VALERIAN TECHNOLOGY

NOVEMBER 2024

Technical Due Diligence Report

PRESENTED TO

MockCo

PRESENTED BY

Valerian Technology

Table of Contents

Executive Summary

MockCo Technologies

The purpose of this document is to **review the code and technology** powering MockCo Technologies. This summary was prepared after virtual meetings with the MockCo team in October 2024, supplemented by email and phone discussions, as well as a review of source code and documentation provided in the data room. The focus of this document is on development practices, code security, infrastructure, and technology architecture, providing a high-level overview along with targeted analysis. Code was evaluated based on the following criteria: **maintainability, adherence to coding standards, complexity, and quality**.

While every effort was made to conduct an in-depth review of all facets of the technology, this document is intended as a broad overview rather than a comprehensive assessment. It highlights the main technical and operational aspects of MockCo's platform and the resources directly involved in its development and management.

MockCo Technologies is a mid-stage SaaS provider in the logistics industry, offering a modular and scalable platform aimed at enhancing operational efficiency and real-time data tracking. Although the company has made strides in product-market fit and platform modularity, several areas require attention to support long-term scalability and stability. These recommendations primarily focus on improving security protocols, refining onboarding practices, managing open-source license compliance, and reducing code complexity.

A total of **six red flags** and **six yellow flags** were identified concerning platform architecture, security, infrastructure, and development practices.

Category	Score	Summary
IT and Infrastructure	75%	AWS and Kubernetes provide scalability, but storage and compute optimizations are needed for growing demands.
Development Practices and Architecture	60%	Modular architecture and agile practices support flexibility, but technical debt and LLM code variability impact maintainability.
People and Process	55%	Agile structure is flexible, but skill gaps in AI, security, and core technical roles create dependencies on key contributors.
Overall Rating	63%	**MockCo has a strong foundation with scalable infrastructure but needs improvements in security, data governance, and team skills to support sustainable growth.**

Red Flags

Data Quality and Governance for AI

Description: AI requires high-quality, structured data for reliable performance. Current data practices, including inconsistent labeling and limited governance, could reduce AI effectiveness and introduce compliance risks, particularly when processing sensitive client data.

Remediation Options: Establish robust data governance policies, standardize data labeling, and implement automated data pipelines to ensure data accuracy and compliance.

Risk: High

Time to Remediate: 3-4 months, 2 FTE

Code Quality and Security for LLM-Generated Code

Description: Approximately 30% of MockCo's codebase is generated by large language models (LLMs). While LLM-generated code accelerates development, it may introduce inconsistencies, security vulnerabilities, or technical debt if not properly managed.

Remediation Options: Implement automated code review and verification tools to detect potential issues in LLM-generated code early and ensure alignment with security standards.

Risk: High

Time to Remediate: Ongoing, with initial setup over 1-2 months, 1 FTE

License Issues

Description: Open-source software scans revealed the usage of problematic licenses within the core codebase, specifically AGPL-licensed libraries that may require MockCo to disclose proprietary code.

Remediation Options: Replace, remediate, or purchase a commercial license as applicable.

Risk: High

Time to Remediate: 3 weeks, 2 FTE

High Number of Security Issues

Description: Security scans identified over 200 vulnerabilities, with 15 critical issues stemming from unpatched dependencies and configuration errors.

Remediation Options: Patch affected libraries, conduct configuration review, and establish regular vulnerability scanning.

Risk: High

Time to Remediate: 6 weeks, 3 FTE

Leaked Credentials

Description: Active API keys and credentials were found within the repository, creating a high-risk exposure to unauthorized access.

Remediation Options: Remove all hardcoded credentials, implement secure vault storage, and enforce secure credential handling policies.

Risk: High

Time to Remediate: 1 week, 1 FTE

No Field-Level Encryption

Description: Sensitive data fields lack field-level encryption, increasing exposure risk for specific data categories.

Remediation Options: Implement field-level encryption for sensitive data fields and update access controls.

Risk: High

Time to Remediate: 4 weeks, 2 FTE

Yellow Flags

Scalability of Infrastructure for AI Workloads

Description: AI models typically require substantial computational power and storage, which may strain MockCo's current infrastructure. While AWS and Kubernetes offer flexibility, the platform may need enhancements, such as GPU-enabled instances, to effectively support AI processing.

Remediation Options: Evaluate the infrastructure requirements for AI, add GPU resources, and consider optimizing Kubernetes clusters to support machine learning workloads.

Risk: Medium

Time to Remediate: 3-6 months, 2 FTE

Legacy Code and Compatibility with AI Tools

Description: MockCo's reliance on legacy components may limit compatibility with modern AI tools and hinder seamless AI integration. Legacy dependencies may require updates or compatibility solutions to ensure alignment with new AI workflows.

Remediation Options: Assess legacy code areas, prioritize updates, and implement compatibility solutions to enable smooth AI adoption and ensure scalability.

Risk: Medium

Time to Remediate: 4-6 months, 2-3 FTE

Dependency on Small Number of Contributors

Description: MockCo relies heavily on three senior developers for core modules, creating a high dependency risk.

Remediation Options: Develop and cross-train other team members to mitigate risk and enhance resilience.

Risk: Medium

Time to Remediate: Ongoing, with initial sessions over 6 weeks, 2 FTE

Deprecated Language Use

Description: Core services still rely on a deprecated version of Python, introducing security vulnerabilities and compatibility issues with newer libraries.

Remediation Options: Upgrade to a supported version of Python to prevent potential disruptions and reduce security risks.

Risk: Medium

Time to Remediate: 2 weeks, 2 FTE

Drop-off in Developer Productivity

Description: Developer productivity has declined by 30% over 12 months, potentially due to technical debt and inefficient onboarding practices.

Remediation Options: Streamline onboarding, reduce technical debt, and introduce productivity tracking tools.

Risk: Medium

Time to Remediate: 2 months, 3 FTE

Onboarding Time

Description: It currently takes MockCo 7 months and many manual processes to onboard a new customer using the existing technology stack.

Remediation Options: Automate onboarding workflows using RPA or software that can dynamically handle workflows.

Risk: Medium

Time to Remediate: 6 weeks, 2 FTE

IT and Infrastructure

Overview

MockCo's infrastructure is primarily hosted on AWS, using a mix of EC2 instances, S3 storage, and RDS databases. While the majority of services are containerized and deployed within Kubernetes clusters, we observed that some legacy components remain on standalone EC2 instances, lacking modern auto-scaling configurations. This setup may hinder MockCo's ability to handle high-traffic events seamlessly and incurs higher operational costs compared to fully optimized cloud-native solutions. Transitioning these legacy components into the containerized Kubernetes environment would improve both performance and cost efficiency. Additionally, adopting Infrastructure as Code (IaC) tools, such as Terraform or CloudFormation, could further enhance scalability and control.

Platform Security

The current security framework includes basic firewalls, SSL encryption, and AWS IAM roles, but it lacks a robust Identity and Access Management (IAM) strategy. Critical infrastructure has no multi-factor authentication (MFA) enforced, which leaves it vulnerable to unauthorized access. Additionally, no centralized identity provider is in place for role-based access control across services, increasing the risk of over-permissioned accounts. Regular security audits and an improved IAM framework are recommended, with specific focus on enforcing MFA, least privilege principles, and adopting a centralized identity provider such as AWS SSO or Okta for improved access management. A Web Application Firewall (WAF) could also provide an added layer of protection, particularly against application-layer attacks.

Failover and Recovery

The DR plan currently relies on nightly backups, which may result in significant data loss if recovery is required during peak transaction periods. While most AWS regions and availability zones are utilized, no multi-region redundancy is established, leaving the infrastructure susceptible to region-specific outages. To ensure business continuity, implementing a multi-region failover capability with more frequent data backups would be advisable. Additionally, regular disaster recovery drills should be conducted to validate the effectiveness of the recovery procedures and to adjust the DR plan based on the outcomes of these tests.

Uptime and Reliability

Analysis of historical performance data shows an average uptime of 98.1% over the past year, with recurrent downtime during peak usage periods. This downtime is primarily due to unoptimized load balancing and insufficient autoscaling policies. To enhance uptime and reliability, MockCo would benefit from implementing advanced load balancing strategies, such as an application load balancer paired with a global CDN like CloudFront, to optimize content delivery and reduce latency. Further, the autoscaling policies should be adjusted based on predictive metrics to proactively handle traffic spikes, ultimately improving user experience during peak times.

Monitoring and Telemetry

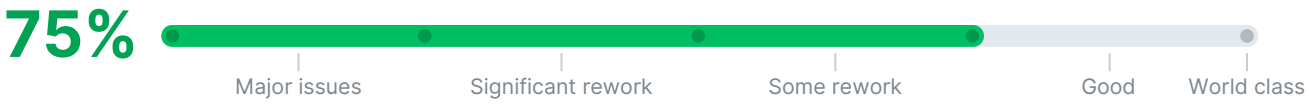
MockCo currently only performs application level logging, reducing their ability to identify root causes of issues and delay response times for incidents. Implementing a more sophisticated observability stack with tools like Prometheus for metric collection and Grafana for visualization would significantly enhance insight into the infrastructure's health. Additionally, integrating a centralized logging solution like ELK (Elasticsearch, Logstash, and Kibana) would enable MockCo to conduct rapid log analysis during incidents. Expanding telemetry to include user behavior and performance metrics would allow for more proactive incident management and improved overall platform stability.

Scaling and Cost

Current infrastructure costs at MockCo are approximately 25% higher than industry benchmarks, largely due to underutilized resources in the cloud environment and reliance on outdated instance types. Scaling challenges are also evident, with latency spikes observed under high load conditions. To address these issues, a cost and performance optimization review is recommended. By right-sizing instances, transitioning to spot instances where applicable, and leveraging reserved instances for predictable workloads, MockCo could achieve substantial cost savings while enhancing scalability. Furthermore, a detailed load test should be conducted to identify and address specific scaling bottlenecks, ensuring that the platform can handle anticipated growth without compromising performance.

KEY TAKEAWAYS

MockCo's AWS and Kubernetes-based infrastructure provides a solid foundation for scaling general workloads, but recurring peak-time slowdowns suggest that infrastructure could benefit from additional resources and optimization. Scaling storage and compute resources will help handle increased data demands and reduce latency, especially as user growth continues.



Recommendations

- **Evaluate infrastructure capacity** to ensure sufficient resources are available for future demands, prioritizing upgrades to prevent bottlenecks.
- Consider **adding reserved and spot instances within AWS** to support cost-effective scalability, focusing on predictable workloads and peak usage needs.
- **Implement a tiered storage system** with Amazon S3 for archival storage and Amazon RDS or Redshift for high-speed data access, enabling efficient data management across storage types.
- **Optimize Kubernetes clusters** to ensure efficient load distribution and minimal resource waste as workloads increase.

Development Practices and Architecture

Overview

MockCo's development practices are built around a modular architecture and agile methodologies, allowing for efficient feature deployment and adaptability to client needs. The modular design provides flexibility and supports scalability; however, challenges with code complexity, technical debt, and inconsistent documentation impact maintainability and operational efficiency. Additionally, MockCo's increased reliance on LLM-generated code (approximately 30% of the codebase) has introduced variability and potential security risks, underscoring the need for standardized coding and documentation practices. Ensuring consistency and quality across both manually written and machine-generated code will be essential for reducing onboarding times and supporting long-term scalability. This section assesses MockCo's current development practices, code quality, and team workflows, providing targeted recommendations to enhance stability, security, and productivity.

Code Complexity and Quality

The codebase is highly complex, with several interdependent modules that lack clear documentation. This complexity has introduced challenges in code maintainability and debugging, often leading to longer issue resolution times. While modularity aids feature development, the lack of a consistent structure in some areas results in fragmented workflows and increases the likelihood of bugs.

Lines of Code

Code size is measured by Non-Blank, Non-Comment (NBNC) lines of code. Blank lines, machine-generated code, and comments are removed to determine the overall active codebase. This analysis provides a comparison across languages used in the codebase, giving insight into code structure and complexity.

Language	Files	Blank	Comment	Code
Python	3,652	80,234	150,912	860,432
JavaScript	1,728	65,487	210,391	540,812
React Native	913	42,013	105,478	320,546
TypeScript	473	21,432	58,237	110,372
HTML	294	12,784	33,512	89,214
CSS	384	9,132	28,019	47,123
SUM	**7,444**	**231,082**	**586,549**	**1,968,499**

The total number of NBNC lines in MockCo's codebase is approximately 1,968,499 lines. The calculation is based on output from cloc, providing an understanding of code volume across languages and offering a benchmark for maintainability assessments.

Code Security

Basic security practices, such as linting and minimal code review, are in place but lack structure. There is no formal secure coding standard, and critical security practices such as dependency scanning are missing, which has led to the accumulation of unpatched vulnerabilities. Introducing a structured code review process focused on security and a dependency scanning tool would significantly improve code security.

Platforms and Languages

The platform is built primarily on Python for backend services and JavaScript with React for frontend and mobile development. While these languages are suitable for rapid development, some portions of the codebase rely on deprecated Python 2 libraries, which pose compatibility and security risks. Transitioning these components to the latest Python version is recommended.

Mobile Applications

The mobile applications use a hybrid approach, leveraging React Native for cross-platform compatibility. This approach supports streamlined updates but has led to some performance issues on Android devices due to optimization challenges within React Native. Testing and optimizing for device-specific performance would improve user experience.

Database

MockCo utilizes AWS RDS for its database solution, primarily structured as PostgreSQL. Backups are managed automatically, though no encryption is enabled for data at rest, which could expose sensitive customer data. Enabling encryption and implementing regular database audits are recommended for data security and compliance.

Deployed Code Versions

Due to unique requirements from numerous customers, MockCo must support 7 different versions of the application all hosted in separate environments and all updated independently from different branches in the source code baseline. As possible, MockCo should try to consolidate these branches by negotiating with customers or streamlining their main baseline to support all these features with configuration.

Open Source Licensing

Open-source license usage was generated from a FOSSA scan across all repositories. The table below highlights potential compliance risks identified within the open-source libraries.

License	Affected Projects
GPL-3.0	Core Backend, Analytics Module
AGPL-3.0	User Authentication Service
LGPL-2.1	API Gateway
MPL-2.0	Frontend Framework, Reporting Service
BSD-3-Clause (with restriction)	Mobile Application, Database Integration

Identified licenses, particularly copyleft ones like GPL-3.0 and AGPL-3.0, may pose compliance risks if not managed correctly. Recommendations include reviewing these licenses and replacing or purchasing commercial licenses where necessary to mitigate legal exposure.

Metrics

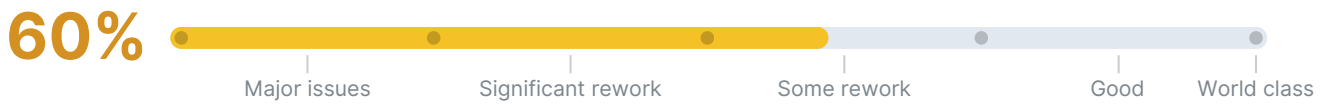
The following metrics provide a breakdown of code complexity, technical debt, and identified security hotspots in core components. Complexity metrics, in particular, help to assess the maintainability of each component.

Component	Complexity	Tech Debt	Security Hotspots
UserService	160,982	140 days	15
DataProcessing	175,263	165 days	20
Analytics	182,907	170 days	18
NotificationService	158,492	135 days	14
Backend	162,088	145 days	22
Frontend	188,104	180 days	30
API	N/A	130 days	N/A

The high complexity and accumulated tech debt in certain modules, such as AnalyticsEngine and WebFrontend, suggest potential bottlenecks in future feature development and increase the risk of system instability.

KEY TAKEAWAYS

MockCo's modular architecture enables flexibility and rapid feature deployment. However, technical debt, inconsistent documentation, and variability in LLM-generated code (30% of the codebase) impact maintainability. These factors introduce complexity, potential security risks, and hinder onboarding. Improving code quality and standardization will strengthen scalability and stability.



Recommendations

- **Establish a structured refactoring plan** focused on reducing technical debt incrementally, starting with high-complexity modules. Schedule regular code review cycles to address parts of the codebase that hinder maintainability.
- **Implement automated code review tools** and static analysis for LLM-generated code to detect security vulnerabilities, coding inconsistencies, and adherence to quality standards before merging into the main branch.
- **Standardize documentation practices** by assigning responsibility to specific team members for each module, ensuring documentation is comprehensive, up-to-date, and accessible for new developers.
- **Create a coding guideline** that applies to both human- and machine-generated code, covering best practices for naming conventions, code structure, and security protocols. This will help unify code quality across contributors.
- **Hold regular training sessions** to reinforce coding standards, building a culture of consistent code quality and maintainability across the team.

AI Readiness and Integration Potential

Overview

MockCo is well-positioned to incorporate AI, but several foundational improvements will enhance its readiness and potential. Ensuring scalable infrastructure is key, particularly by adding GPU-enabled instances and tiered data storage to support resource-intensive AI workloads. Reliable data quality and governance practices are also essential, as they directly impact AI model performance, accuracy, and compliance. Additionally, MockCo's reliance on LLM-generated code necessitates rigorous quality control to maintain consistency and security across the platform. Addressing these areas will enable MockCo to deploy effective AI capabilities that are both resilient and adaptable to future needs, helping the company stay competitive and meet evolving client demands.

Current Capabilities and Infrastructure Suitability

MockCo's existing data lake provides the necessary data to provide to Machine Learning models for training and utilization for higher level business intelligence activities. To fully optimize for AI, MockCo could consider incorporating GPU-enabled instances and exploring frameworks like TensorFlow or Kubeflow. These additions would allow MockCo to manage the entire AI model lifecycle, from data ingestion to deployment, seamlessly within the existing infrastructure. Given that approximately 30% of MockCo's codebase is generated by large language models (LLMs), implementing best practices for automated code validation would help ensure consistency, security, and alignment with project standards. Additionally, AI requires substantial data storage and processing capacity; tiered data storage solutions like Amazon S3 for archival purposes and Amazon Redshift for real-time analysis would provide scalable and cost-effective options, allowing AI models to access large datasets efficiently for training and improving model accuracy.

Potential AI Applications

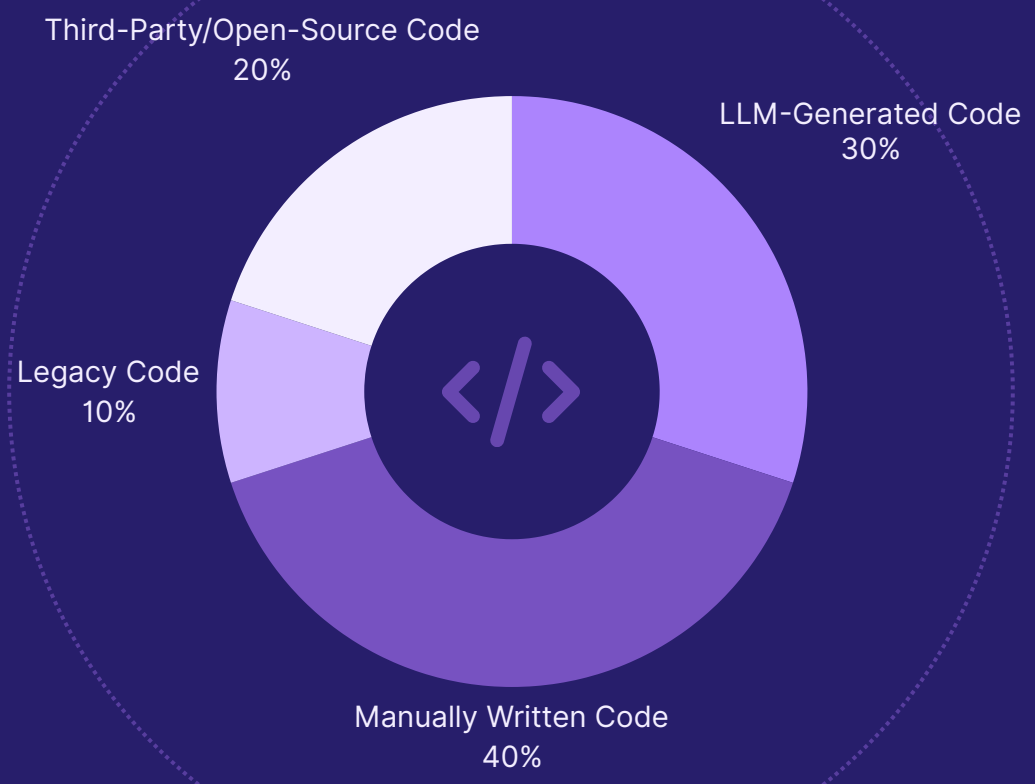
AI can drive transformative improvements for MockCo in logistics, customer support, and security. For example, MockCo could use predictive maintenance models internally to identify potential infrastructure issues. Analyzing historical performance data, these models could help prevent downtime by proactively resolving system challenges. AI-driven automation, such as virtual assistants, can also streamline customer support, handling repetitive inquiries, resolving common issues, and escalating complex cases to human agents when needed. This approach would enhance customer experience, reduce response times, and lower operational costs. In logistics, AI could support MockCo's clients by analyzing supply chain data to identify inefficiencies, optimize routing, and suggest resource adjustments, thereby reducing waste and improving service delivery. AI-enabled anomaly detection offers another key application, particularly for real-time threat detection and response. By automatically identifying unusual access patterns, AI could trigger security protocols, such as restricting access to sensitive data or alerting administrators, adding a proactive layer of protection against data breaches.

Challenges and Considerations

MockCo's data practices present challenges, as inconsistent labeling and limited data governance may reduce the effectiveness of AI models. Ensuring compliance with data privacy regulations will be crucial, particularly as AI applications process sensitive customer information. Additionally, AI integration could require addressing existing technical debt, which may otherwise hinder a seamless adoption process. MockCo may also need to expand its team's expertise, either by hiring or upskilling in areas like data science and machine learning to effectively manage AI projects.

A further consideration is the reliance on large language models (LLMs) for code generation, which currently accounts for approximately 30% of MockCo's codebase. While LLMs can accelerate development, the generated code may introduce consistency or security issues that require additional oversight. The pie chart below illustrates the proportion of LLM-generated code versus manually written code within the codebase:

IP Considerations



Finally, compatibility with modern AI tools may be limited by MockCo's reliance on legacy components; updating these systems or implementing compatibility solutions will be necessary to enable smooth AI integration and future scalability.

Recommendations for AI Enablement

To meet AI processing demands, MockCo should consider **adding GPU-enabled instances and tiered storage solutions**, like Amazon S3 and Redshift, to handle both model training and large data volumes efficiently. Expanding storage and compute resources would enable faster model training, streamlined data access, and better scalability. Improving data governance is essential as well; establishing policies for data standardization and automating data processing pipelines would enhance data quality and minimize errors in model training. These practices will ensure clean, consistent data for AI applications, reducing manual intervention and supporting more reliable outcomes. Moreover, MockCo could establish a centralized data repository to facilitate collaboration across teams, allowing AI models to draw from a unified source of truth.

With LLM-generated code making up a significant portion of the codebase, MockCo should implement **automated code review tools to verify machine-generated code for security and consistency**. This step would help prevent issues early, reducing potential technical debt while ensuring compliance with coding standards. Regularly monitoring and updating these tools to reflect new security standards would further protect MockCo's platform from vulnerabilities. Additionally, providing training on best practices for interacting with LLMs could help developers optimize the quality of generated code and reduce rework. As the volume of LLM-generated code grows, incorporating formal documentation practices specific to AI-driven development will also be important, ensuring that the rationale behind automated code changes is clear and traceable.

AI-driven observability tools, integrated with platforms like Prometheus and Grafana, could further enhance real-time monitoring, allowing MockCo to detect system anomalies and respond to issues proactively. These tools could also provide valuable metrics for continuously improving infrastructure resilience, particularly during peak loads. Together, these actions provide a robust pathway to support scalable, secure AI enablement across MockCo’s platform, laying a foundation for sustainable growth and improved client outcomes.

People and Process

Overview

MockCo's team structure and agile methodology support flexibility in development but lack formalized processes in key areas, such as onboarding, release management, and documentation. The absence of structured training and knowledge-sharing practices has led to longer ramp-up times and increased reliance on core team members. Addressing these gaps with more standardized workflows and enhanced documentation would improve productivity and scalability as MockCo continues to grow.

Project Management

All management and technical leads report directly to the Chief Technology Officer (CTO). This hierarchical structure centralizes technical oversight and ensures alignment across development, support, and quality assurance teams. Key roles within the project management framework include:

- **CTO:** Oversees all technical decisions, guiding product feature and UX directions, and setting the strategic direction for the technology team.
- **Chief Architect:** Manages day-to-day development activities, ensuring technical cohesion and making architectural decisions.
- **Head of Support:** Manages customer-facing technical resources, addressing customer inquiries and technical challenges.
- **Quality Manager:** Ensures that quality assurance processes meet internal standards, supporting product reliability and customer satisfaction.
- **Product Marketing Manager:** Acts as a communication bridge, gathering user feedback to inform product improvements and aligning product development with market needs.

This structure promotes accountability by assigning clear responsibilities to each technical lead, improving coordination and resource allocation across teams.

Release Management

Prior to initiating work on a new major or minor release, the source code structure is branched. This branching allows for the parallel development of new features alongside ongoing maintenance of previous releases.

- **Branching for Release:** Each release branch is created by the Development Manager. Any bug fixes or changes made in a non-current branch are propagated to all future branches to maintain consistency across versions.
- **Release Finalization:** As the release date approaches, the team completes feature development and bug fixes, conducting final rounds of testing. Developers with completed tasks assist with testing, support, or minor feature development.
- **Technical Release Criteria:**
  - All requirements (PBIs) for the current version must be completed and verified.
  - All bugs for the current version must be resolved.
  - All requirements (PBIs) must be covered by test cases.
- All automated unit tests, build tests, and UI tests must pass successfully.

Additional non-technical release criteria, such as documentation updates and training, are also required before finalizing a release.

For each major release, Azure DevOps manages an automated build process with steps including pre-build tests, source retrieval, unit testing, binary obfuscation, staging setup, digital signing, and UI tests. After a release, assets are archived for backup and future branch management. For urgent fixes, hotfixes with limited testing are deployed, prioritizing speed over extensive testing.

Development Process

MockCo uses a modified Agile Scrum methodology, emphasizing value-added activities like team interaction, development, and testing. Flexibility is prioritized, allowing adaptation to project needs. Key roles in the development process include:

- **Business Manager:** Provides product functionality direction and final approval on development-related guidelines and methodologies.
- **Subject Matter Experts (SMEs):** Experts in relevant fields who contribute product feature ideas and offer technical guidance.
- **Development Manager:** Oversees development and quality, ensuring projects stay on schedule and guiding technical and design decisions.
- **Software Architect:** Senior developers responsible for the technical strategy and design of the product.
- **Software Developers, Support Engineers, Quality Manager, Quality Engineer:** Contribute to development, support, and QA tasks.

Code Review

Code reviews are conducted for complex tasks or on request. The Development Manager or Software Architect reviews tasks for coding standards, reusability, and consistency. Quick consultations are used for minor issues, while formal reviews ensure adherence to quality standards.

Feature Development

Product Backlog Items (PBIs) are managed in Azure DevOps, with entries handled by the Development Manager for consistency. PBIs are assigned to specific releases or prioritized for future development based on business needs.

Backlog Grooming and Prioritization

The Development Manager, along with input from the Business Manager and SMEs, reviews the backlog to prioritize PBIs and bugs for upcoming releases. Items are selected based on business needs, complexity, and development time available.

Sprint Planning

Software Developers plan their sprint work based on priorities, schedules, and anticipated distractions. For a two-week sprint, developers aim to schedule 80 hours, including tasks, code reviews, and other responsibilities. The Development Manager reviews and approves sprint plans.

Onboarding and Training

Onboarding at MockCo is informal, leading to longer ramp-up times and reliance on existing team members for guidance. Training materials are minimal, and documentation for key workflows is inconsistent. A structured onboarding program, complete with dedicated resources and training materials, would streamline new hires' integration and reduce dependencies on existing staff.

Build Tools and Automations

MockCo’s CI/CD pipeline is rudimentary, providing basic automation for building and testing but lacking advanced features like security integrations and automated deployment. Expanding automation tools to include security testing, code quality checks, and deployment automation would enhance efficiency and improve overall code quality.

Documentation

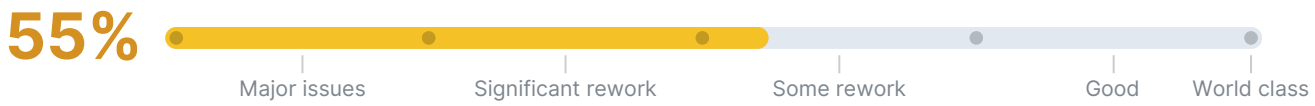
Documentation is limited, particularly in core systems and critical workflows, which impedes new developers' understanding of the codebase. Documentation is typically written reactively. A proactive documentation strategy, with assigned documentation owners, would improve knowledge sharing and reduce dependency on tribal knowledge.

QA and Test Automation

MockCo has minimal QA processes with limited test automation, primarily focused on unit testing. There is no consistent use of integration or end-to-end testing, which increases the risk of production issues. Expanding test automation to include integration and end-to-end testing would enhance quality assurance and reduce the frequency of post-release bugs.

KEY TAKEAWAYS

MockCo's team structure supports agility, but critical skill gaps—particularly in AI, secure coding, and specialized technical roles—create dependencies on a few key contributors. Filling these gaps and fostering knowledge sharing will reduce bottlenecks, increase resilience, and enhance team productivity as demands grow.

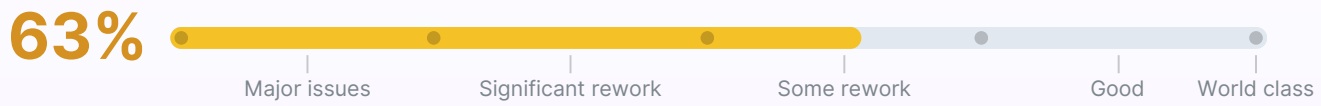


Recommendations

- **Hire or upskill team members** in areas like AI, security, and specialized technical skills to fill existing gaps and distribute knowledge across the team.
- **Implement cross-training programs** to reduce dependencies on core contributors, ensuring knowledge transfer and enhancing team flexibility.
- **Introduce structured knowledge-sharing sessions** to facilitate continuous learning and improve alignment across technical roles.

Overall Outlook for MockCo Technologies

OVERALL RATING



MockCo Technologies **demonstrates strong potential** within the logistics software space, with a product that effectively addresses market demands for operational efficiency and real-time data visibility. While the company has achieved significant traction and built a scalable platform, **several foundational areas require attention to support long-term growth**. Currently, MockCo's focus on rapid feature development has led to some trade-offs in scalability, security, and process structure, which could impact its resilience and competitive standing if not addressed.

Investments in AI, automation, and data governance present key opportunities for MockCo to differentiate itself within the industry. By enhancing its existing infrastructure and standardizing development practices, MockCo can better accommodate future growth while improving product reliability. These improvements will be particularly valuable as AI integration expands, creating new client-facing capabilities in logistics optimization and automated support. Addressing these areas will allow MockCo to continue innovating while building a stable foundation that ensures sustained growth and client satisfaction.

Moving forward, MockCo's ability to balance rapid feature releases with quality control and security standards will be critical. Emphasizing structured development processes, robust data governance, and technical debt management will position MockCo as a resilient, client-focused provider. The company's adoption of emerging technologies, particularly in AI-driven analytics and automation, could offer unique value to its clients and give MockCo a competitive edge in logistics solutions. Overall, with a focus on foundational improvements and client-centered innovation, **MockCo is well-positioned to capitalize on market opportunities and establish itself as a trusted leader in the logistics technology sector**.