1. System Architecture

The system uses a microservices-based architecture with a single-page frontend.

It consists of,

- React Frontend
- Design Service (Microservice)
- Quote Service (Microservice)
- (Future Implementation / Conceptual) Order & Payment Services
- (Future Implementation / Conceptual) External Storage

Since the defined business problem involves multiple independent responsibilities, clear separate data ownerships, and possibility of easier future expansion microservices based architecture is selected to be the most suitable system architecture.

Identified microservices are,

- Design Service (Microservice)
- Quote Service (Microservice)

Design Service

POST /designs → submit design

GET /designs → list designs

Quote Service

POST /quotes → submit quotes

GET /quotes/{designId}

- For demo purposes, in-memory storage is used. No database has been utilized.

Designer - Frontend

Design Title

Select Category

Description of the Design

Upload Design (Image/PDF)

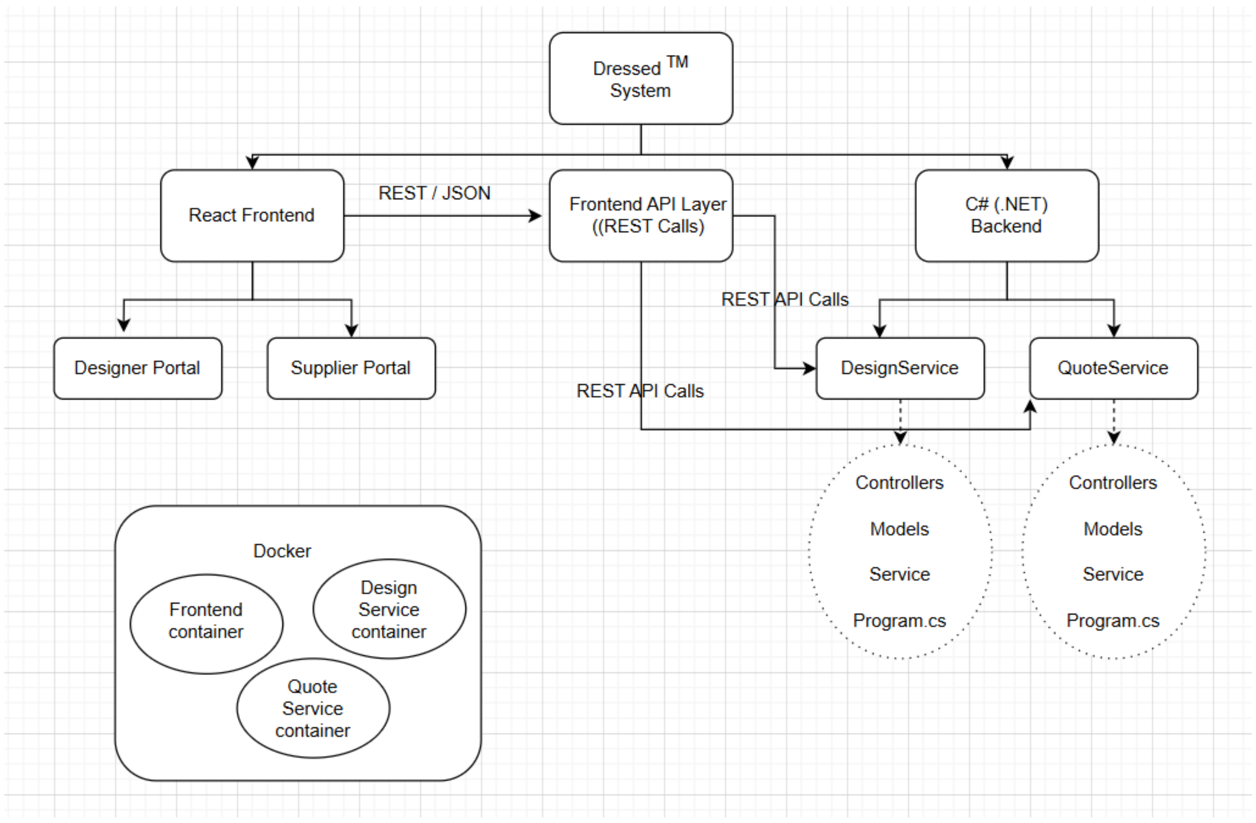
Supplier - Frontend

Filter by Category

Load Designs

Price

Message



5. System Architecture

Target Environment

The application is designed to be deployed in a cloud environment such as AWS, Azure, or GCP using Docker containers. For local development and demonstration, Docker Compose is used.

Deployment Approach

Each microservice (Design Service, Quote Service, Frontend) is containerized using Docker and orchestrated using Docker Compose. This ensures consistency across development and deployment environments.

Scalability Considerations

Since services are stateless, they can be horizontally scaled by running multiple container instances behind a load balancer. Individual services can scale independently based on system demand.

Reliability Considerations

Containerization allows services to be restarted automatically in case of failure. Future improvements may include health checks, centralized logging, and monitoring tools.