1. Project Name: Vectoring Account Origination System (VAOS)

2. Criteria 1 - Partnership:

1. **Collaboration with stakeholders:** The transition from the legacy New Account Origination System (NAOS) to the innovative VAOS requires close collaboration between internal IT teams, business/users, vendors, and external teams. We anticipate fostering a cooperative environment, bridging the gap between technical and non-technical stakeholders.
2. **Team spirit:** With a shared vision to modernize our account application system, our team is driven to build a high-performing solution that will bring immense value to our company.
3. **Business satisfaction:** Implementing VAOS will address the performance and architectural issues faced by NAOS, undoubtedly enhancing user experience and garnering trust from the business side.

3. Criteria 2 - Proficiency:

1. **Project success rate:** By adopting a cutting-edge "data graph vectoring" design pattern, inspired by the SAGA pattern, and employing a microservices approach, we aim for a successful rollout and seamless integration with existing infrastructure. This flexibility allows teams to independently develop, test, and deploy components, reducing bottlenecks and ensuring quicker time-to-market.
2. **Innovation and skills:** VAOS, unlike its predecessor, will tap into innovative data graph vectoring. This design will reshape how data flows, ensuring high performance and offering advantages like scalability, distribution, and maintaining the ACID properties. It also provides a visual representation, aiding in debugging and monitoring.

4. Criteria 3 - Cost Optimization & Saving:

1. **Cost optimization support:** Transitioning from the legacy Java and IBM WebSphere system to VAOS is expected to reduce maintenance overheads. Given its microservices architecture, future upgrades or modifications to individual components won't necessitate a system-wide overhaul.
2. **Efficiency of project delivery:** With native coding and a focus on maximizing performance, VAOS promises efficient project delivery, reducing infrastructure costs in the long run, ensuring we get the best value for our investments.

5. Criteria 4 - Technology Modernization:

1. **Data Graph Vectoring:** This innovative approach offers a structured yet flexible way of processing and storing data. It ensures optimal data flow, minimizes redundancy, and allows for intuitive mapping and representation of data relations and flow.
2. **Transition from Monolithic to Microservices:** The shift from NAOS's monolithic architecture to VAOS's microservices-oriented design signifies a giant leap in

modernization. This allows for independent deployment of services, facilitating easier updates, debugging, and scalability.

3. **Containerization:** By utilizing container technologies such as Docker and orchestration tools like Kubernetes, VAOS ensures consistent deployments, isolates application dependencies, and provides scalability. This approach is in line with the modern best practices for deploying microservices.
4. **Enhanced Security:** VAOS will incorporate the latest in security technologies and practices. This includes the use of secure communication protocols, data encryption at rest and in transit, and modern identity and access management solutions to safeguard sensitive data and ensure unauthorized access prevention.
5. **Integration with Modern CI/CD Pipelines:** Continuous Integration and Continuous Deployment (CI/CD) pipelines will be utilized, allowing for seamless integration of code changes, automated testing, and swift deployments. This ensures a consistent and efficient development-to-deployment flow.
6. **Cloud Compatibility:** VAOS is designed with cloud architecture in mind. Whether opting for public, private, or hybrid cloud, VAOS can seamlessly integrate, providing benefits like auto-scaling, disaster recovery, and distributed data storage.
7. **Event-Driven Architecture:** Drawing inspiration from the SAGA pattern, VAOS will implement an event-driven model. This ensures real-time response, enhances the system's reactivity, and decouples service dependencies, further optimizing the system's performance.
8. **API Gateway & Service Mesh:** With multiple microservices in play, an API Gateway will manage external requests, acting as a single-entry point, ensuring optimized request routing, load balancing, and authentication. Service meshes like Istio may be implemented for enhanced intra-service communication, monitoring, and security.
9. **Decentralized Logging and Monitoring:** VAOS will utilize tools like Elasticsearch, Logstash, and Kibana (ELK Stack) or Grafana and Prometheus for real-time monitoring, analytics, and visualization. This aids in quickly identifying, diagnosing, and rectifying issues.
10. **Decoupled Data Storage:** Instead of a one-size-fits-all database, VAOS will implement a polyglot persistence approach. Depending on the service's need, it might use relational databases, NoSQL stores, or graph databases, optimizing data retrieval and storage efficiencies.
11. **Native Executable**: The utilization of native coding techniques in VAOS ensures the software is directly compiled and optimized for specific platforms. This provides a significant boost in performance and responsiveness. By tapping into platform-specific capabilities, VAOS can ensure the best user experience and interface compatibility, further leveraging the underlying hardware more effectively.

6. Submitted by: Narupol Hongthai