Testability – Potential Remedies and Improvements

Observability • Logging in each layer of the architecture and between components • Controllable levels of logging by request to gather further information on consumer requests • Aggregated logging to bring together logs from various layers and interfaces to show holistic pictures of systems events • Create the ability to replay traffic to observe events in the architecture • Monitoring which combines diagnostics (CPU usages for example) and application information	**Controllability** • Feature flagging techniques to control exposure of new technology or functionality • Trial management to release new technology or functionality to a limited number of consumers • Automated deployment of the IDResolver application to be able to deploy new versions on demand • Blue Green deployments to switch between new and old versions to protect current and new functionality • Create test environments which mirror the production environment to a high degree
Decomposability • Employing a microservice architecture to decouple critical components • Create circuit breakers between layers to handle persistent error conditions between internal and external services. • Add queueing technology such as Kafka to manage high load scenarios or to queue infrequent operations to be processed at a different time. • Create mocks for external services to isolate parts of the architecture for testing and issue diagnosis	**Simplicity** • Automated API documentation tooling such as Swagger • Developers and testers supporting the system in Production • Policies to pay back and discourage build-up of technical debt such as test-driven development • Pair and mob programming to promote knowledge sharing between developers and operations people. • Adopt Lightweight Architecture Records in source control to track changes to the architecture