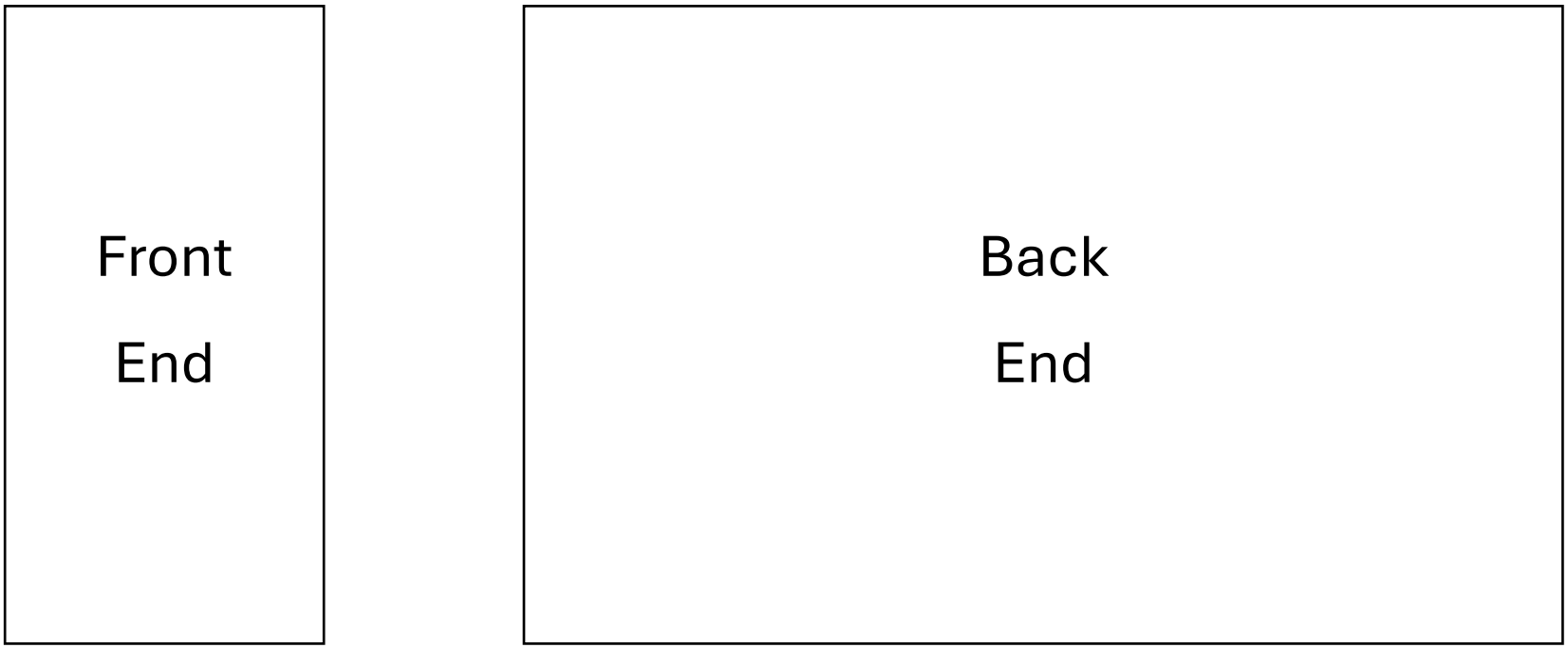
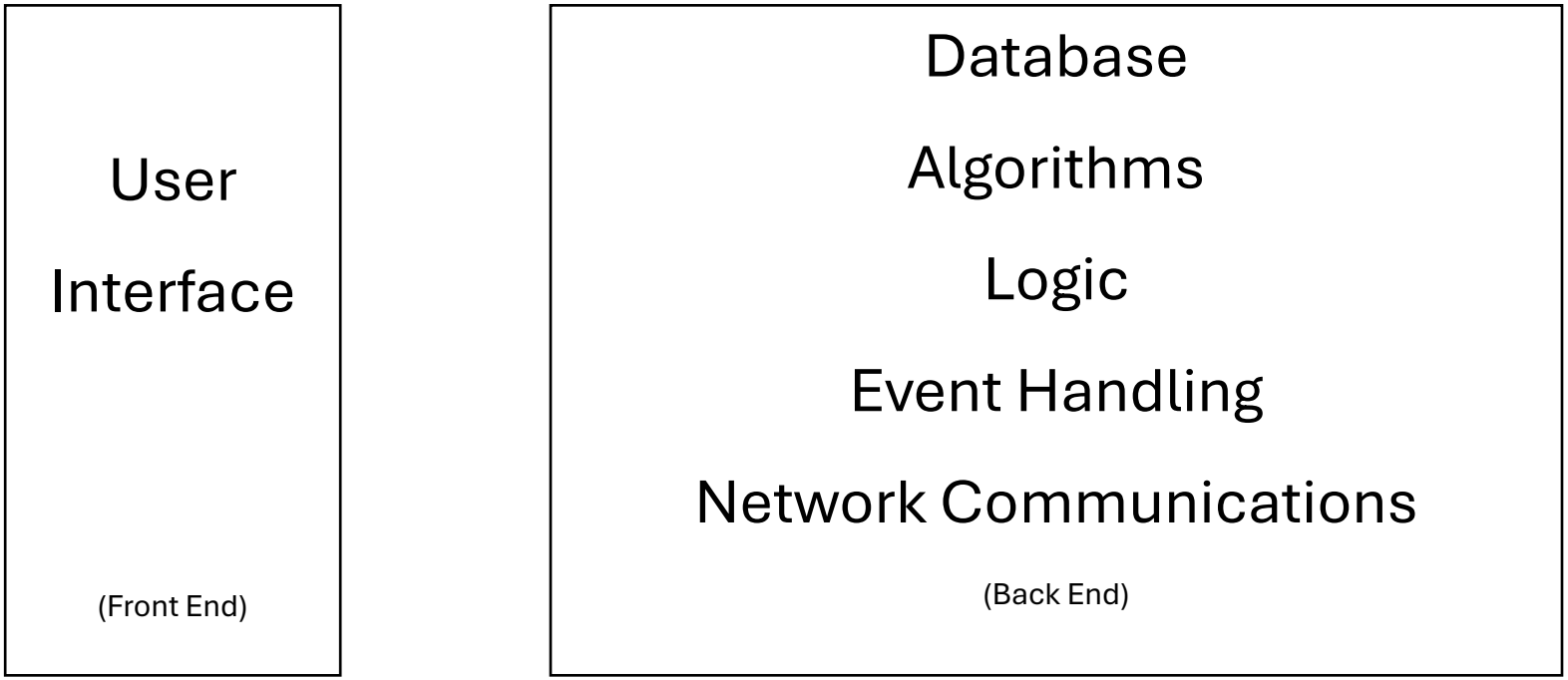


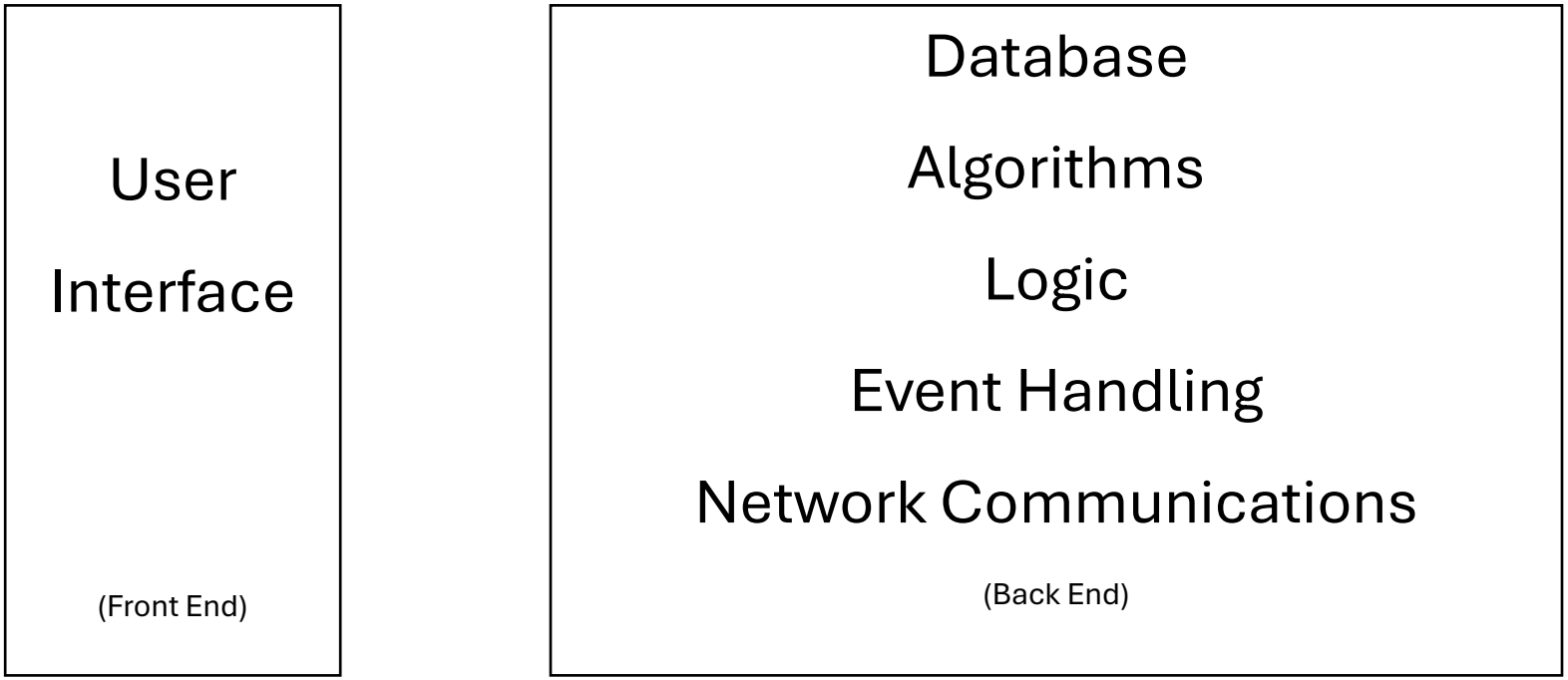
Generic Software Architecture

1. **Front end** – what the user sees. It is human-centric.
2. **Back end** – what the computer sees. It is machine-centric.



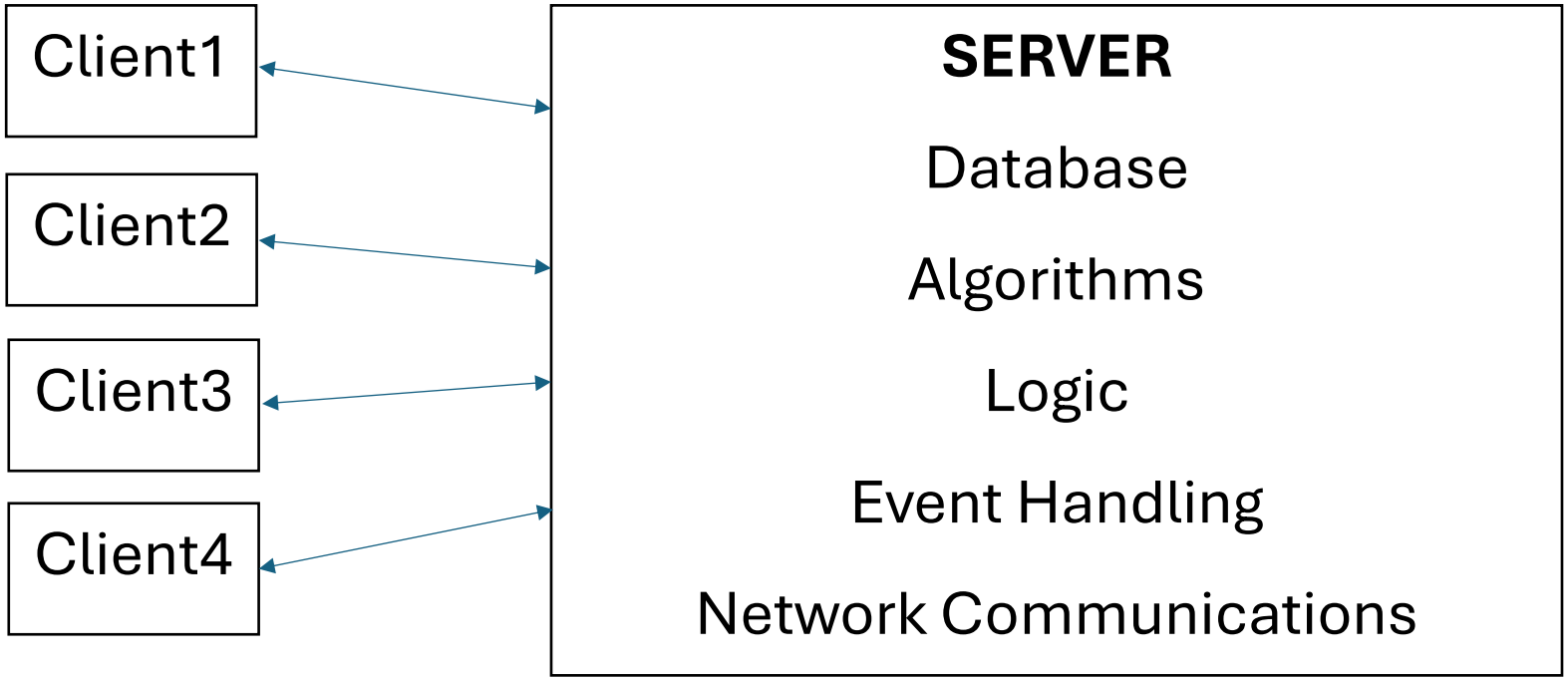
Generic Software Architecture – The Components

1. **User Interface, Customer based** – aesthetic, simple, easy to use for a non-technical person.
2. **User Interface, Engineering based** – basic, complex, easy to use for an engineer.



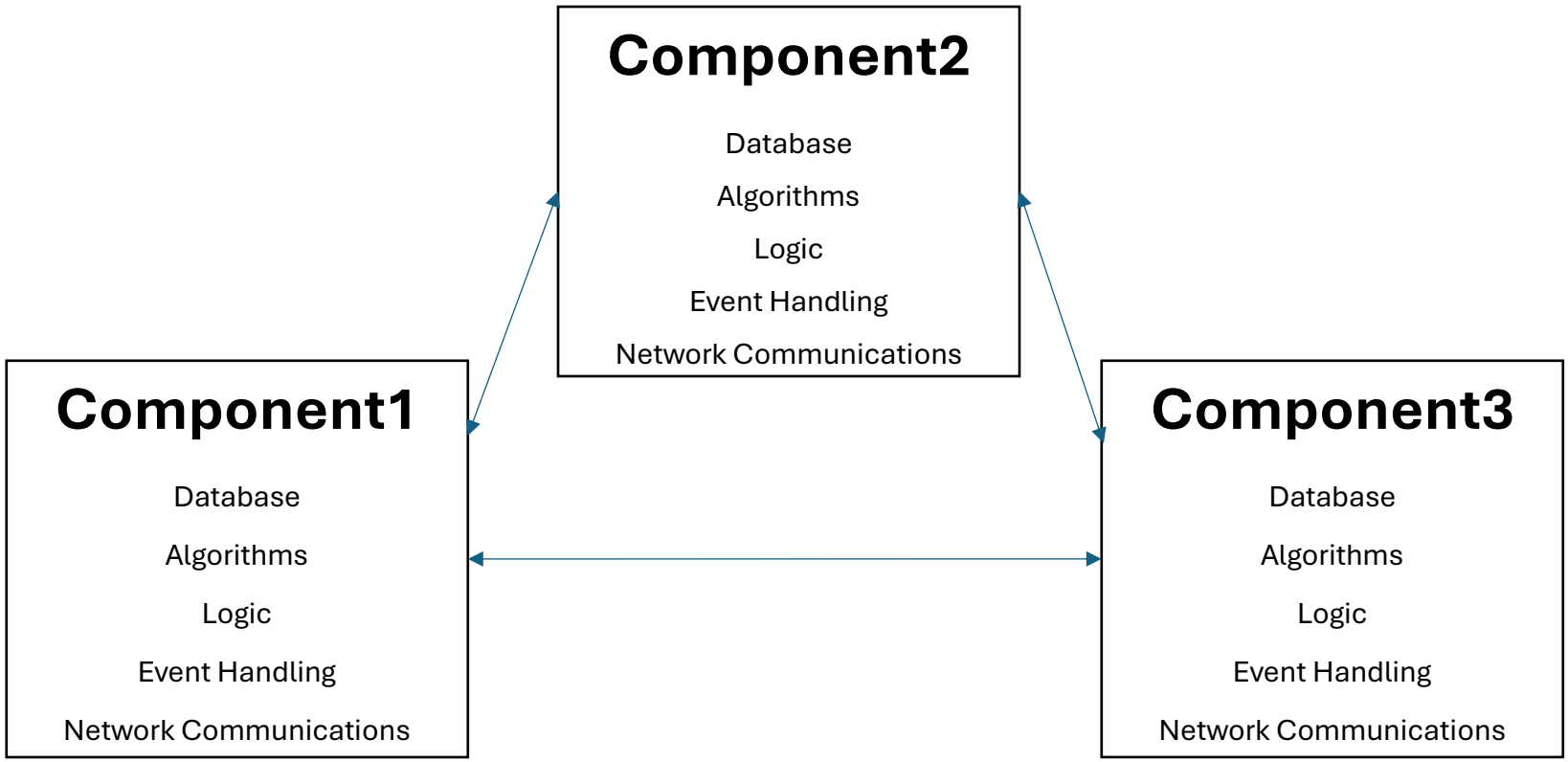
Generic Software Architecture – Testing

1. **Unit tests** – test the internal functioning of: the user interface, the database, algorithms, logic, event handling, and network communications. Usually done by software developers/engineers.
2. **Integration testing** – test the interfaces or interactions between the components. Usually done by the test department.
3. **System testing** – test everything as a whole. Usually done by the test department.
4. **Acceptance testing** – the customer accepts (or rejects) the product as a whole. Usually done by the customer.



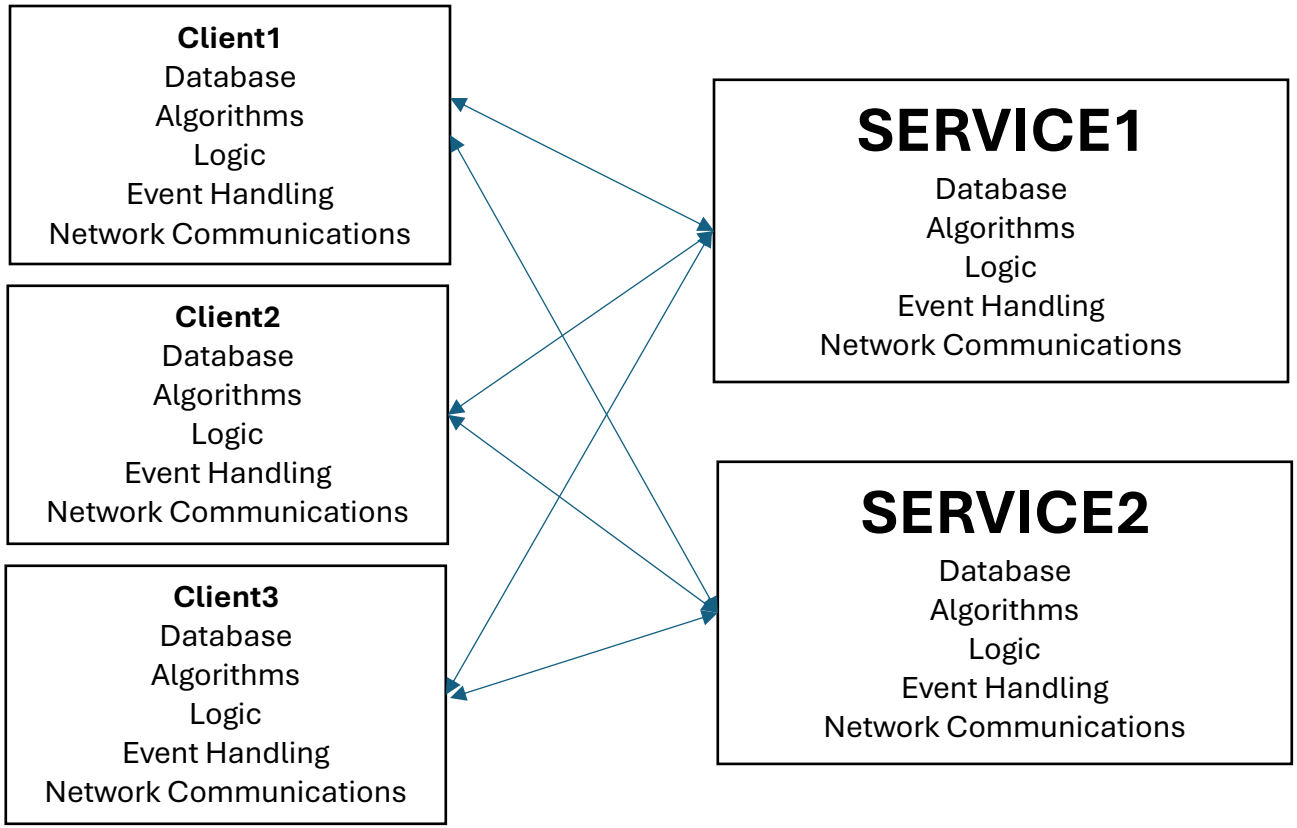
Client/Server Architecture

1. Active components on the clients: customer based user interface, network communications.
2. Active components on the server: database, algorithms, logic, event handling, network communications. May have an engineering based user interface.



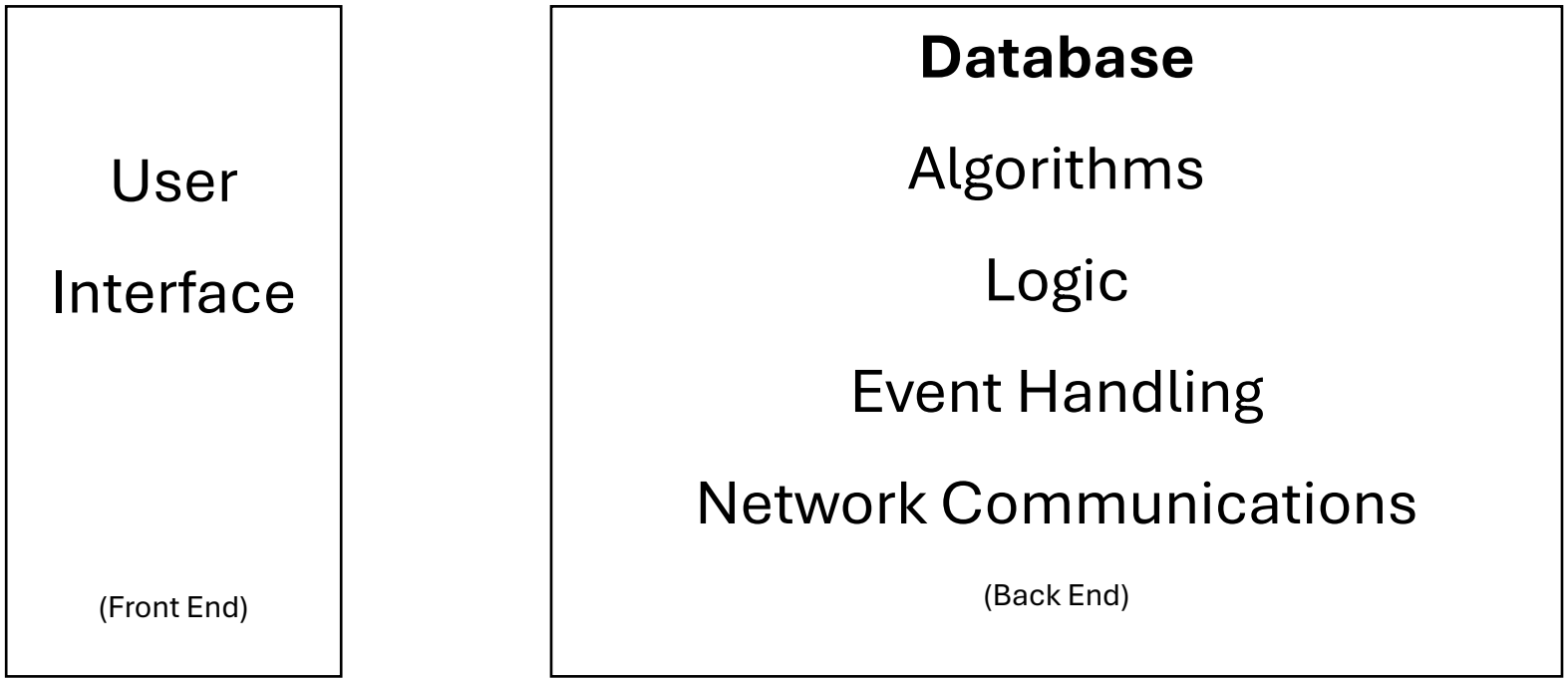
Component-Based Architecture

1. Active components on each Component: customer based user interface, some components of the back-end, network communications.
2. Each Component specializes in one or more components of the back-end.
3. The Components together can be seen as a single entity.



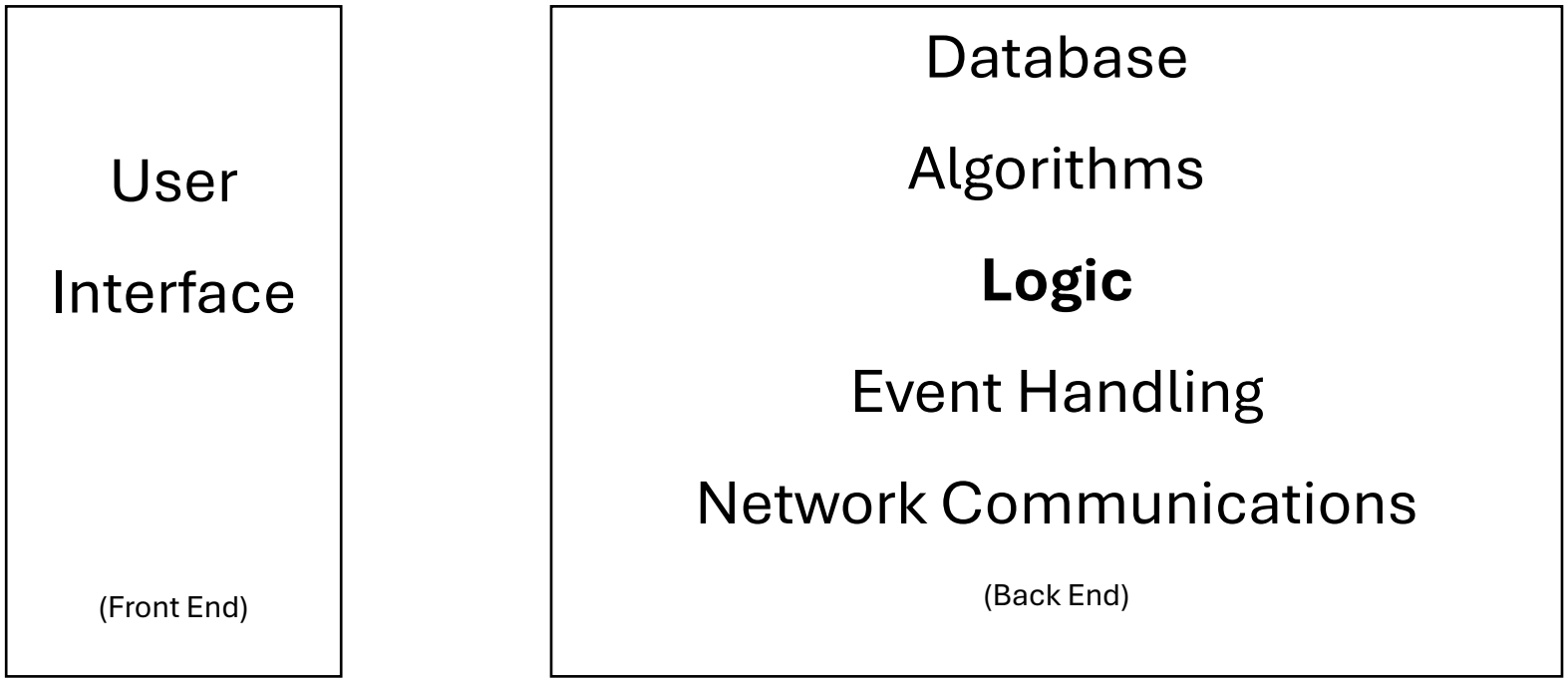
Service-Based Architecture

1. Active components on the clients: customer based user interface, some components of the back-end, network communications.
2. Active components on the services: engineering based user interface, some components of the back-end, network communications.
3. Each service specializes in one or more components of the back-end.



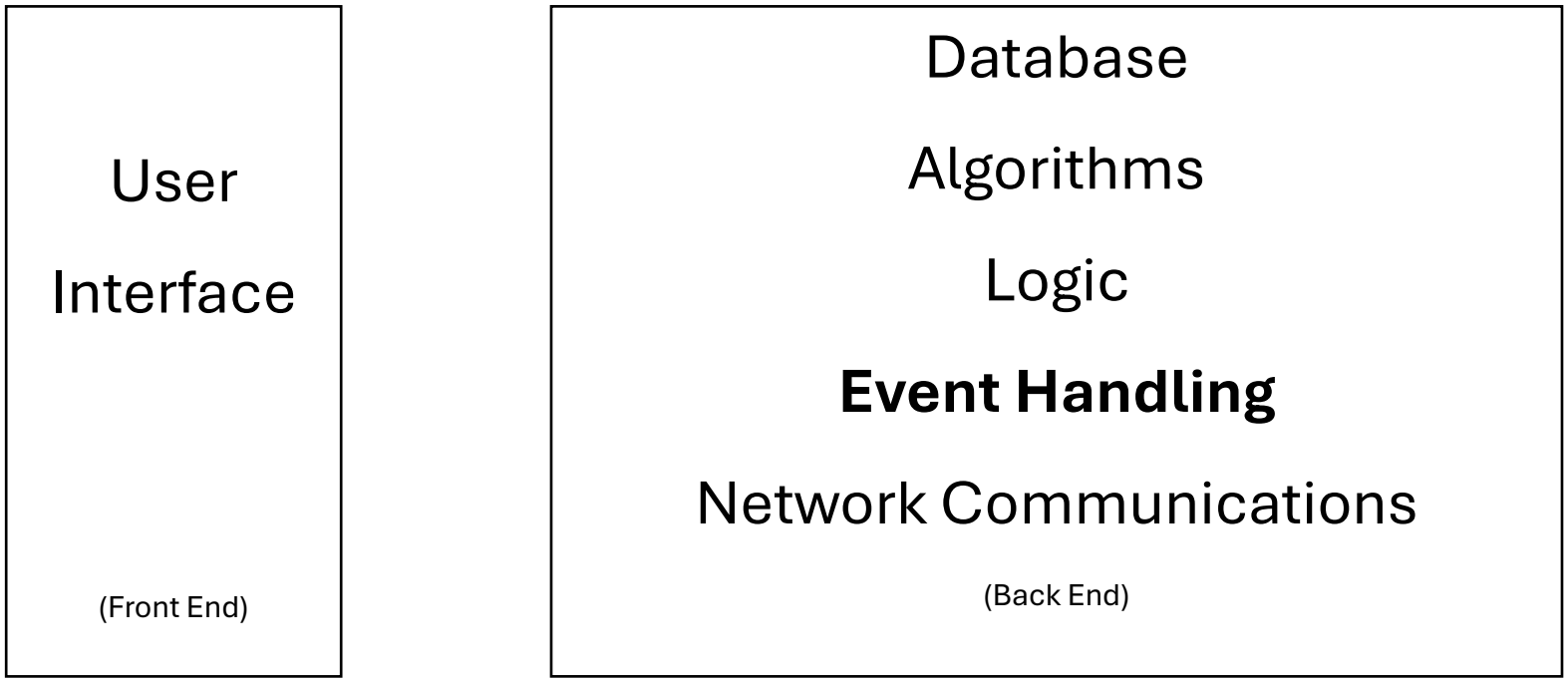
Data-Centric Architecture

1. The architecture focuses on data.
2. The data-centric architecture can be used in combination with any of the other architectures. Here we say the data-centric architecture is the main architecture, and other architectures are secondary architectures.



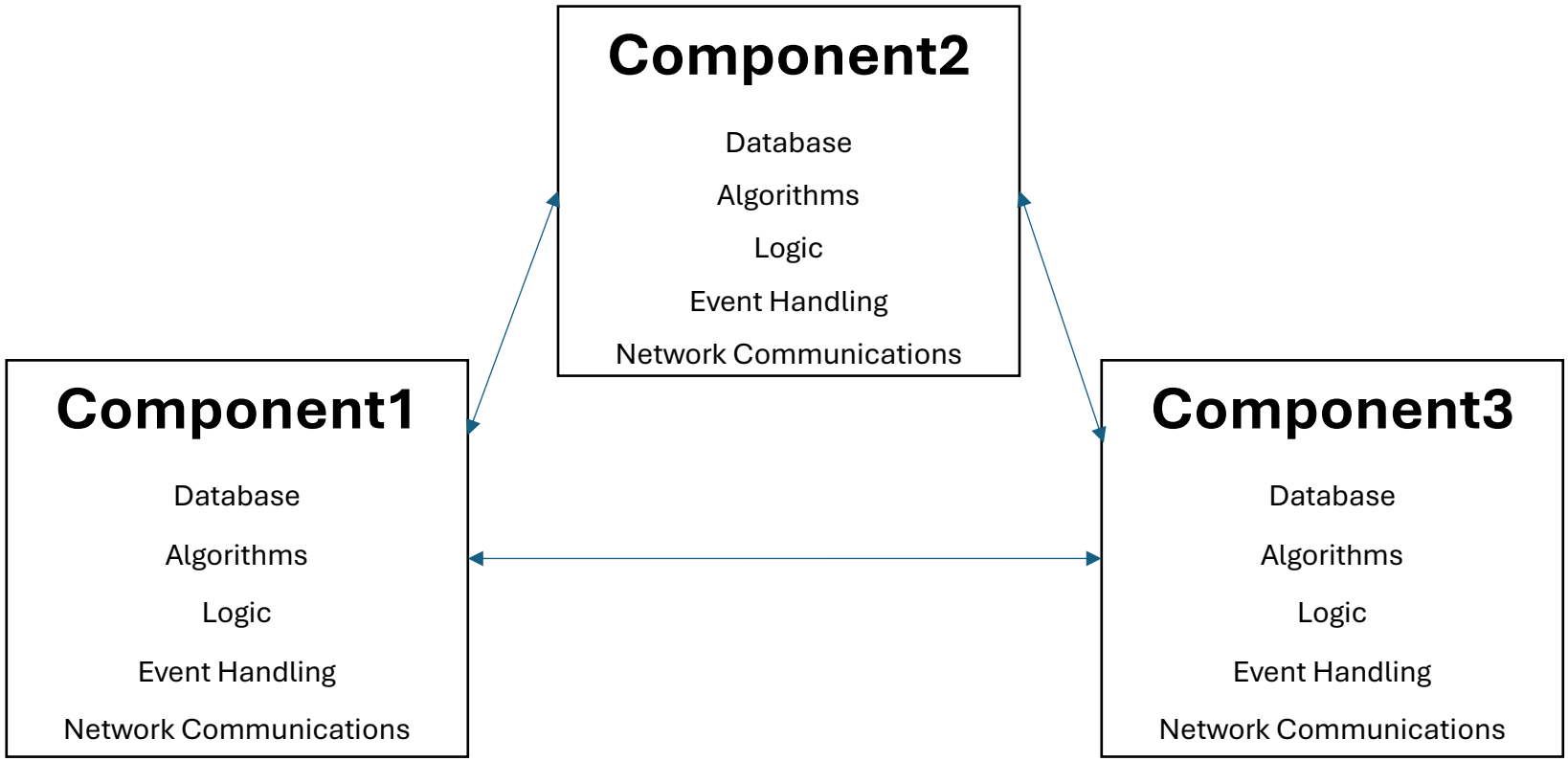
Rule-Based Architecture

1. The architecture focuses on logic. It mimics human intelligence. For AI, this is the most common type of architecture.
2. The rule-based architecture can be used in combination with any of the other architectures. Here we say the rule-based architecture is the main architecture, and other architectures are secondary architectures.



Event-Driven Architecture

1. The architecture focuses on event-handling.
2. The event-handling architecture can be used in combination with any of the other architectures. Here we say the event-handling architecture is the main architecture, and other architectures are secondary architectures.



Distributed Architecture

1. Similar to Component based architecture, except the Components are distributed over different computers in different locations, but acting as a single entity.
2. Each Component specializes in one or more parts of the back-end.
3. The database can be distributed, the algorithms and logic can be distributed, the event handling can be distributed.