Meeting Notes - VV Platform Tech Review

Key Observations

1. **Modern AWS-Native Architecture**
   The system is built using a serverless AWS stack (Cognito, Lambda, DynamoDB, CloudFormation), which supports scalability and cost-efficiency. However, certain architectural decisions (e.g., lack of API Gateway) may introduce potential risks that should be reviewed for production readiness.

2. **Limited Documentation and Operational Runbooks**
   While the platform is functional, there is minimal technical documentation, runbooks, or onboarding material. This creates barriers for future engineers or support staff and reinforces reliance on core individuals like Kristian.

3. **Security Posture Good but Needs Maturation**
   Basic security mechanisms are in place (e.g., AWS Cognito, RBAC), and the platform passed a recent penetration test. However, there is no routine audit process, unclear API protection strategy, and under-documented data privacy handling — all of which must be addressed to support enterprise adoption.

4. **Support Structure Is Thin but Supplemented**
   Kristian currently works part-time (2 days/week), with supplemental support from “Made” Labs (~3 weeks/month). This setup limits velocity and increases delivery risk, particularly if technical issues arise or additional integrations are required.

5. **Strong Foundation, but Requires Mature Operational and Technical Planning**
   The platform demonstrates good product-market fit and solid infrastructure fundamentals. However, scaling it into a mature, supportable, and maintainable solution will require strategic and tactical planning up front to ensure a smooth, cost effective way to position the platform for growth. |

6. **Kristian’s Role and Impact**
   The platform’s architecture, development, and institutional knowledge are heavily dependent on Kristian, who has acted as the sole builder and primary architect. His continued involvement is critical in the short to mid term for maintaining platform stability, enabling knowledge transfer, and supporting any scale-up or transition efforts.

7. **VV 360 and VV Pro Are Not Feature-Equivalent**
   VV360 is not currently at parity with VV Pro in terms of functionality. This discrepancy warrants further discussion to evaluate the intended purpose of each product, understand the feature gaps, and determine if they are meant to converge or serve different market segments.

Technology Stack Overview

The vessel management platform is built on a modern cloud-based technology stack leveraging Amazon Web Services (AWS) as its foundational infrastructure. Below is a detailed breakdown of the key components:

- **AWS Hosting**: The platform is hosted on AWS, providing scalable and reliable cloud infrastructure. This ensures high availability and flexibility as the platform grows.

- **Authentication with AWS Cognito**: For authentication and user management, the platform uses AWS Cognito. This provides out-of-the-box security layers and identity management features, ensuring secure access to the system.

- **DynamoDB for Data Storage**: The data layer is built on DynamoDB, a fully managed NoSQL database service. This choice supports flexible, scalable data storage and is well-suited for future growth, including potential integration with machine learning or AI features.

- **AWS Lambda for Backend & Business Logic**: The application uses AWS Lambda functions to handle business logic. These serverless functions allow for event-driven execution and scalability without the need to manage server infrastructure.

- **Front-End with React**: The current front-end of VV360 is built using React. While this provides a dynamic user interface, there is an open discussion about potentially modernizing it by transitioning to Next.js for enhanced performance and server-side rendering capabilities.

- **GraphQL for API Interactions**: The platform utilizes GraphQL for CRUD operations, enabling flexible and efficient data fetching tailored to the front-end's needs.

- **Infrastructure as Code with CloudFormation**: For infrastructure deployment, the team uses AWS CloudFormation, which allows them to define and provision infrastructure through code, ensuring consistency and easier management.

- **DNS and SSL/TLS via Route 53** AWS Route 53 is used for domain name system (DNS) management and for provisioning and maintaining SSL/TLS security certificates, ensuring reliable routing and encrypted communication across environments.

- **Source Control and CI/CD with GitHub**: Source control is managed via GitHub, with CI/CD pipelines implemented using GitHub Actions to automate testing and deployment workflows.

- **Environment Separation** The platform is deployed across development, staging, and production environments, enabling structured release workflows, feature validation, and operational isolation.

- **Feature Flags for Controlled Rollouts** Some platform functionality is wrapped behind feature flags, allowing for safer deployments, easier testing, and controlled rollouts without requiring full code redeploys.

- **Offline-Capable Architecture** The platform is architected to support local data management in offline conditions. When internet connectivity is lost, the application can continue to function locally, with data syncing automatically once the connection is reestablished — a critical feature for vessel-based operations.

Integrations and Dependencies Overview

In addition to its core technology stack, the vessel management platform incorporates several third-party integrations and dependencies to extend functionality and streamline operations. Below are the key integrations and their purposes:

- **QuickBooks Integration**: The platform integrates with QuickBooks to handle customer-related information, likely focusing on billing and financial workflows. This ensures seamless financial management and invoicing.

- **HubSpot for CRM**: HubSpot is used as the CRM tool, managing email correspondence, messaging, and customer interactions. This helps maintain strong customer relationships and communication.

- **Polaris for Oil Analysis**: There's an integration with Polaris for oil analysis, which likely supports maintenance and condition monitoring of vessels. This enables proactive upkeep and operational efficiency.

- **Admin Platform for Configuration**: A native admin platform is used to configure new customers, manage templates, and handle vessel inventory setup. This simplifies onboarding and customization for clients.

- **OEM Data Sources**: The platform sources data from various OEMs to enhance functionality and analytics. This allows for more detailed insights and personalized features for each vessel.

- **Linear for Project Management**: The team uses a project management tool called Linear to catalog service tickets and manage the development backlog. This ensures that development tasks and support issues are tracked and addressed efficiently.

Security Assessment

The platform demonstrates a foundational approach to security, leveraging some AWS-native services with secure configurations, but still presents notable areas for improvement to mature its posture and reduce operational risk.

Strengths

- **Authentication via AWS Cognito**
  The application uses AWS Cognito to manage user authentication and identity, offering built-in security, scalability, and compliance advantages. This choice provides a strong baseline for access control and is well-integrated into the broader AWS ecosystem.

- **Successful Penetration Test**
  A third-party penetration test was conducted at the request of a customer. The platform passed the test with favorable results, which is a positive indicator of its baseline security resilience and current risk exposure.

- **Role-Based Access Control (RBAC)**
  The system enforces role-based permissions to control access across user types and functionalities. However, the configuration and enforcement of RBAC rules may not be well-documented, potentially complicating future audits or onboarding of additional engineering/support staff.

- **DNS and SSL/TLS via Route 53** AWS Route 53 is used for domain name system (DNS) management and for provisioning and maintaining SSL/TLS security certificates, ensuring reliable routing and encrypted communication across environments.

Areas for Improvement

- **Lack of Ongoing Security Audits**
  There is currently no recurring process for internal or external security audits. While the successful pen test is encouraging, the absence of routine assessments increases the risk of future vulnerabilities going undetected as the codebase evolves.

- **Unclear API Gateway Strategy**
  Lambda functions are used to handle backend logic, but there is no clear documentation of how requests are routed to them. The absence of a defined API

Gateway raises questions about:

  - How endpoints are exposed or secured
  - Whether access is appropriately throttled or monitored
  - Whether TLS, rate-limiting, and WAF (Web Application Firewall) protections are applied
    This represents a potential surface area of vulnerability and should be clarified and addressed.

- **Limited Documentation for Security Configurations**
  Key security configurations, including IAM permissions, environment separation, and deployment procedures, are under-documented. This may slow down incident response, reduce transparency, and create knowledge silos that increase dependency on core contributors.

- **Data Privacy and Compliance Gaps**
  While AWS offers tooling to manage PII and ensure compliance (e.g., with GDPR), there is no indication that a formal privacy impact assessment or data classification policy has been conducted. It's unclear:

  - How sensitive data is identified and protected
  - Whether encryption-at-rest and encryption-in-transit are consistently applied
  - What retention or data deletion policies exist

These gaps should be addressed as part of a maturing compliance posture particularly if the product is targeted toward enterprise clients or global markets.

DoD Compliance Considerations

1. **Security and Data Protection:** We have to follow special rules (like NIST standards) that make sure any data we handle is well-protected. This is to keep everything secure and private.

2. **GovCloud and Compliance Regions:** We might use a special part of Amazon's cloud service called GovCloud. This is a version of the cloud that's designed just for government-related work, making it easier to meet federal standards.

3. **Using Secure Tools:** We're using tools like AWS Cognito for managing logins and DynamoDB for storing data. Both of these have strong security features, like encryption,

that help us stay compliant.

4. **Certifications We'll Need:** We'll need to meet certain certification standards, like CMMC, which is basically a checklist to prove we're handling data safely and securely.

Key Recommendations

Prioritize Product Objectives: Build a structured product prioritization framework that balances new features, technical debt, and quality-of-life improvements based on impact, level of effort, and customer value.

Collaborate and formalize a focused 90-day product roadmap that aligns technical execution with strategic product objectives. The roadmap should:

- **Prioritize Feature Development**: Evaluate and rank upcoming features based on user value, business need, and complexity.
- **Assess Technical Debt**: Identify areas of the codebase or architecture that require refactoring or optimization to improve long-term maintainability.
- **Estimate Impact and Effort**: Use a structured framework (e.g., RICE or MoSCoW) to evaluate the value, impact, and effort of each initiative.
- **Balance Quick Wins with Strategic Goals**: Incorporate both short-term improvements and foundational investments that support future scalability and stability.
- **Create Visibility and Accountability**: Translate roadmap items into engineering deliverables with clear milestones, owners, and success metrics.
- **Facilitate Knowledge Transfer**: Build documentation and workflows into the roadmap to ensure knowledge is distributed and not locked with a single contributor.