Application Development Architecture Document

1. Application Runtime Environment
The application is designed using a modern web stack for scalability, efficiency, and ease of development. The chosen technologies are:

- Programming Language: JavaScript (Node.js for backend, React.js for frontend)
- Backend Framework: Express.js (lightweight and efficient for RESTful API development)
- Frontend Framework: React.js (component-based UI framework for dynamic web applications)
- Runtime Environment: Node.js (server-side execution, non-blocking I/O for better performance)
- Middleware: Express.js (for handling API requests and responses)
- Operating System: Ubuntu 22.04 LTS (Linux-based, chosen for security and performance in cloud deployment)

Justification:
-Node.js enables high-performance, event-driven backend processes.
- Express.js simplifies routing and middleware implementation.
- React.js provides a reusable component-based structure for UI.
- Linux (Ubuntu) offers stability, cost-efficiency, and security advantages for deployment.

---

2. API Structure (RESTful)
The application follows the RESTful API design to ensure modularity and scalability. All API endpoints communicate using JSON for structured data exchange.

API Endpoints:
User Management
- `GET /users` → Retrieve all users
- `POST /users` → Add a new user
- `GET /users/:id` → Retrieve user by ID
- `PUT /users/:id` → Update user details
- `DELETE /users/:id` → Delete a user

Product Management (Example for e-commerce or inventory system)
- `GET /products` → Retrieve all products
- `POST /products` → Add a new product
- `GET /products/:id` → Retrieve product by ID
- `PUT /products/:id` → Update product details
- `DELETE /products/:id` → Delete a product

API Data Format (JSON Example):
User Object

```json
{
  "id": 1,
  "name": "Tom Hanks",
  "email": "Tomhanks@example.com",
  "role": "Admin"
}
```

Product Object
```json
{
  "id": 101,
  "name": "Laptop",
  "price": 1200.99,
  "category": "Electronics"
}
```

---

3. Port Configurations
To ensure proper communication between services, the following ports are designated:

Component	Port
Frontend (React.js)	3000
Backend (Express.js API)	5000
Database (MySQL on AWS RDS)	3306
HTTP (Redirected to HTTPS)	80
HTTPS (Secure API communication)	443

Justification:
- Port 3000: Used for local development of the React.js frontend.
- Port 5000: Used for Express.js backend API requests.
- Port 3306: Default MySQL port, restricted to backend access only.
- Port 80/443: Ensures secure API communication using HTTPS.

---

4. Diagrams

4.1 API Architecture Diagram
The API architecture outlines how the frontend, backend, and database interact.

(Diagram will be generated)

4.2 Network Architecture Diagram
The network architecture details AWS services, security groups, and port configurations.

(Diagram will be generated)