can own more than one car but one car only has one owner.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Includes an event bus as a side component, does not seem used by the main processes

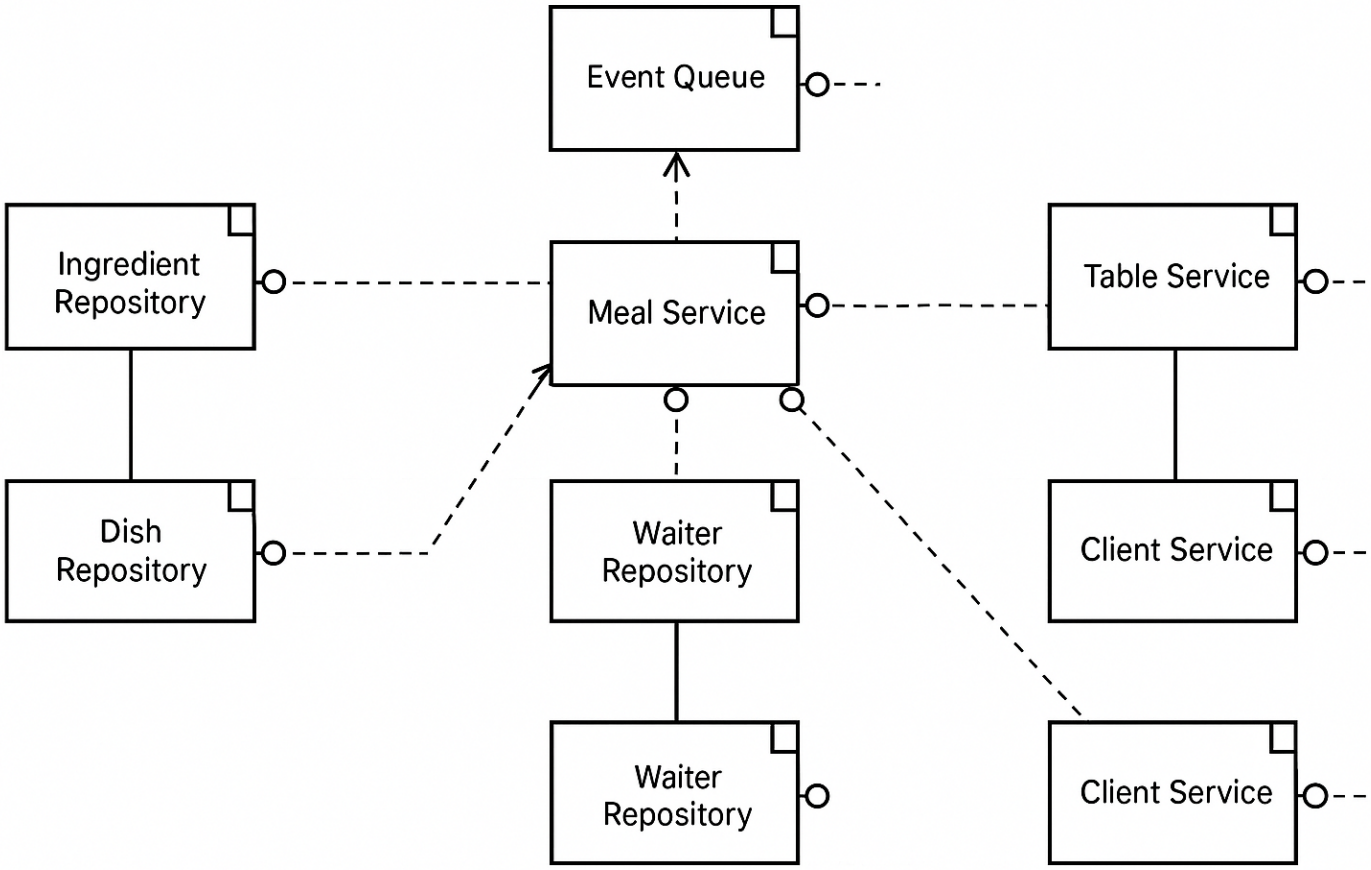
Q9.

Requirements:

```
10. Restaurant

The owner of a small restaurant wants a new information system to store data for all meals
consumed there and also to keep a record of ingredients kept in stock. After some research he
reached the following requirements list:
• Each ingredient has a name, a measuring unit (e.g. olive oil is measured in liters, while eggs
are unit based) and a quantity in stock. There are no two ingredients with the same name.
• Each dish is composed of several ingredients in a certain quantity. An ingredient can, of
course, be used in different dishes.
• A dish has an unique name and a numeric identifier.
• There are several tables at the restaurant. Each one of them has an unique numeric identifier
and a maximum ammount of people that can be seated there.
• In each meal, several dishes are consumed at a certain table. The same dish can be eaten more
than once in the same meal.
• A meal takes place in a certain date and has a start and end time. Each meal has a responsible
waiter.
• A waiter has an unique numerical identifier, a name, an address and a phone number.
• In some cases it is important to store information about the client that consumed the meal. A
client has a tax identification number, a name and an address.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	◉	○

Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	◉	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

There is an event bus and other services, but only one service uses the events.

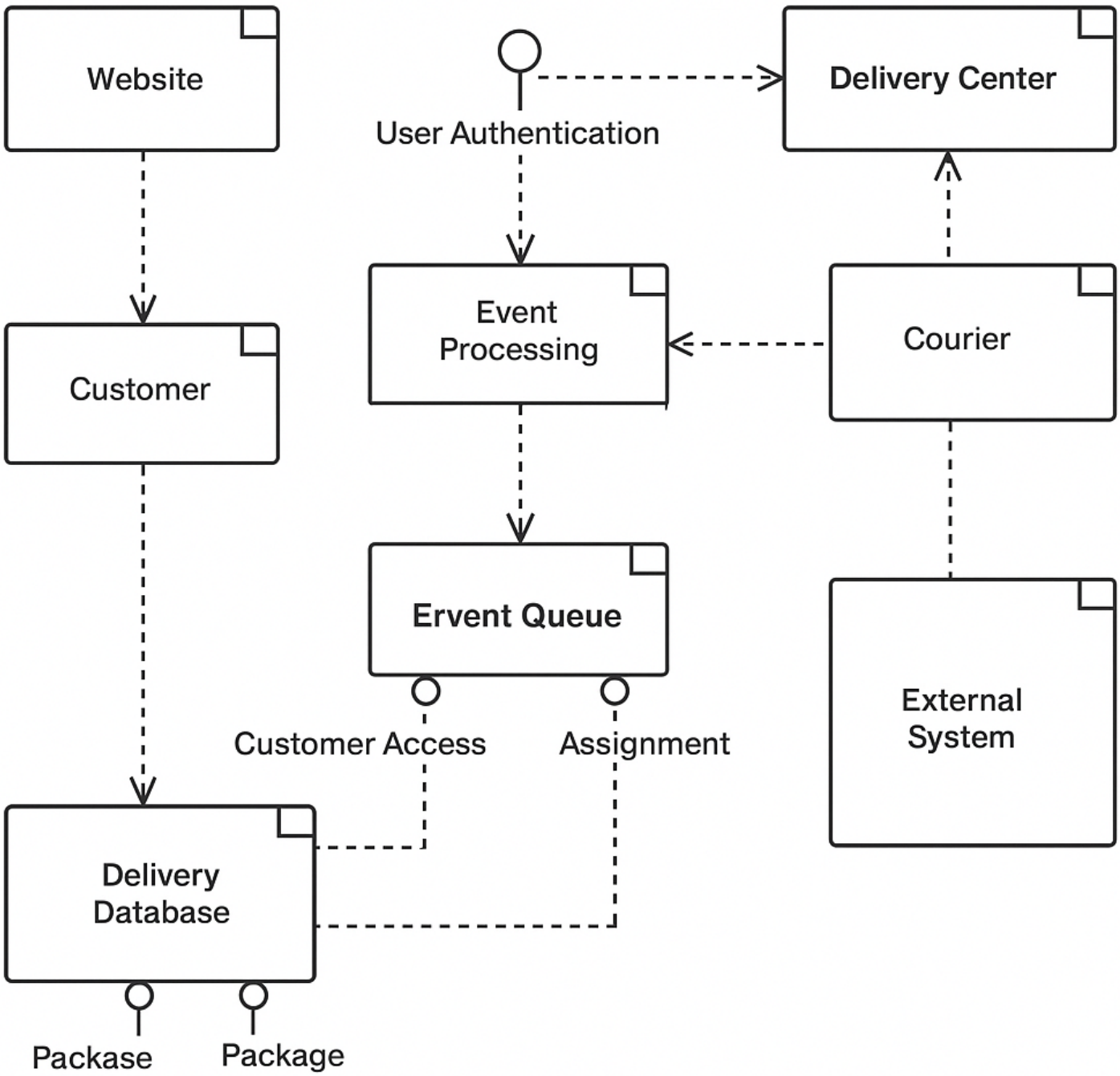
Q9.

Requirements:

```
Deliveries

The owner of a small delivery company plans to have an information system that allows him to save
data about his customers and deliveries. After some time studying the problem, he reached the
following requirements:
● Each customer has a VAT number, a name, a phone number and an address. There are no two clients
with the same VAT number.
● When a customer wants to send a package to another customer, he just has to login to the
company website, select the customer he wants to send the package to, enter the package's weight
and if the delivery is normal or urgent. He then receives an unique identifier code that he
writes on the package.
● The package is then delivered by the customer at the delivery center of his choosing. A
delivery center has a unique name and an address.
● Each client has an associated delivery center. This delivery center is chosen by the company
and it is normally the one closest to the customer's house.
● The package is them routed through an internal system until it reaches the delivery center of
the recipient.
● The package is then delivered by hand from that delivery center to the recipient by a courier.
● Couriers have a single VAT number, a name and a phone number. Each courier works in a single
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q11. Please provide any other comments you might have about this component diagram:

Generally clear event workflow. Connects orders and couriers, at least

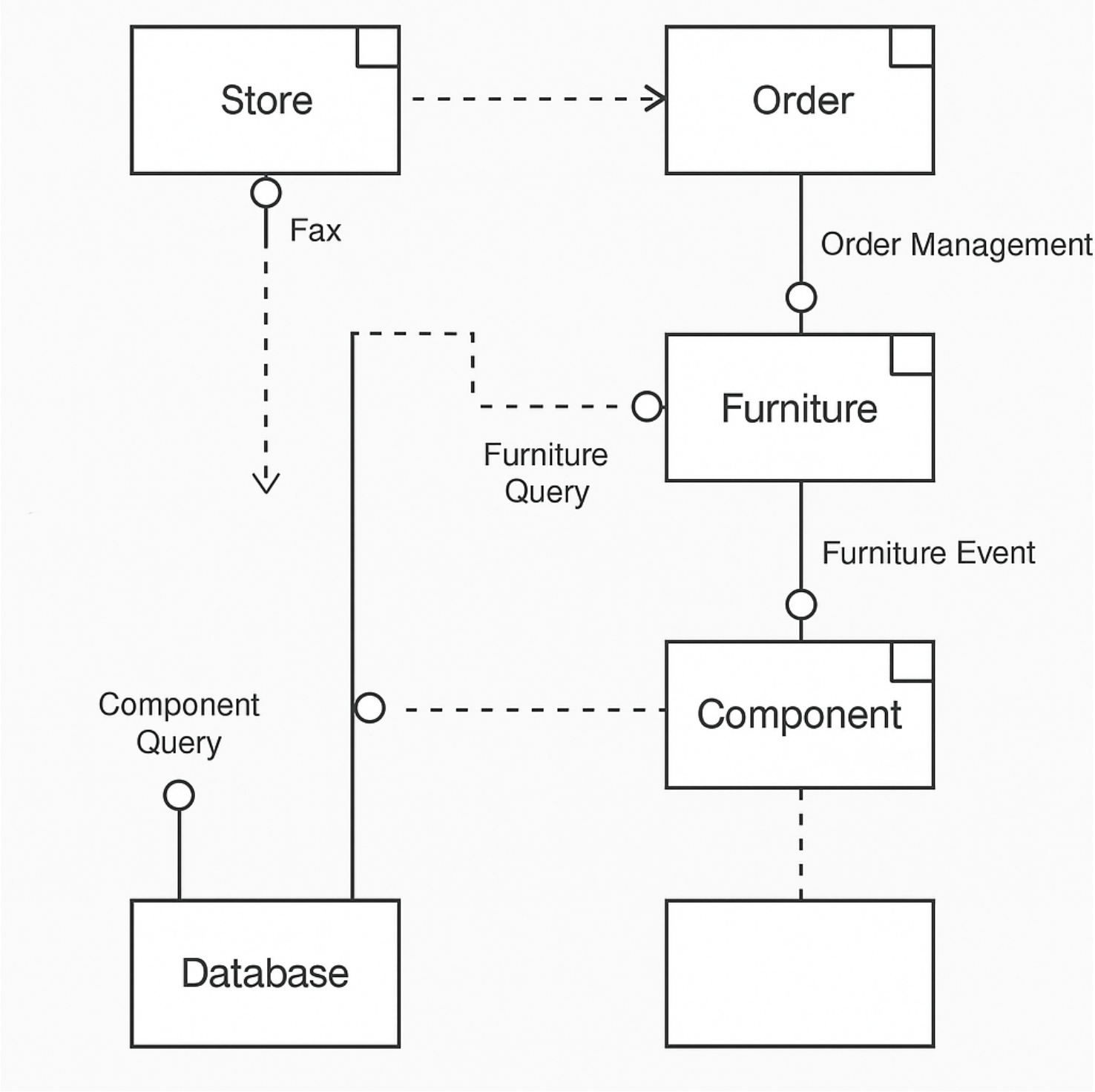
Q9.

Requirements:

```
Furniture

The known furniture factory Hi-Key-Ah, intends to implement an information system to store all
data on the different types of furniture and components it produces:
● The factory produces several lines of furniture, each with a different name and consisting of
several pieces of furniture of different types (beds, tables, chairs, …).
● All furniture pieces have a type, a single reference (eg CC6578) and a selling price.
● The major competitive advantage of this innovative plant is the fact that each component
produced can be used in more than one piece of furniture.
● Each piece of furniture is thus composed of several components. The same component can be used
more than once in the same piece.
● Every type of component produced is assigned a unique numerical code, a manufacturing price and
a type (screw, hinge, shelf …).
● The furniture is then sold in various stores throughout the world. Each store has a different
address and a fax number.
● To make the manufacturing process more efficient, stores have to place orders everytime they
need to replenish their stock. These orders must also be stored in the database.
● Each order has a order number, a date, the store that placed the order as well as a list of all
the ordered furniture and their quantities.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	◉	○	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Overall bad. Strange lines, empty components, no event bus

Q9.

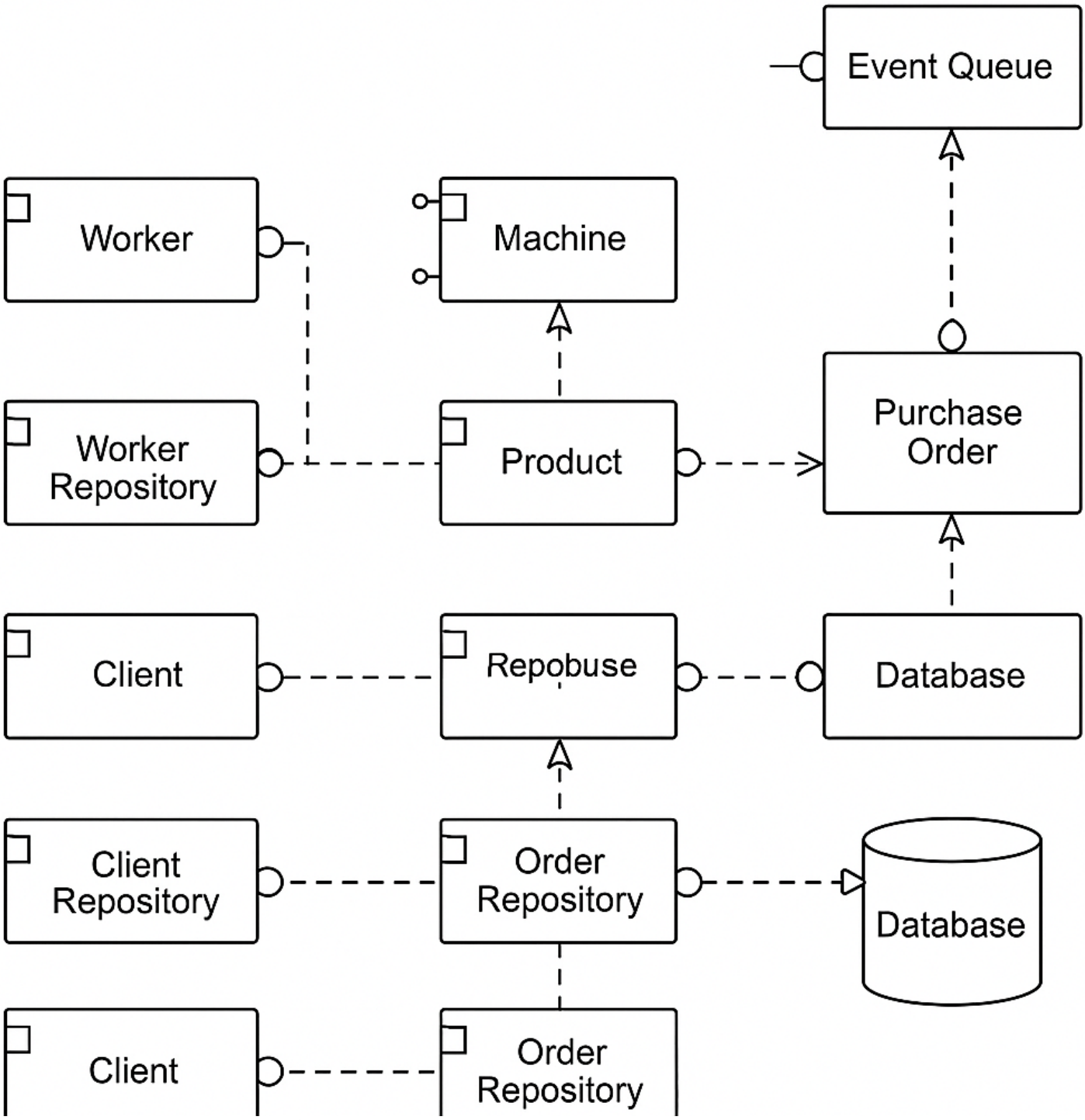
Requirements:

Factory

Create a database for a factory with the following requirements. Don’t forget to add unique identifiers for each one of the entities if needed.

- A factory has several machines. Each one of them is operated by several workers.
- A worker might work in more than one machine.
- In this factory, several products of different types, are produced. Each different type of product is produced in a single machine. But, the same machine can produce more than one type of product.
- Products from the same type are all produced from the same single material and have the same weigth.
- Clients can issue purchase orders. Each order has a list of the desired products and their quantity.
- For each worker, the following data should be stored in the database: name (first and last), birth date, address and a list of his skills.
- For each machine, the following data should be stored: serial number, make, model and purchase date.
- For each client, the followig data should be stored: name, address, phone number and name of the contact person (if any).

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	◉	○	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	◉	○	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Overall bad. Duplicate components, poorly connected event queue

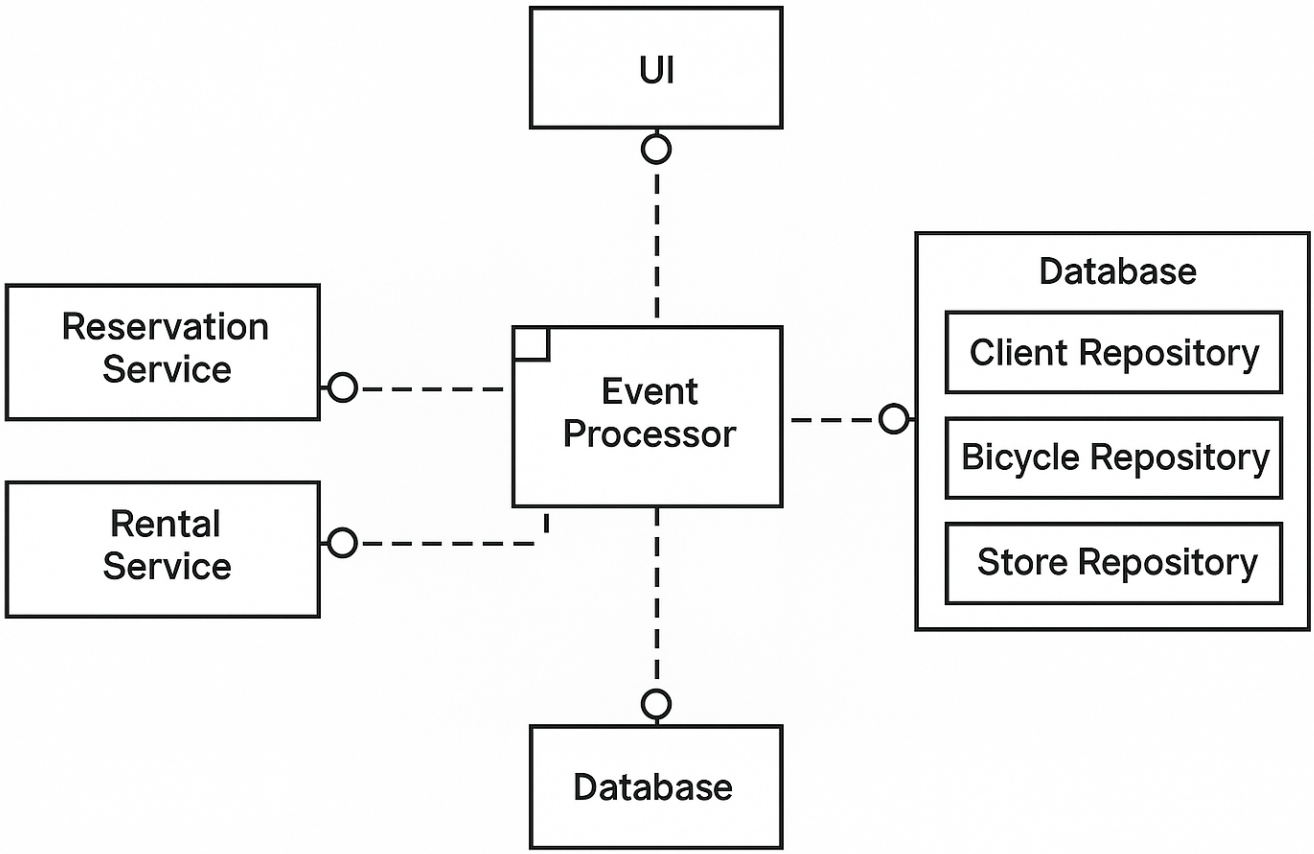
Q9.

Requirements:

```
Bycicle Rental

A bicycle renting company wants to create an information system that allows it to store the data
regarding all their reservations and rentals. The system should follow these requirements:
● It should be possible to store the national id number (NIN), tax identification number (TIN),
name and address for every client. The NIN and TIN must be different for every client and all
clients should have at least a TIN and a name.
● The database should also contain information about the bicycle models that can be rented- Each
model has an unique name, a type (that can only be road, mountain, bmx or hybrid) and the number
of gears.
● Each bicycle has a unique identifying number and a model.
● The company has several different stores where bicycles can be picked up and returned. Each one
of these stores is identified by an unique name and has an address (both mandatory).
● When a reservation is made, the following data must be known: which client did the reservation,
when will he pick up the bike (day), which bike model he wants and where will he pick up the bike
(store).
● When a bike is picked up, the actual bike that was picked up must be stored in the database.
● When a bike is returned, the return date should be stored in the database.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	○	○	◉
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	○	○	◉
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	○	○	◉

Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	○	◉
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	○	○	◉

Q11. Please provide any other comments you might have about this component diagram:

This one is nice. Would include a "client" component, drop the spare database

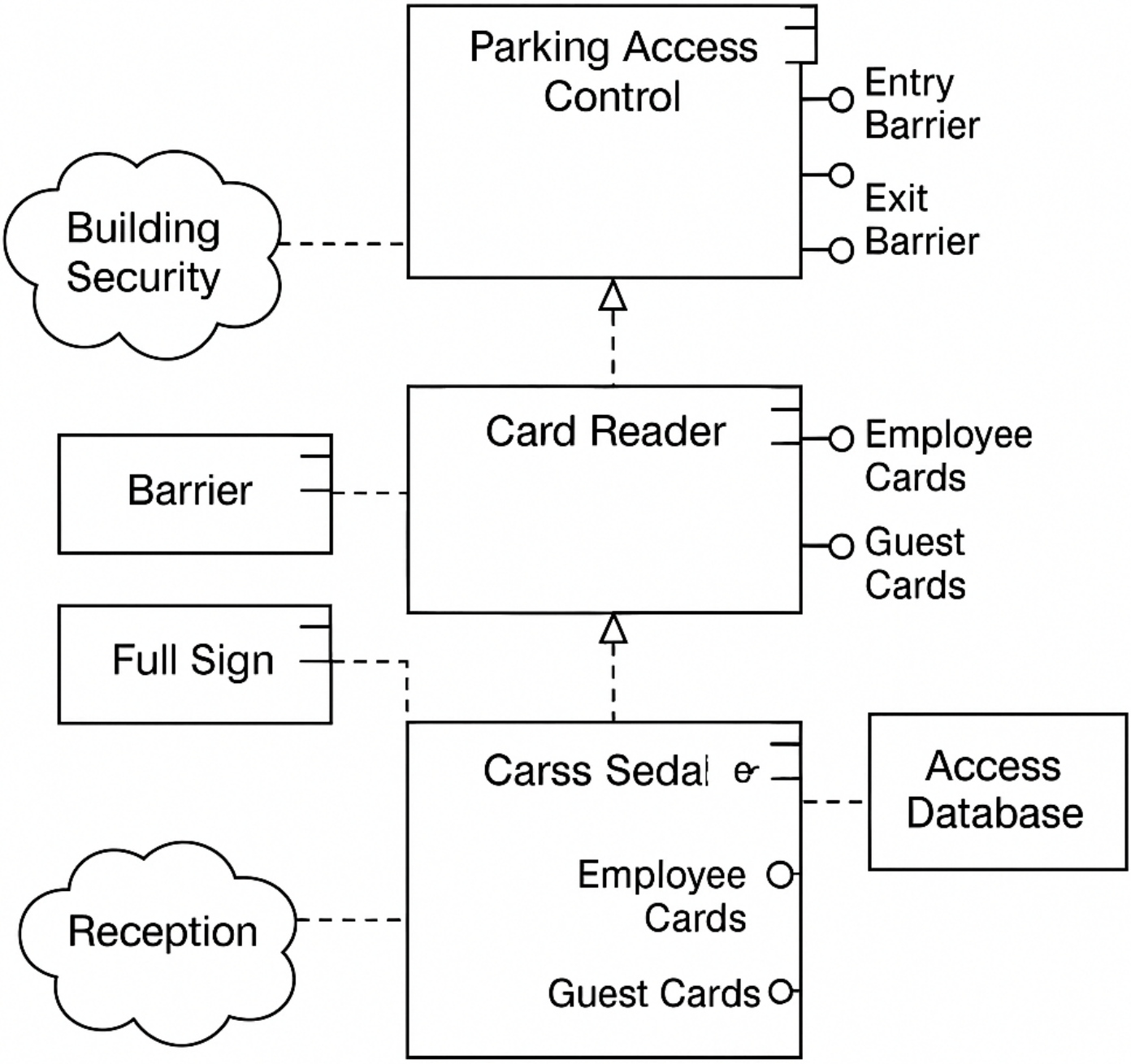
Q9.

Requirements:

```
Saturn Int. Management

Saturn Int. management wants to improve their security measures, both for their building and on
site. They would like to prevent people who are not part of the company to use their car park.
Saturn Int. has decided to issue identity cards to all employees. Each card records the name,
department and number of a company staff, and give them access to the company car park. Employees
are asked to wear the cards while on the site.
There is a barrier and a card reader placed at the entrance to the car park. When a driver drives
his car into the car park, he/she inserts his or her identity card into the card reader. The card
reader then verify the card number to see if it is known to the system. If the number is
recognized, the reader sends a signal to trigger the barrier to rise. The driver can then drive
his/her car into the car park.
There is another barrier at the exit of the car park, which is automatically raised when a car
wishes to leave the car park.
A sign at the entrance display “Full” when there are no spaces in the car park. It is only
switched off when a car leaves.
There is another type of card for guests, which also permits access to the car park. The card
records a number and the current date. Such cards may be sent out in advance, or collected from
reception. All guest cards must be returned to reception when the visitor leaves Saturn Int.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	◉	○	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

No event bus, odd clouds? Not exactly sure what it all means

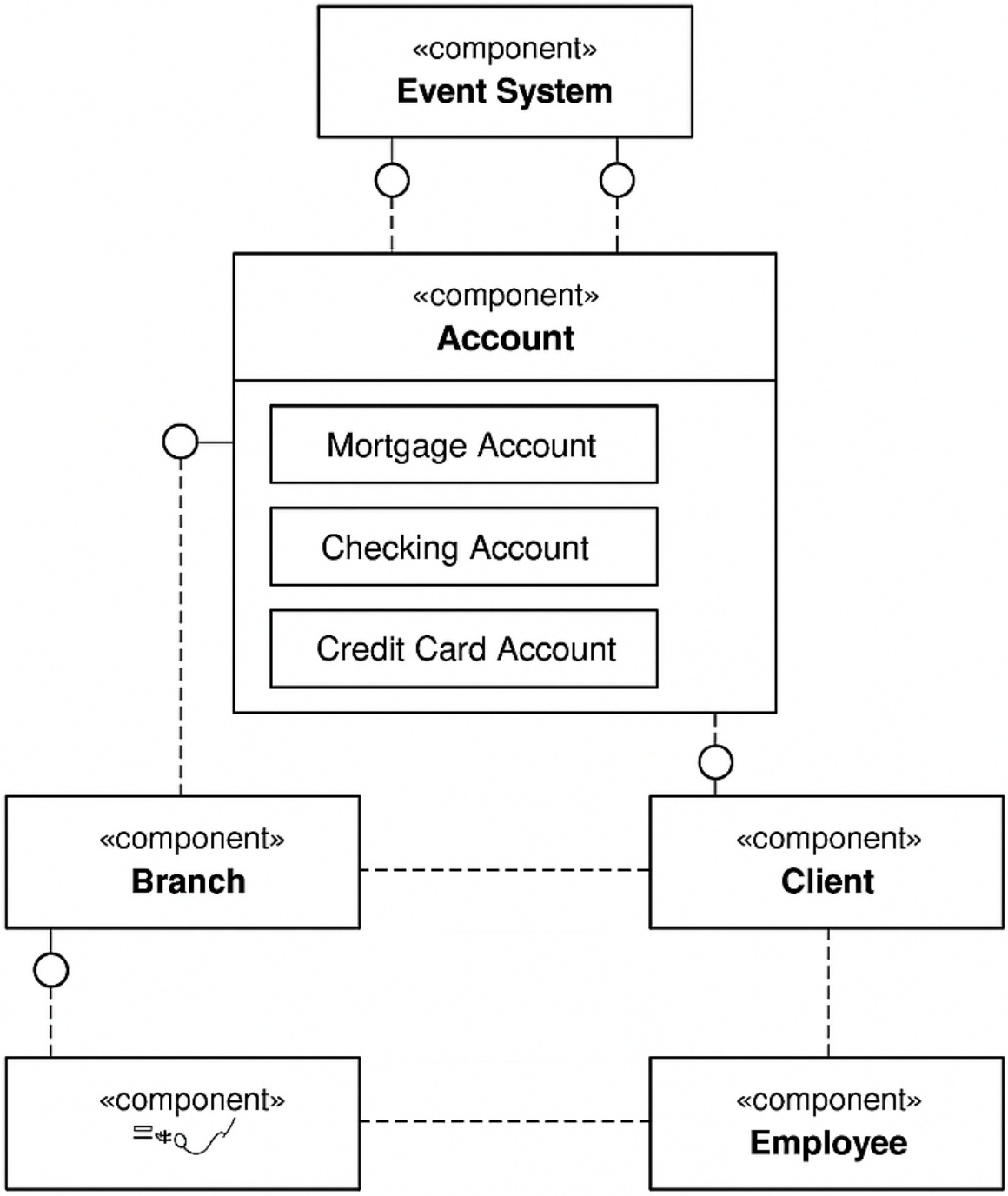
Q9.

Requirements:

```
OOBank

This system provides the basic services to manage bank accounts at a bank called OOBank. OOBank
has many branches, each of which has an address and branch number. A client opens accounts at a
branch. Each account is uniquely identified by an account number; it has a balance and a credit
or overdraft limit. There are many types of accounts, including: a mortgage account (which has a
property as collateral), a checking account, and a credit card account (which has an expiry date
and can have secondary cards attached to it). It is possible to have a joint account (e.g. for a
husband and wife). Each type of account has a particular interest rate, a monthly fee and a
specific set of privileges (e.g. ability to write checks, insurance for purchases etc.). OOBank
is divided into divisions and subdivisions (such as Planning, Investments and Consumer); the
branches are considered subdivisions of the Consumer Division. Each division has a manager and a
set of other employees. Each customer is assigned a particular employee as his or her 'personal
banker'.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Event bus doesn't mean anything here. Components have multiple connections to each other, overall odd

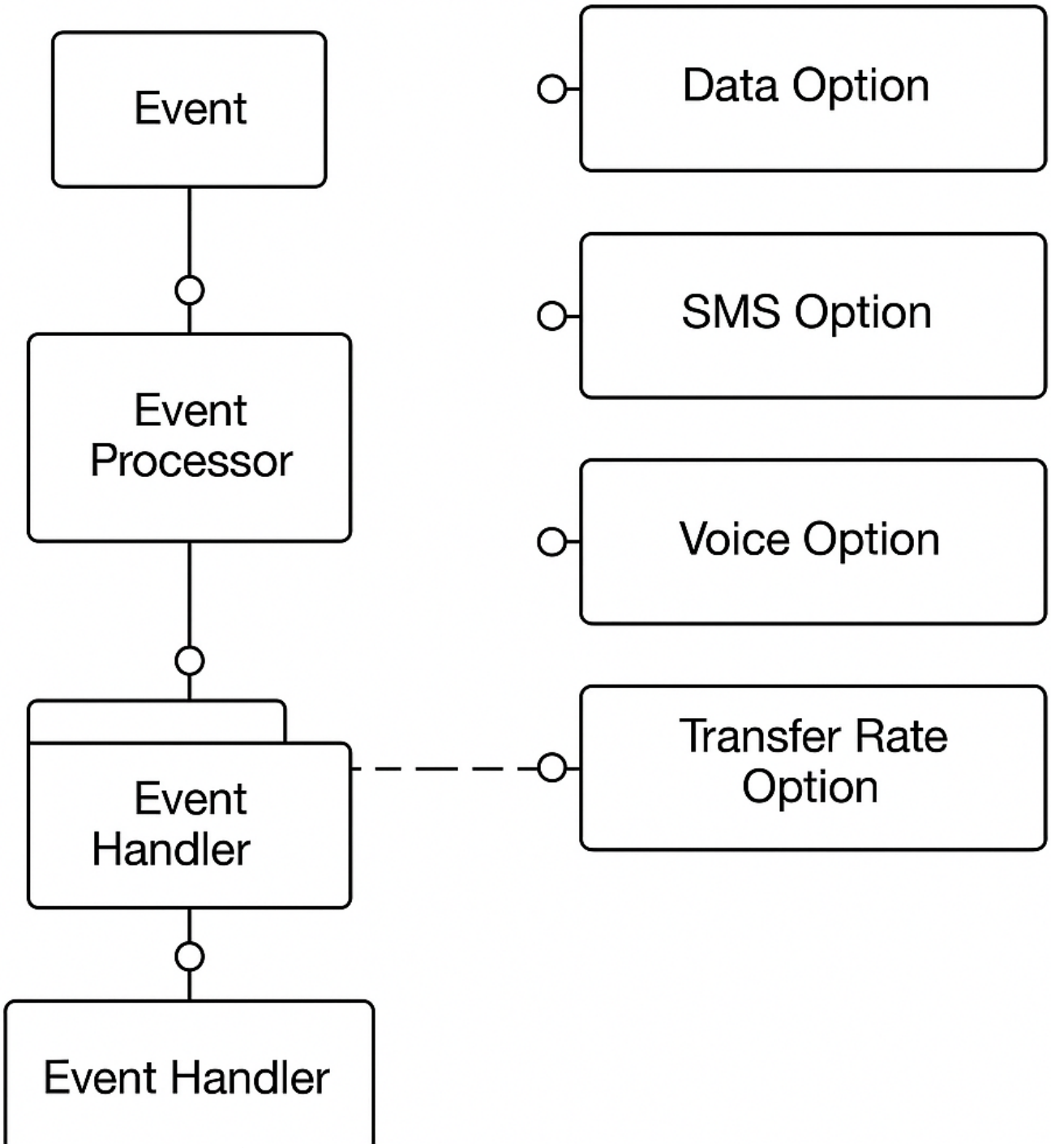
Q9.

Requirements:

```
Prepaid Cell Phone

The contract of a prepaid cell phone should be modelled and implemented. A basic contract has a
contract number (of type int) and a balance (of type double), but no monthly charges. The
contract number is not automatically generated, but is to be set as a parameter by the
constructor as well as the initial balance. The balance has a getter and a setter. The following
options can be added to a contract (if needed also several times):
● 100 MB of data (monthly charge 1.00€)
● 50 SMS (monthly charge 0.50€)
● 50 minutes (monthly charge 1.50€)
● Double Transfer Rate (monthly charge 2.00€) implement this requirement with the help of the
decorator pattern. All contract elements should be able to understand the methods
getCharges():double, getBalance():double and setBalance(double).
The method getCharges() should provide the monthly charge of a contract with all its options
selected. The methods getBalance() and setBalance() should be passed through and access the basic
contract.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Event bus seems there just to satisfy the words. No real connection. Disconnected services

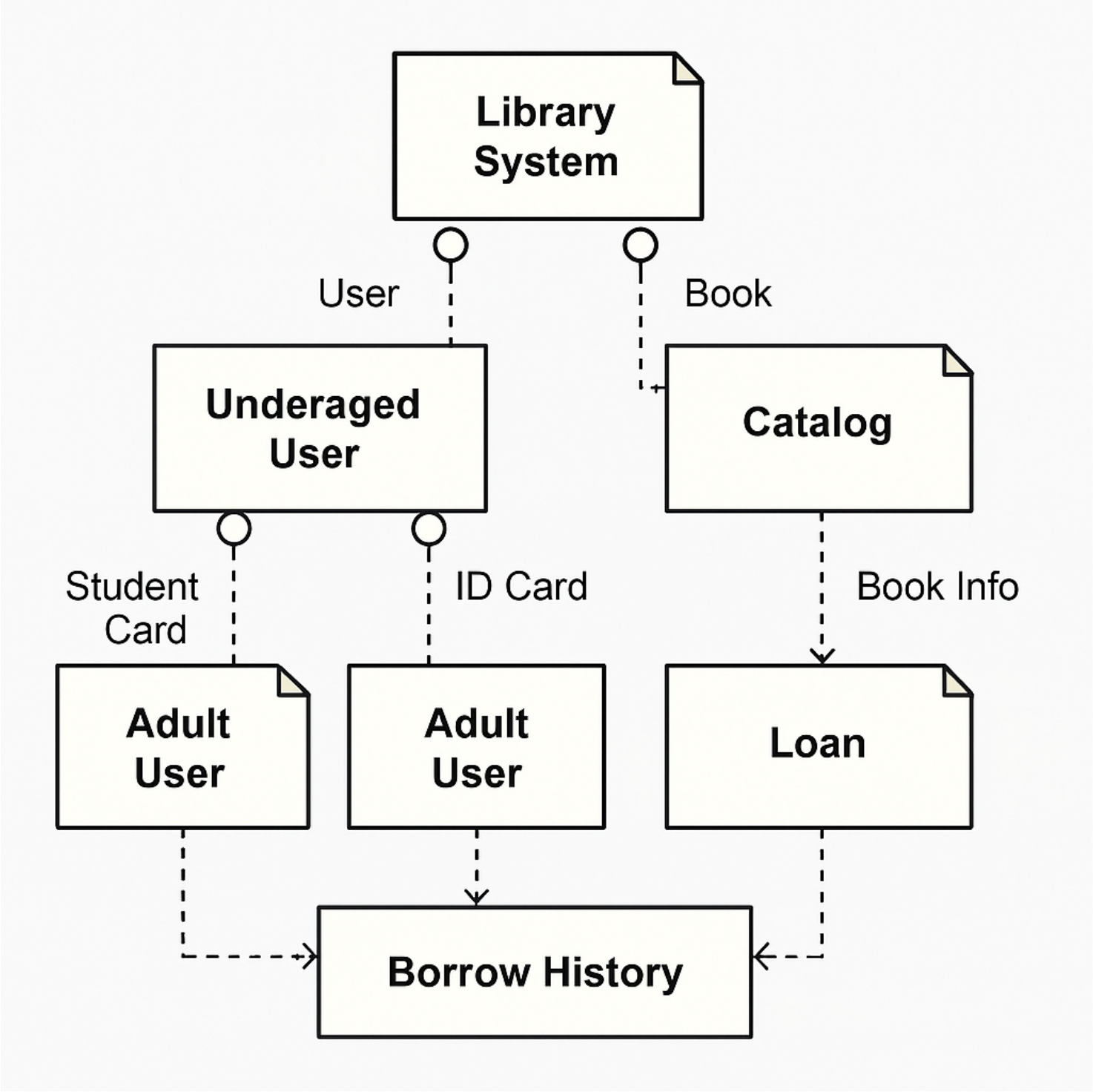
Q9.

Requirements:

```
Library System

The exercise id to design a class structure for a library system. It should fulfil those
requirements:
● There are two type of users – under-aged and adults.
● Under-aged users are identified with usage of their full name and student card.
● Adult users are identified with usage of their full name and ID card.
● The library contains books.
● There is basic information about every book (title, author, etc).
● The user can borrow at most 4 books at the same time.
● There is a history of previously borrowed books for every user (along with all the dates).
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	◉	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	◉	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

No event bus. Seems to have most of the pieces, but hard to see connections for a system

Q9.

Requirements:

MyDoctor

The MyDoctor application aims to be a management tool for the appointments of a doctor.
A hospital has multiple offices.
The users of the application can be doctors and patients.
The doctors can apply to practice in offices and create a schedule for an office. The schedules in different offices can't overlay.

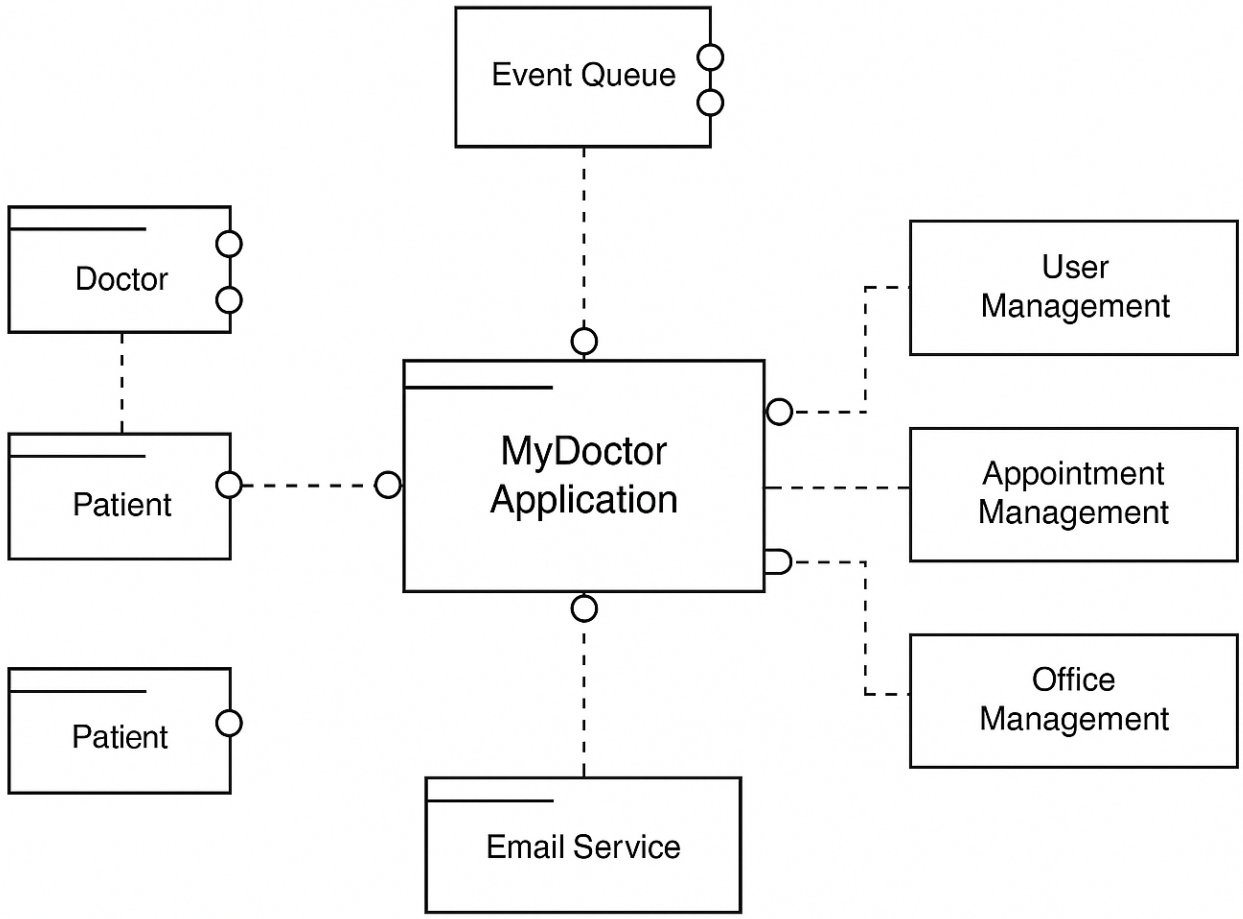
📝 Example:
Doctor Ana is available in Office 4 on the 4th of September during 1 PM - 5PM.
Doctor Ana can't practice in Office 5 on the 4th of September during 3PM - 8 PM, but she can practice in Office 5 on the 4th of September during 5:30PM - 8 PM.
The patients can see the existing doctors in the system, the schedule of the offices and can book appointments for specific doctors and for specific schedules. The appointments can be of 3 types:

- Blood Test - 15 mins
- Consultation - 30 mins
- Surgery - 60 mins

The booking of an appointment will not be possible if another appointment is already booked at the same time frame. An email is sent to the patient with the confirmation of the appointment.

📝 Example:

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled
Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	○	◉	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	○	○	◉	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	○	◉	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	○	○	◉	○	○

Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	○	○	◉	○	○

Q11. Please provide any other comments you might have about this component diagram:

Not too bad, if we argue the event queue is managed by the central service

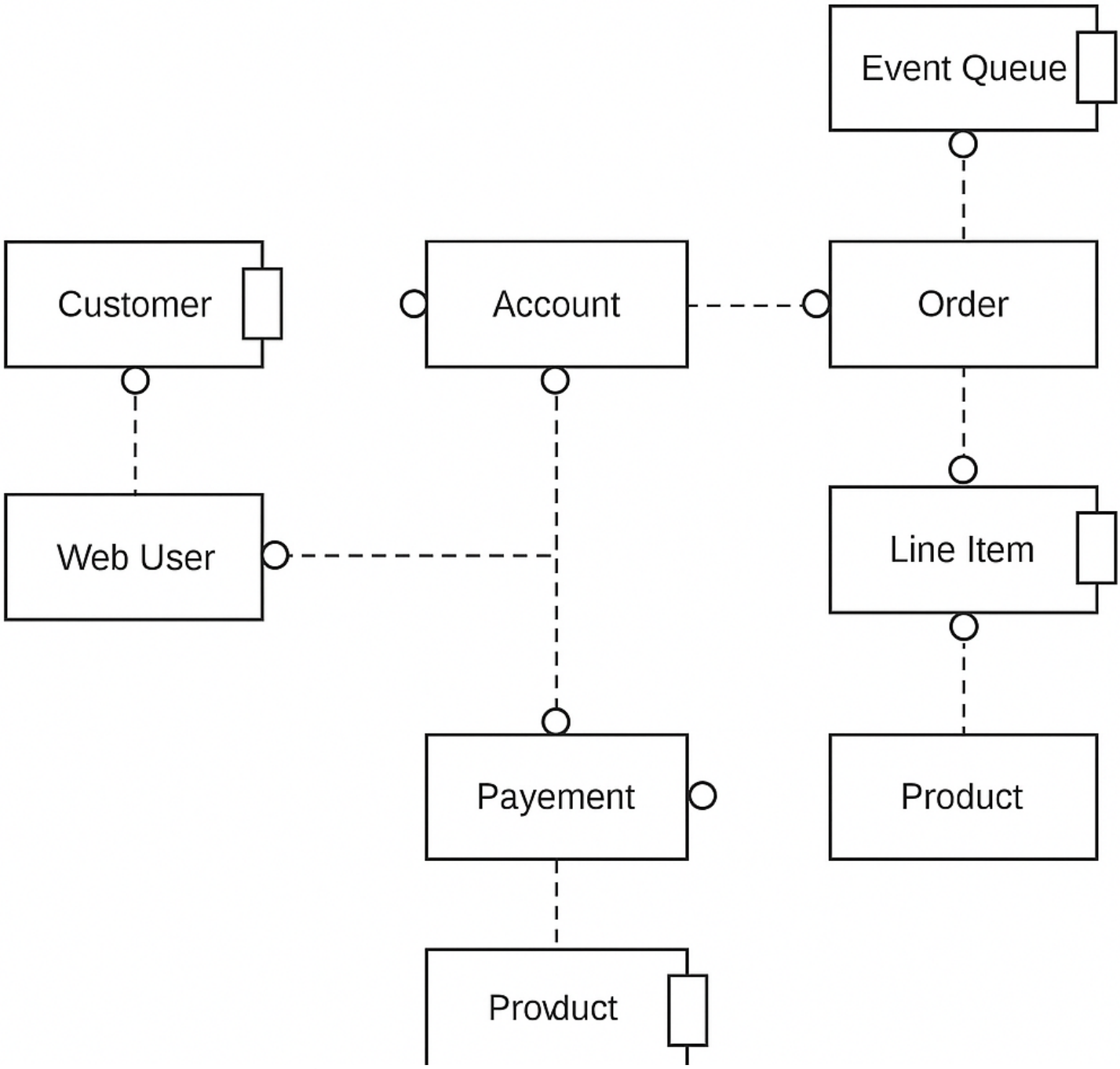
Q9.

Requirements:

```
Online Shopping

Each customer has unique id and is linked to exactly one account. Account owns shopping cart and
orders. Customer could register as a web user to be able to buy items online. Customer is not
required to be a web user because purchases could also be made by phone or by ordering from
catalogues. Web user has login name which also serves as unique id. Web user could be in several
states - new, active, temporary blocked, or banned, and be linked to a shopping cart. Shopping
cart belongs to account.
Account owns customer orders. Customer may have no orders. Customer orders are sorted and unique.
Each order could refer to several payments, possibly none. Every payment has unique id and is
related to exactly one account.
Each order has current order status. Both order and shopping cart have line items linked to a
specific product. Each line item is related to exactly one product. A product could be associated
to many line items or no item at all.
```

Component Diagram:



Requested Architectural Style:

Event-driven architecture: An event-driven architecture involves components which communicate by producing and consuming asynchronous events rather than direct function / API calls. Events are typically mediated through some central event broker / bus component that handle subscriptions and event publishing.

- There should be some central component representing the event broker / bus.
- All components should communicate exclusively with the central event bus component.

Q10. Using the requirements specification and architectural style specified above, rate the component diagram displayed above on the following criteria:

	Not fulfilled	Minimally fulfilled	Partially fulfilled	Mostly fulfilled	Completely fulfilled

Completeness: the diagram covers the content of all the requirements with a sufficient degree of detail to communicate with stakeholders	○	○	◉	○	○
Correctness: the diagram specifies behavior that is coherent and consistent with the requirements	◉	○	○	○	○
Adherence to the standard: the diagram is syntactically correct and semantically sound with respect to the standard for UML component diagrams	○	◉	○	○	○
Clarity / degree of understandability: the diagram is sufficiently clear and could be interpreted by stakeholders	◉	○	○	○	○
Terminological alignment with requirements: the terminology in the generated diagram aligns with the specified requirements	○	○	○	◉	○
Adherence to architectural style: The diagram satisfies the constraints typically indicative of the desired architectural style	◉	○	○	○	○

Q11. Please provide any other comments you might have about this component diagram:

Event queue not a player here. Hard to say how the system operates. Abandoned, unlabeled interfaces, overall bad.

Location Data

Location: (41.2884, -95.9972)

Source: GeoIP Estimation