Nail It

Nail It

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tagline: Constructing dreams, One brick at a time.

This document will serve as a project outline for Nail It, an Inventory Management app that has been designed to help users find streamlined solutions to manage construction materials. It allows businesses to manage stocks of what remains and what is needed thus ensuring that there is a good number of materials on site ensuring continuity of events in real time.

TEAM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Members: Brian Bett & Njenga Mwangi

Roles of Team Members:

Brian Bett- Main Front-End Web Developer

Njenga Mwangi - Support Front-End and Back-End Developer

(In Search for the last team member as the main backend developer)

Brian Bett was chosen to be the main Front-End Developer because he has a background in Front-End development. He’s W3C compliant and also good at making websites responsive.

Njenga Mwangi was chosen to act as support because he is good at making research, graphics design for our web app’s logo and all graphics design needed and he’s also good at color selection. He is good at building test cases and searching for bugs.

At the moment, we don’t have the main Back-End Developer for our team to be complete.

TECHNOLOGIES

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LIBRARY

Core Library: Redux

Redux is chosen because of its features in state management, which in the case of a single-page application like Nail It, is very important in the complex interaction of data handling.

Problem: State Management in Vue.js

Another library for state management that can be used is MobX. It is less boilerplate and more compact in terms of syntax than Redux. Despite this, Redux was selected because of its wide adoption and good ecosystem in terms of middleware and tools.

LANGUAGES

Main Language: JavaScript

We chose JavaScript so that this tool would be versatile and appropriate for web development.

Alternative Implementations: Python

Python, on the other hand, is another main language in web development, with a very wide set of libraries. However, it was not picked because of its asynchronous limitations against JavaScript, which is crucial for real-time updates within an inventory management app.

PLATFORMS

Main Platform: Node.js

Node.js is a good fit for an event-driven and non-blocking I/O nature, supporting the asynchronous operation of a real-time application.

Alternative: Django

Django is a high-level Python web framework encouraging rapid development and clean, pragmatic design; reusability and simplicity Node.js was chosen because of its performance advantages and how it fits into the selected frontend technologies.

FRAMEWORKS

Core Framework: React

I chose to use React.js because it is component-based, which dramatically simplifies how one goes about creating dynamic user interfaces.

Alternative: AngularJS

AngularJS and React.js both share some functionality. Still, AngularJS provides more functionality than React.js, and it is difficult to learn and less flexible in working with libraries and frameworks.

BOOKS

Books: "Learning React: A Hands-On Guide to Building Web Applications Using React and Redux" by Kirupa Chinnathambi

RESOURCES

Online Resources: MDN Web Docs, React documentation, Redux documentation.

CHALLENGES \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem Statement: Nail It aims to help individuals and businesses manage their inventory in a responsive and intuitive manner across multiple job sites. It helps solve issues such as delays and wasted time searching for materials or tools that, in the end are not there which then results in workers idling at the construction site.

Limitations: Where Nail It provides convenience in trying to manage inventory, it does not help in solving bigger problems like vulnerability to loss for sites that do not have physical checks and poor demand forecasting by using a company’s ales history to estimate future customer demand.

Target group: People who are involved retail, manufacturing, real estate and other construction inventory-intensive sectors: Nail It is developed to be used globally. It is not tied to a certain localization.

RISKS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Technical Risks:

Data privacy: Because most of the inventory management applications demand access to one's social media accounts, the applications can thus have access to crucial personal information contained in the accounts, including passwords and personal data that are always prone to compromise through hacking or misuse.

Security Vulnerabilities: Being a software program like any other, the inventory management app undoubtedly includes security vulnerabilities that black-hat hackers can easily take advantage of, therefore possibly leaking information or gaining unauthorized access to some accounts.

Reputation management: the inventory management apps create errors in posting, post the wrong message, or post at the wrong time if proper management is not done. Usually, these are very small mistakes that could have huge impacts on the brand and reputation of a company.

Legal and Compliance: A inventory management app has to be compliant with diverse laws and regulations regarding data protection, advertising, and consumer rights. Failing to do so might bring them some trouble and impose fines.

Dependency on Third-party Platforms: Third-party social media management means that you're subject to their terms of service, any changes in pricing, and their ability to disrupt services at any given time. In that scenario, downtime of the platform or discontinuation of service could imply total disturbance in social marketing efforts.

Algorithm Changes Often social platforms make changes to their algorithms, which has a significant impact on the reach and visibility of your content. This change is not tracked in most inventory management apps; therefore, these changes might directly impact your social media strategy.

Overall, these risks can be mitigated by various factors, such as carefully researching and vetting any inventory management apps that are going to be used, ensuring strong cybersecurity measures, reviewing the company’s social media strategy regularly and updating, and staying current with industry trends and best practices. Furthermore, a policy of openness and transparency with an audience may help create a positive reputation for the business, despite the use of automation and analysis tools.

Non-Technical Risks:

Inconsistency and inauthenticity in your brand voice are huge risks. With heavy dependence on automation features, the content that goes live on your social media pages may sound robotic or detached from your brand's personality, alienating your audience.

Real-Time Engagement: In general, apps cut down the struggle of managing numerous inventories keeping accounts, but they can, by the same measure, impede the real-time engagement with your users, which might leave them unsatisfied and unengaged.

Over-Emphasis on Metrics: The performance analytics in apps for managing construction inventory are ideal in the tracking of performance, but in overvaluing some of the vanity metrics, one can really go the wrong way, losing sight of more meaningful engagement metrics and real interactions with the audience.

The risk of amplifying negative feedback: Unmonitored, management applications for inventory management can take part in amplifying such negative feedback or criticism automatically, with their automated responses to controversial content. This is one of the ways that problems escalate and could damage your brand if not handled well.

Legal and Regulatory Risks: Legal and reputational views of your business could be most punishing from the viewpoint of non-compliance with laws and regulations governing advertisement, privacy, and consumer rights. Hence, the relevant law and guidelines will be effectively managed by the management of inventory management applications.

These nontechnical risks can be reduced through the provision of clear guidelines and protocols in managing construction inventory, continuously demonstrating genuine engagement and authenticity in what takes place in and through the system, and remaining agile to changes in the landscape of track-keeping, while taking the opportunity to review and update the inventory in accordance with values and expectations of the audience.

INFRASTRUCTURE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For branching and merging in our team's repository, we would likely follow a Git workflow such as GitHub flow. Here's how our process might look:

Main Branch: The main branch would represent the stable version of our app. All development work would branch off from main, and it would only be merged into after thorough testing and code review.

Feature Branches: When working on a new feature or a bug fix, team members would create a new branch off of main. These branches would have descriptive names related to the feature or issue being addressed.

Pull Requests: Once a team member completes their work on a feature branch, they would create a pull request (PR) to merge their changes back into main. PRs would undergo code review by other team members to ensure code quality and adherence to coding standards.

Continuous Integration (CI): We would set up CI pipelines to automatically run tests whenever a new pull request is created or updated. This ensures that changes don't introduce regressions and helps catch issues early in the development process.

Merge: After a PR is approved and passes all tests, it can be merged into main. We would use merge commits to preserve the history of changes and provide context for future development.

For deployment, our strategy would depend on the specific requirements and infrastructure of our app. However, a common approach might involve:

Staging Environment: Before deploying to production, we would deploy changes to a staging environment for final testing and validation. This environment closely mirrors production but is isolated from real users.

Continuous Deployment: Once changes are merged into main and pass all tests, they would automatically be deployed to production. We would use deployment pipelines and automation tools to streamline this process and minimize downtime.

Rollback Plan: In case of any issues or unexpected behavior in production, we would have a rollback plan in place to quickly revert to a previous stable version.

To populate our app with data, we might use a combination of techniques:

Manual Data Entry: For smaller datasets or initial testing, we could manually enter data directly into the app or database.

Seed Data: We would create scripts or fixtures to generate sample data for development and testing purposes. These scripts could be run automatically as part of the deployment process.

Importing Data: If migrating from an existing system or integrating with external sources, we would develop import scripts or use data migration tools to transfer data into our app's database.

For testing, we would employ a variety of tools, automation, and processes to ensure the quality and reliability of our app:

Unit Tests: Developers would write unit tests to validate the behavior of individual components and functions in isolation.

Integration Tests: We would write integration tests to verify that different parts of the system work together correctly.

End-to-End Tests: Using automated testing frameworks like Selenium or Cypress, we would conduct end-to-end tests to simulate user interactions and verify that the app functions as expected from the user's perspective.

Continuous Testing: We would integrate testing into our CI/CD pipelines to automatically run tests whenever changes are made to the codebase. This helps catch bugs early and ensures that new features don't introduce regressions.

Manual Testing: While automation is valuable, we would also supplement it with manual testing by QA engineers to explore edge cases, validate user experience, and ensure overall product quality.

By following these practices for branching and merging, deployment, data population, and testing, we aim to maintain a stable and reliable social media management app that meets the needs of our users and delivers a positive experience.

EXISTING SOLUTIONS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Zoho Invetory:

Similarities: Like Nail It, Zoho Inventory is an inventory management platform that allows users to manage stock levels, manage activity amongst multiple warehouses etc.Differences: Zoho Inventory offers more advanced features such as Report generation within seconds, sentiment analysis, and team collaboration tools, which may be beneficial for larger businesses.

ERPAG:

Similarities: ERPAG is another inventory management tool that enables users to streamline their order-to-delivery process and optimize inventory management. Differences: ERPAG focuses more on simplicity and ease of use, with a streamlined interface and fewer advanced features compared to Zoho Inventory an also manages cash flow.

Just-In-Time:

Similarities: Just-In-Time management system provides a form of inventory that requires working closely with suppliers so that raw materials arrive a production is scheduled to begin, no sooner. Differences: Just-In-Time offers robust reporting and analytics tools, as well as customer relationship management (CRM) integrations, making it suitable for businesses focused on customer engagement and relationship building.

Choice of Reimplementation: Nail It aims to reimplement proven solutions in the inventory management space, leveraging the best features from existing platforms while addressing their limitations.

After evaluating various solutions, including ERPAG, Just-In-Time and Zoho Inventory, Nail It chose to focus on a user-friendly interface, seamless integration with multiple inventory management platforms, and real-time analytics capabilities. By combining the strengths of these platforms while providing a more intuitive and customizable user experience, Nail It aims to differentiate itself in the market and meet the evolving needs of the manufacturing industry and businesses.