**System Design Document: Vortexify - A Builder**

**1. Project Overview**

**Vortexify: A Builder** is a modular DevOps automation platform designed to streamline code deployment from Git repositories to VMware environments. It features a unique three-module architecture: **Skin (UI Layer)**, **Brain (Backend Intelligence)**, and **Heart (Automation Core)**. This system enables users to log in, manage projects, automate builds, and deploy containers effortlessly.

**2. System Architecture**

**Modules Breakdown**

1. **Skin Module (UI Layer)**
   * Built using Laravel MVC Framework.
   * Handles user authentication, dashboard interface, and project management views.
   * Uses React (optional) for password reset and dynamic UI flows.
2. **Brain Module (Backend Layer)**
   * Built with Spring Boot (Java).
   * Responsible for user management, project metadata, deployment status, and API endpoints.
   * Stores user-to-deployment mapping, container info, and logs in a relational database (e.g., MySQL).
3. **Heart Module (Automation Layer)**
   * Python scripts handle:
     + Git repository cloning
     + Docker image building
     + Running shell scripts
     + Deploying images to VMware
   * Shell scripts are used to manage container lifecycle.
   * Future plan: Integrate performance logging using C++ for high-performance tracking.

**3. Deployment Workflow**

1. User logs in through the Laravel UI.
2. User submits a Git URL and selects configurations.
3. The backend (Brain) validates and records the deployment intent.
4. Brain sends task parameters to the Heart Module.
5. Heart clones the Git repo, builds the Docker image, and deploys it to VMware.
6. Brain updates deployment status and stores the container ID, user ID, and timestamps.
7. (Optional) Logs are generated for future C++ integration.

**4. Technologies Used**

* **Frontend:** Laravel, React (Optional)
* **Backend:** Spring Boot (Java)
* **Automation:** Python, Shell Scripts
* **Containerization:** Docker
* **Virtualization:** VMware
* **Database:** MySQL
* **Performance Logging (Future):** C++

**5. Features**

* Role-based access control and user management
* Automated deployment pipeline
* Git-based code ingestion
* Docker image creation
* VMware virtual machine deployment
* Modular, scalable architecture
* Future enhancements: Logging, load balancing, Kubernetes (when possible)

**6. User Roles & Interactions**

* **Admin:** Can manage all deployments, view logs, and modify user roles.
* **Developer/User:** Can submit Git repos, view their deployments, and receive feedback/status.

**7. Future Roadmap**

* C++ Integration for real-time container performance logging
* React-based dynamic password recovery flow
* Support for multi-VM deployment
* Optional Ansible for provisioning
* RESTful API exposure for external clients

**8. System Design Evaluation**

* **Scalability:** Modular architecture allows independent scaling of each module.
* **Security:** Token-based auth, database validation, and potential HTTPS.
* **Maintainability:** Code and function separation across modules.
* **Extensibility:** New deployment strategies (e.g., K8s, Ansible) can be easily integrated.

**9. Use Case & Relevance**

This project is ideal for organizations seeking an on-premise DevOps automation builder tailored for VMware infrastructures. It reflects industry-level architecture and is a powerful addition to any DevOps portfolio.

**10. Conclusion**

Vortexify is more than a builder — it’s a robust deployment platform combining automation, clean architecture, and future extensibility. With real-world applicability and a clear modular vision, it’s built to solve real deployment pain points in mid to large-scale environments.

*Prepared by: Rishav Raj* *Date: 7 April 2025*