MakerSuite: Lightweight ERP For Makers
Interim Report

TU856
BSc in Computer Science

Student Name: Jenny Thao Huynh
Student Number: C22448184

Supervisor: Jelena Vasic

School of Computer Science
Technological University, Dublin

November 2025

Abstract

The rise of the Maker Movement has enabled individuals to turn creative practices into small-scale entrepreneurial ventures, yet many maker entrepreneurs face significant challenges in managing the diverse and fragmented business operations required to sell handmade, tangible products across multiple platforms. Makers often work across digital marketplace platforms, social media channels and personal websites, each with their own systems for managing listings, inventory, customer communication and order fulfilment. Existing tools such as spreadsheets, generic productivity platforms and traditional ERP systems lack the flexibility, affordability and personalisation required to support the highly individualised workflows of maker businesses.

MakerSuite is proposed as a lightweight, web-based ERP solution designed specifically to meet the niche needs of maker entrepreneurs. Its purpose is to centralise core operations, including product listings, inventory tracking, workflow management and customer communication, into a single integrated platform that reflects how makers actually work. Drawing on both secondary research into existing solutions and primary research conducted with current and aspiring makers, this interim report presents the background, research findings and system analysis that inform the initial specification of MakerSuite. The report outlines the key functional and non-functional requirements identified, and introduces the requirements diagram that will guide subsequent design and development. As no implementation has yet been undertaken, the emphasis of this interim stage is on establishing a clear understanding of user needs and translating these into a structured foundation for the system’s design and future development.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Jenny Thao Huynh

_____

Jenny Thao Huynh

November 2025

Acknowledgements

I would like to recognise my Final Year Project supervisor, Jelena Vasic, for her invaluable patience, feedback and support during the development of this interim report and project. Many thanks to the Final Year Project coordinators, Ciaran O'Driscoll and Sean O'Leary for their guidance. I would also like to express gratitude for the user group that agreed to participate in my survey for user research and to test early prototypes of MakerSuite.

Contents

1. Introduction 1
1.1 Project Background 1
1.2 Project Description 2
1.2.1. Project Purpose 2
1.2 Project Aims and Objectives 3
Overall Aim 3
Objectives 3
1.4 Project Scope 4
In Scope 4
Out Of Scope 5
1.5 Thesis Roadmap 5
2. Literature Review 7
2.1 Introduction 7
2.2 Alternative Existing Solutions 7
2.2.1 Inventora 7
2.2.2. Craftybase 7
2.2.3. Vendoo 7
2.2.4. Front CRM 8
2.3 Technologies Researched 8
2.3.1. Frontend Technologies 8
Comparison Of Technologies 8
Selection 10
2.3.2. Backend Technologies 10
Introduction 10
Comparison Of Technologies 10
Selection 12
2.3.3. Database Technologies 13
Introduction 13
Comparison Of Technologies 13
Selection 14
2.4 Other Research 14
2.4.1. Marketplace API's 14
2.5 Existing Final Year Projects 15
2.5.1. Culture Bridge, Sara Egan 15
2.5.3. QueueEase, Ian Hipolito 16
2.6 Conclusions 17

3. System Analysis 19
3.1 System Overview 19
3.2 Requirements Gathering 19
3.2.1. Stakeholder Identification 19
3.2.2. Research Activities 20
3.3 Requirements Analysis 22
3.3.1 Analysis of User Needs 22
3.3.2 Prioritisation of Functional Areas 22
3.3.3 Mapping Pain Points to System Capabilities 23
3.3.4 Non-Functional Considerations 23
3.3.5 Summary of Analysis 23
3.4. Initial System Specification 23
3.4.1. Functional Requirements 23
3.4.2. Non-Functional Requirements 24
3.4.3. Requirements Diagram 25
3.5. Conclusions 26
4. System Design 27
4.1 Introduction 27
4.2 Software Methodology 27
4.3 Overview of System 27
4.4 Design Of The System 29
4.4.1. High-Level Use Case Diagrams 29
4.4.2. High-Level Sequence Diagrams 35
4.4.3. Database Design (ER Diagrams) 38
4.5. Conclusions 39
5. Testing and Evaluation 40
5.1 Introduction 40
5.2 Plan for Testing 40
5.3 Plan for Evaluation 40
5.4 Conclusions 41
6. System Prototype 41
6.1 Introduction 41
6.2 Prototype Development 41
6.3 Results 42
6.4 Evaluation 42
6.5 Conclusions 42
7. Issues and Future Work 43

7.1 Introduction 43
7.2 Issues and Risks 43
7.3 Plans and Future Work 43
7.3.1 Project Plan with GANTT Chart 44
References A-1
A) Appendix A: Online Survey Google Form (Help Shape A New Tool For Small Handmade Business Owners) A-2
B) Appendix C: Online Survey Data (Help Shape A New Tool For Small Handmade Business Owners) B-1

1. Introduction

1.1 Project Background

(Some background and literature, start with an interesting fact or a newspaper item)

The "Maker Movement" which gained momentum in the 1990's, popularised the do-it yourself creation of physical products, crafts and arts, with underlying support from technologies to enable the creation of designs and fabrication of goods [1]. This movement gave rise to a new genre of entrepreneur, *"the maker entrepreneur"*. These are individuals who turn their creative endeavours into business opportunities by independently making and selling tangible products (e.g. jewellery, knitwear, ceramics). A recent study of young people's career aspirations revealed that roughly two thirds of those aged 18-34 desire to start their own business, with 37% wanting to work independently [2].

The emergence of technologies such as digital marketplaces platforms, social media platforms and digital tools allow for the easy transition from maker, to maker entrepreneur. However, this transition introduces certain barriers to entry for small-scale entrepreneurship. Many maker entrepreneurs are makers first and entrepreneurs second, and their pain points include lacking the time, resources and expertise to manage the overwhelmingly wide variety of business operations (e.g. product design and creation, inventory management, product listing, finances, order fulfilment, customer retention) [3].

Small-scale *"maker businesses"* face unique challenges of independently making and selling tangible products. Handmade products are often custom, made-to-order, one-of-a-kind and each can require its own creative workflow to track raw materials, tools, production time, pricing and delivery logistics [4]. Unlike standardised manufacturing processes common in larger SMEs, these workflows are highly flexible and personal, making them difficult to manage with generic business tools that weren't designed to meet their niche needs. Additionally, in today's technological landscape, maker entrepreneurs often sell their products across diverse channels, including digital marketplace platforms (e.g. Etsy, Shopify, Depop), social media platforms (e.g. Instagram, TikTok, Facebook) and personal websites [5]. Each of these channels have their own systems for managing listings and inventory, customer communications and order fulfilment. Not only do maker entrepreneurs have to learn the tools and processes associated with their craft, they have the additional task of managing workflows across multiple channels, creating needs that are fundamentally different from that of traditional SMEs [6].

While makers have access to a range of creativity-focused tools, there remains a significant gap in tools that address their niche needs. Existing general-purpose tools such as Notion and Excel allow users to manually build custom workflows but these platforms require significant time investment, technical effort and prior business knowledge [7]. Similarly, existing software solutions like SAP Business One, CrossList, Inventora, Vendoo and QuickSync cater to traditional SMEs but lack the flexibility, affordability and personalisation required by maker businesses.

The above research, combined with personal experience as the owner of a maker business selling handmade products (beaded jewellery, crochet bouquets, illustrations) and insights gathered from the maker community, highlights the need for a solution in the form of an integrated platform tailored to meet the niche needs of a maker businesses, such as MakerSuite.

1.2 Project Description

1.2.1. Project Purpose

MakerSuite is a lightweight ERP solution designed specifically for maker businesses. An ERP (Enterprise Resource Planning) system is a type of software that centralises and streamlines key business processes, such as inventory management, product listings, communications, and analytics. into one integrated platform.

The purpose of MakerSuite is to provide a web-based ERP solution designed to meet the niche needs faced by maker entrepreneurs who independently create and sell handmade, tangible products (e.g. jewellery, knitwear, ceramics) so that they can manage their business operations without the complexity or cost of traditional ERP tools.

Users often juggle creative production with unique workflows and complex business operations across multiple channels such as digital marketplace platforms (e.g. Etsy, Shopify, Depop), social media platforms (e.g. Instagram, TikTok, Facebook) and personal websites. Each have their own system for business operations, MakerSuite aims to bring all these systems across multiple platforms into a single, integrated platform that centralises business operations (e.g. managing listings, inventory, order fulfilment and customer communications). Additionally, it aims to allow users to manage the production of their handmade products by being designed around their unique workflows and track raw materials, tools, production time, pricing and delivery logistics.

MakerSuite has features designed around how makers actually work and by offering a simple integrated solution, it helps reduce barriers for individuals who may not have had the resources, technical skills or business background to use existing tools, empowering more people to grow their handmade businesses while reducing waste.

1.2.3. Core Features

Product Listing Management

Centralises product listings across multiple digital marketplace platforms (e.g. Etsy, Shopify, Depop) with one, integrated dashboard where users can update product details, pricing and inventory. This allows users to manage product listings in a convenient and consistent way, instead of individually using each platform’s unique systems.

Inventory Management

Manages and tracks the inventory of finished products, as well as the raw materials, tools and other supplies required to create them. This allows users to be able to reproduce products, reorder materials, track material usage, in turn supporting more sustainable practices and improving waste reduction.

Product Workflow Logger

Allows users to document the full creative and production process for each handmade product and link it directly to MakerSuite’s Inventory Management Feature and Product Listing Management Feature. The user can log the raw materials and tools used, time spent, photo uploads and add custom notes before linking it to a product listing. This allows users to stay organised, stay consistent across custom orders and calculate pricing

Customer Communications Management

Combines communications from a variety of sources (e.g. built-in messaging systems from digital marketplace platforms, emails, social media messages) into one single inbox. This allows users to track customer conversations in one place without switching platforms.

Analytics And Reports

Tracks sales performance, inventory usage, and customer engagement across all connected platforms. This helps users identify trends in their best-selling products, monitor material consumption, and understand customer behaviour. This allows users to reduce waste, improve sustainability, and make informed decisions to retain customers and grow their business.

Accessibility and Inclusive Design

Follows accessibility best practices and standards, the web application will be usable by keyboard, mouse, touchscreen, screen reader and voice assistant users. This allows users regardless of auditory, visual, physical or cognitive abilities to understand and use MakerSuite. Beyond technical accessibility, MakerSuite also reduces certain barriers to entry for individuals who previously couldn't afford the time effort, technical effort and lacked the prior business knowledge to use existing solutions. This empowers users who otherwise would not find the means to grow their maker business.

1.2 Project Aims and Objectives

Overall Aim

The overall aim of MakerSuite is to develop a lightweight ERP system for maker businesses that meet their niche needs so that they can balance creative production with managing complex business operations. It aims to do this by supporting the unique workflows associated with the creation of handmade products as well as integrating different systems across multiple channels in a single, easy-to-use platform. MakerSuite aims to be flexible and adapt to maker entrepreneurs of any craft, whether it's jewellery creation, ceramics or 3D printing.

Objectives

- Design a flexible, intuitive system that can be used regardless of a user's specific craft.
- Develop a web application follows accessibility best practices as well as inclusive design standards.
- Ensure the web application is responsive and performs well on all device sizes.
- Centralise product listings across multiple digital marketplace platforms (e.g. Etsy, Shopify, Depop) and allow for real-time, synced management of these listings.
- Implement a dual inventory management system that tracks both raw materials and finished products to support reproducibility, documentation and reduce waste.
- Design a product workflow logger that allows users to upload photo as well as document materials, tools, time and process steps for each handmade product.

- Centralise customer communications that combines messages across multiple channels into a single inbox.
- Build data analytics and reporting tools to help users identify sales trends, monitor material usage and improve customer retention.
- Comprehensively test the solution for system reliability and customer satisfaction using testing frameworks and sample groups.

1.4 Project Scope

In Scope

Traditional ERP solutions cover a wide range of business operations, however MakerSuite is a lightweight ERP solution that focuses on providing core features and functionality that directly support the niche needs of maker businesses. These include:

- **Product Listing Management**: Develop a centralised interface where users can create, edit, import, and manage product listings across multiple online marketplaces such as Etsy, Shopify, and Depop. Include bulk actions and real-time syncing of product details.
- **Inventory Management**: Implement a dual inventory tracking system that monitors both finished products and raw materials (e.g., beads, yarn, clay, packaging). Users can link raw materials to specific products, track leftover materials, and receive low-stock alerts.
- **Customer Communications Management**: Build a unified inbox that aggregates messages from multiple channels (email, social media, marketplace platforms). Include features for search, quick replies, and template messages to streamline customer interaction.
- **Product Workflow Logger**: Enable users to document the creative process for each product, including materials used, tools, time spent, process steps, and photo uploads. This supports reproducibility, accurate pricing, and workflow documentation.
- **Analytics and Reporting**: Provide simple dashboards and reports that show sales trends, inventory usage, material consumption, and customer engagement. This helps makers make data-driven decisions, plan production, and improve sustainability practices.
- **Flexibility Across Crafts**: Ensure the system is adaptable to a variety of handmade products, from jewellery and crochet to cards, soaps, and other crafts, without being tailored to a single product type.
- **Accessibility and Inclusive Design**: Design the web application to follow accessibility best practices (e.g., WCAG AA guidelines), including keyboard navigation, colour contrast, and responsive layouts for mobile, tablet, and desktop use.
- **Testing and Documentation**: Conduct thorough unit, integration, and user acceptance testing to ensure reliability. Provide detailed documentation for system features, API integrations, and deployment instructions to support maintenance and future development.

Out Of Scope

MakerSuite will not attempt to deliver functionality outside of its overall aim to support the niche needs of maker businesses. Traditional ERP systems cover a wide range of functional areas such as providing financial accounting, marketing tools, manufacturing automation, HR management, third-party payment processing etc., these are out of scope for MakerSuite.

Aside from generic ERP features, other out of scope items include:

- Hardware integration
- The development of native mobile apps

1.5 Thesis Roadmap

This report is organised into seven main chapters/sections, each documenting a different stage of the research, design, development and evaluation of MakerSuite.

Chapter 1 – Introduction:
This chapter provides the project background, describes the purpose of MakerSuite, outlines the aims and objectives, defines the scope of the work, and presents the roadmap for the remainder of the report.

Chapter 2 – Literature Review:
This chapter reviews existing solutions relevant to maker-business management and examines technologies, methods and design approaches that influenced the development of MakerSuite. It also evaluates other final year projects and identifies gaps that the proposed system aims to address.

Chapter 3 – System Analysis:
This chapter presents the analysis phase of the project. It includes an overview of the intended system, details the requirements-gathering process, analyses user needs, and defines the initial system specification. This chapter provides the foundation for the system's design by translating research insights into functional and non-functional requirements.

Chapter 4 – System Design:
This chapter outlines the design of MakerSuite, including the chosen development methodology, high-level system structure, interface design, and detailed component-level design. It explains how the requirements from Chapter 3 were transformed into practical, implementable system models.

Chapter 5 – Testing and Evaluation:
This chapter describes the testing strategy for ensuring system reliability and performance, including both technical tests and user-focused evaluation. It outlines the testing plan, the evaluation methods used and summarises the outcomes.

Chapter 6 – System Prototype:
This chapter documents the development of the MakerSuite prototype. It covers the implementation process, presents the results, and evaluates the prototype against the project requirements. It also highlights key challenges encountered during development.

Chapter 7 – Issues and Future Work:
This chapter discusses risks, limitations and issues identified during the project. It also outlines proposed improvements and future development pathways for MakerSuite, including a project plan and Gantt chart illustrating possible next steps.

The report concludes with references and several appendices containing supplementary materials such as survey results.

2. Literature Review

2.1 Introduction

The Literature Review section details an analytical review of existing software solutions, possible technologies to use and other research relevant to the development of MakerSuite, a lightweight ERP for makers. The purpose of this review is to identify what pain points are addressed by the solutions that are currently available for makers and how, identify suitable technologies for the development of this project and finally, identify other domain-specific concepts relevant to the development of this project. This ensures that the project is informed by relevant prior work and builds upon them.

2.2 Alternative Existing Solutions

2.2.1 Inventora

Inventora is an web-based, all-in-one inventory management system designed for makers and manufacturers, that keeps track of raw materials, finished products and more [8]. The strength of Inventora is that its inventory management functionality is specifically designed to cater to the needs of makers from the start of product creation to fulfilment. Its features include COGS calculation, integration/syncing across multiple sales channels (Shopify, Wix, Etsy, Square), raw material tracking, business reports, wholesale/supply/sales order management and customer management. The weaknesses reported by users are bugs, delays in syncing across multiple sales channels and performance issues with the application [9]. Inventora addresses many of the pain points that MakerSuite aims to address however it is not a full replacement, inspiration will be taken from Inventora on how to design MakerSuite's user-friendly inventory management feature.

2.2.2. Craftybase

Similarly to Inventora, Craftybase is another web-based inventory management system designed for makers and manufacturers and keeps track of raw materials and finished products. The strength of Craftybase is that it offers more business reports and integration/syncing across more sales channels than Inventora (Shopify, Wix, Etsy, Square, Faire, Amazon Handmade, Squarespace/ Square) [10]. Similarly, weaknesses reported by users are bugs, lack of batch tracking functionality and inability to sync sales from physical sales channels outside of the online ones (e.g. from a pop-up market). Craftybase is an example of what features makers expect from a system that provides inventory management [11].

2.2.3. Vendoo

Vendoo is a cloud-based cross-listing application designed for online resellers and allows users to manage product listings across multiple online marketplaces in one place. Its features include product listing management, inventory management, bulk actions sale detection/auto delisting and business reports. The strength of Vendoo is that it allows for syncing across multiple online marketplaces used by crafters (eBay, Facebook Marketplace, Etsy, Depop, Whatnot and Vinted) and easy setup (importing existing listings or creating new ones). The weakness of Vendoo is that its tailored for the needs of resellers rather than makers and lacks functionality outside listing management, however inspiration will be taken from Vendoo on how to design MakerSuite's user-friendly product listing management feature.

2.2.4. Front CRM

Front CRM is a cloud-based all-in-one customer service software solution that manages support requests through multiple communications channels (email, SMS, social media, live chat and more) [12]. Its features include centralising customer communications on one platform, a shared inbox with teammates, automated workflows as well as analytics. Its strengths lie in allowing teams to collaboratively track, respond to and provide analytics for their customer conversations across multiple communications channels as well as save time by automating workflows. The weakness of Front CRM as reported by users is the high and unpredictable pricing structure, limited affordability for small teams, high battery consumption, and difficulty with group emails – Front CRM aggregates emails sent to a large number of users into one thread which makes it difficult to follow separate conversations [13]. Front CRM is a great example on how to build a user-friendly customer communications management system as well as provides a reference for drawbacks MakerSuite should be mindful of.

2.3 Technologies Researched

This section reviews the technologies considered for the development of this project's key components, including the frontend, backend and database. There exist a wide range of frameworks, each with their own set of advantages and limitations. An informed selection must be made by evaluating each option against key considerations including project requirements, community support, performance, scalability and learning curve.

2.3.1. Frontend Technologies

Introduction

The frontend technology selected for MakerSuite must support the development of an accessible, user-friendly and responsive user interface. As MakerSuite is intended to function as a browser-based web application, the chosen technology must perform consistently across different devices and operating systems.

Comparison Of Technologies

React

Reach is an open-source JavaScript based library for building web application user interfaces. It's designed for building dynamic, interactive and scalable web applications using reusable components, virtual DOM rendering and a strong ecosystem with tooling. It also provides JSX, a powerful syntax that lets you write HTML-like code within your JavaScript. A contributor to React's popularity is its relatively gentle learning curve as it's JavaScript based, boasts clear documentation and has a large, active community [14]. Finally. React is designed for gradual adoption, meaning you can use as little or as much react as you need [15].

React Native

React Native is an open-source JavaScript based library that uses React principles for building native mobile applications. It benefits from the component-based architecture and other features React offers while providing modules used for app development. A key feature of React Native is its "write once, run anywhere" approach. It compiles to native UI

components, enabling cross-platform development: applications can run on both iOS and Android using a largely shared codebase, and components can also render in a browser as a web application [16].

Angular

Angular is a TypeScript-based open-source library used for building web applications. It includes a component-based framework for building scalable web applications, a collection of well-integrated libraries that cover a wide variety of features (e.g. state management, forms, testing etc.) as well as a suite of developer tools to help develop, build, test and update your code. Angular applications are built for scalability, allowing for the development of enterprise-level applications [17].

Technology	Advantages	Limitations
React	• Large, active and mature community support allows for long-term stability and numerous resources for troubleshooting. • Virtual DOM rendering allows for high performance and efficiency. • Component-based architecture allows for scalability. • The strong ecosystem with several options for libraries and tooling allows for flexibility.	• Lack of built-in libraries require additional libraries to be used for functionality such as state management, adding complexity. • The use of several libraries and tooling can lead to inconsistency across the project.
React Native	• Enables cross-platform mobile app development with a shared codebase for iOS and Android. • Builds upon JavaScript and React principles. • Other benefits offered by React including the virtual DOM rendering and component-based architecture.	• Introduces complexity due to platform-specific considerations and native module management. • Increases project scope by requiring mobile deployment pipelines, device testing and updates for multiple OS versions. • Some third-party modules vary in quality, can lead to requiring custom native code for full functionality. • Not suitable for projects where the primary goal is to develop a web application.
Angular	• The component-based architecture and modularity	• Steeper learning curve due to being TypeScript based,

	improves maintainability and scalability. • Several options for robust built-in tools reduces reliance on external libraries. • Strong support exists for accessibility and enterprise-level features. • Strongly typed code with Typescript reduces runtime errors.	decorators and extensive configuration. • Slower initial load times compared to smaller frameworks. • Heavier and more complex than necessary for a lightweight ERP system.

Figure 2.3.1.1. Frontend Technologies Comparison Table

Selection

After looking at React, React Native, and Angular for MakerSuite's frontend, React was chosen as the most suitable option. It fits the project needs, scales well, has strong community support, and is relatively easy to learn. React Native can create cross-platform mobile and web applications, but it is too complex because of platform-specific modules, managing dependencies, and mobile deployment. Angular was deemed unsuitable because its TypeScript-based architecture comes with a steep learning curve and adds unnecessary complexity for a lightweight ERP system. React helps us develop a user-friendly and responsive web application that works on various devices and operating systems. Its large and active community, extensive documentation, and rich ecosystem of libraries and tools, like Redux for state management, support flexible and maintainable development. Additionally, my previous experience with JavaScript and React components makes the focusing on developing MakerSuite's key features easier instead of dealing with a steep learning curve.

2.3.2. Backend Technologies

Introduction

The backend of MakerSuite is responsible for handling business logic, API integrations, data processing and database interactions. Therefore, the backend technology is required to deliver high performance, scalability and maintainability while supporting the project requirements for a lightweight ERM system.

Comparison Of Technologies

Django

Django is a full-featured, Python-based web application framework designed for rapid development and clean, maintainable code. It includes built-in functionality such as an Object Relational Mapping (ORM), authentication system, admin interface, form handling and security features. It also offers RESTful API support, an active community and clear documentation. Django's strong conventions and modular architecture allow for scalability and consistency, making it suitable for applications that require structured backend logic and reliable data management [18].

Flask

Flask is a Python-based open-source, lightweight WSGI (web server gateway interface) web application framework. It's designed to make getting started quick and easy, with the ability to scale up to complex applications and is suitable for small-scale applications. Its considered a microframework because it's lightweight, simple to use and just provides the essentials for web development without unnecessary complexity. It comes with minimal built-in features, RESTful API support, a built-in development server and an intuitive routing system. Another thing to note about Flask is that it does not come with built-in Object Relational Mapping (ORM) so developers can choose their own, such as SQLAlchemy or simply use raw SQL [19].

Node.js

Node.js is a JavaScript-based runtime environment designed for building fast, scalable network applications. It operates on a non-blocking, event-driven architecture, allowing it to efficiently handle large numbers of simultaneous requests. When used with backend frameworks such as Express.js or Nest.js, Node.js supports the creation of RESTful APIs, middleware-based request handling, and modular application structures. It also benefits from the extensive npm ecosystem, offering a wide range of packages for tasks such as authentication, data validation, and real-time communication. Node.js has a large, active community and strong documentation, and its ability to use JavaScript on both the frontend and backend can simplify development workflows. Its performance and scalability make it suitable for applications that require asynchronous operations or real-time data processing [20].

Technology	Pros	Cons
Django	• Built-in tools such as an ORM, authentication system, admin interface, form handling, and robust security features. • MVC-style architecture encourages clean, maintainable code. • Highly scalable and suitable for larger systems that require consistent logic, stable data management and clear separation of concerns. • Large, active community with long term support releases ensures stability and documentation quality.	• Steeper learning curve compared to more lightweight frameworks like Flask. • ORM Limitations may require writing raw SQL queries for highly complex queries. • Can feel heavy for small web applications that do not require the full suite of Django's built-in features.
Flask	• Lightweight framework that allows developers to build applications with full control over structure, components and dependencies.	• Lacks built-in tools such as an ORM, admin dashboard, authentication system and form handling, requiring

	• Simple, easy to understand routing and request handling enables rapid development/ • Suitable for building microservices where performance, low overhead and modularity are priorities. • Clear documentation and a strong Python community provide support and troubleshooting resources.	developers to integrate and configure these manually. • More setup is required compared to full-stack frameworks like Django, increasing development time for projects with briad requirements. • Not as well-suited for large complex applications that require enterprise-level features,
Node.js	• Allows developers to use JavaScript on both the frontend and backend, simplifying development workflows. • Strong ecosystem through npm provides access to a vast range of third-party packages for tasks such as authentication, validation, real-time features, and database interaction. • Supports building RESTful APIs that require real-time communication, such as messaging or live updates. • Large, active community and extensive documentation provide long-term support and troubleshooting resources.	• Asynchronous programming introduces complexity and can make code harder to read, maintain, and debug. • Increased complexity introduced with npm packages and dependency conflicts, versioning issues and security vulnerabilities. • Lacks built-in features such as an ORM, admin interface or authentication system, requiring additional libraries and setup.

Figure 2.3.2.1. Backend Technologies Comparison Table

Selection

After evaluating Django, Flask, and Node.js as backend options for MakerSuite, Django was selected as the best framework because of its built-in features, organized structure, and ability to scale. Flask is lightweight but needs a lot of manual setup for important functions like authentication, data handling, and security. This extra work would lengthen development time for a system that needs to manage inventory, workflows, and product data across different platforms. Node.js offers high performance and a wide range of tools, but it adds complexity with asynchronous programming and does not include the necessary tools for an ERP-style application. Django comes with an ORM, an authentication system, an admin interface, and security features right out of the box. This ensures that the backend logic is stable and easy to maintain. Its established ecosystem, strong community support, and clear documentation make development easier. My previous experience with Python also helps me learn quickly and focus on implementing MakerSuite's main features efficiently.

2.3.3. Database Technologies

Introduction

The database technology selected for MakerSuite must reliably handle structured data, support efficient querying, and maintain data integrity across the application. As MakerSuite functions as a lightweight ERP system with features such as inventory tracking, supplier management, and order handling, the chosen database must provide strong relational capabilities, scalability, and flexibility.

Comparison Of Technologies

PostgreSQL

PostgreSQL is an advanced, open-source relational database system known for its reliability, strong ACID compliance, and support for complex queries. It offers powerful features such as JSON support, full-text search, indexing options, and extensions that make it suitable for scalable, long-term production use. PostgreSQL is widely used in enterprise systems due to its robustness and performance under heavy workloads [21].

MongoDB

MongoDB is an open-source, document-oriented NoSQL database designed for flexibility and horizontal scalability. Instead of using traditional tables, it stores data as JSON-like documents, allowing dynamic schemas that can easily accommodate varying fields and custom data structures. MongoDB is well-suited for applications that require rapid iteration, semi-structured or unstructured data storage, and distributed data handling. It provides features such as secondary indexes, aggregation pipelines, full-text search, and built-in replication and sharding for scalability [22].

Technology	Pros	Cons
PostgreSQL	• ACID-compliant relational database ensuring strong transactional integrity for inventory, orders and other business-critical operations. • Native support for semi-structured data via jsonb, enabling flexible custom fields (custom units, attributes, variants) while retaining relational guarantees. • Excellent integration with ORMs (including Django's ORM). • Strong security features and role/permission support	• More complexity than embedded/file databases (requires server setup, tuning, and resources). • Slightly steeper learning curve for advanced features (indexes, planner tuning, partitioning strategies) compared to simpler databases.

	suitable for business data protection.	
MongoDB	• Document-oriented model (JSON/BSON) is suitable for flexible, evolving schemas. • Schema flexibility speeds up iteration during early development.	• Not relational by design: joins across collections are less natural and can be less efficient; modelling many-to-many relationships or normalized inventory relations is more complex. • Integration with Django is non-native, requires third-party libraries (e.g., MongoEngine, Djongo).

Figure 2.3.3.1. Database Technologies Comparison Table

Selection

After looking at SQLite, PostgreSQL, and MongoDB as database options for MakerSuite, PostgreSQL was selected as the best choice. It offers reliability, a relational structure, and flexible data storage with jsonb. MongoDB's document-based model makes it harder to manage structured relationships between products, inventory, workflows, and marketplace listings. PostgreSQL enables MakerSuite to store custom fields, user-defined units, and different product attributes while maintaining transactional integrity. This integrity is crucial for inventory management, workflow logging, and precise reporting. Its smooth integration with Django, established ecosystem, strong indexing, and effective query features make sure it can be maintained, performs well, and remains reliable over time. This supports MakerSuite's aim to provide a lightweight but powerful ERP solution tailored to the specific needs of maker businesses.

2.4 Other Research

In addition to evaluating frontend, backend, and database technologies, domain-specific research was conducted to inform the design and development of MakerSuite. This focused on understanding the API's available for existing online marketplace platforms as well as their capabilities and limitations.

2.4.1. Marketplace API's

Marketplace APIs would allow MakerSuite as a web application to access the data and functionality offered by online marketplace platforms. They allow external applications to interact with product listings, inventory, orders and customer communications in a controlled and secure manner. For MakerSuite, integrating with marketplace APIs is essential to deliver the core functionality of centralising business operations across multiple platforms. For the purpose of this prototype, only two marketplace APIs were researched and will be implemented, further work on this project would integrate more marketplace APIs. The available ones include but are not limited to Etsy API, Shopify API, Depop API, Facebook Marketplace API etc. MakerSuite plans to integrate Etsy API and Shopify API.

Etsy API
The Etsy API is a RESTful service that enables external applications to access Etsy shop data. It provides endpoints for managing product listings, retrieving orders, accessing inventory information, and handling customer messages. The API uses OAuth 2.0 for authentication, ensuring secure access to user data.

- **Data Access:** Retrieves detailed information on listings, including title, description, price, quantity, and variations.
- **Inventory Management:** Allows updating of stock levels and product details programmatically.
- **Order Management:** Provides access to order details, status updates, and buyer information.
- **Limitations:** Etsy imposes rate limits on API requests and requires adherence to strict usage policies. Certain actions, such as bulk updates or automated messaging, may have restrictions.

Shopify API
The Shopify API is a REST and GraphQL-based service that allows external applications to interact with Shopify stores. It provides endpoints for products, inventory, orders, customers, and other store resources. Shopify also uses OAuth 2.0 for secure authentication.

- **Data Access:** Supports retrieval and updating of product details, pricing, inventory levels, and variants.
- **Order and Customer Management:** Enables access to orders, fulfilment status, and customer contact information.
- **Integration Capabilities:** Supports webhooks to notify external applications of changes in real-time, allowing for dynamic data synchronization.
- **Limitations:** Rate limits apply to both REST and GraphQL endpoints. Some features, such as custom reporting or multi-channel inventory, may require paid Shopify plans.

When implementing these APIs, key considerations include handling API rate limits, ensuring secure storage of authentication tokens, mapping data models between MakerSuite and external platforms, and accommodating differences in platform-specific features such as product variations, shipping options, and order workflows. Proper error handling and logging mechanisms will be essential to maintain data integrity and system reliability.

2.5 Existing Final Year Projects

The existing final year projects researched are web applications similar in complexity to MakerSuite, “CultureBridge” and “QueueEase”. Although their functionality isn’t similar to MakerSuite, they’re relevant to MakerSuite as they both are full-stack web applications with connection to multiple API’s. The areas studied of these final year projects in my research include their complexity, strengths, weaknesses and the value these learnings bring to MakerSuite.

2.5.1. Culture Bridge, Sara Egan

Overview:
Culture Bridge is a web application that helps TU Dublin students explore Erasmus city options by aggregating cultural, social, and university information for 13 European cities. It integrates APIs (Spotify, Ticketmaster, Europeana) and web scraping to provide insights on music, events, art, and student experiences. A recommender system suggests cities based on user preferences.

Complexity:
The project required integrating multiple data sources, handling API authentication and rate limits, and developing a multi-step recommender system. Designing a modern, accessible UI that presented data consistently across all cities added further complexity.

Strengths:

- Centralises diverse data sources in one platform.
- Personalised recommendations using a structured multi-step algorithm.
- User-friendly, accessible interface with a modern aesthetic.
- Modular architecture allows for future expansion.

Weaknesses:

- Some features were incomplete, including expanded city coverage and advanced recommendation functions.
- Dependence on external APIs and web scraping introduces potential reliability risks.
- Limited testing beyond the initial student groups.

Key Learnings:
From Culture Bridge, several practical insights can inform the development of MakerSuite:

- **Modular Architecture:** Structuring the system into separate, well-defined modules makes it easier to integrate multiple APIs and add new features in the future.
- **API Integration Practices:** Handling authentication, rate limits, and data consistency across multiple external APIs is critical for reliability.
- **User-Centric Design:** Designing an intuitive, accessible interface improves usability, especially when presenting complex data from multiple sources.
- **Data Aggregation and Presentation:** Consolidating information in a clear, centralised dashboard helps users manage multiple workflows efficiently.
- **Planning for Scalability:** Considering future expansion and new data sources during design prevents major refactoring later.

These learnings will help ensure that MakerSuite provides a cohesive, reliable, and user-friendly platform for makers managing listings, inventory, and workflow across multiple marketplaces.

2.5.3. QueueEase, Ian Hipolito

Overview:
QueueEase is a web application designed to improve queuing experiences across industries such as hospitality, healthcare, and government services. It provides virtual queues, real-time updates, QR code-based check-ins, geolocation-enabled service discovery, and an admin dashboard with analytics. The system integrates Django REST APIs, React frontend components, PostgreSQL, Firebase Cloud Messaging, and Mapbox for a full-stack solution.

Complexity:
The project required extensive API design for authentication, queue management, appointments, and feedback collection. Implementing real-time notifications, geolocation services, sentiment analysis, and queue transfers added technical and architectural complexity. Ensuring cross-platform responsiveness and integrating multiple data sources with real-time updates further increased the challenge.

Strengths:

- Full-stack architecture with modular APIs supporting scalability and maintainability.
- Real-time queue management and QR code integration improve user convenience.
- Comprehensive admin dashboard provides actionable insights via analytics and sentiment analysis.
- Cross-platform compatibility ensures consistent user experience across devices.

Weaknesses:

- Some mobile optimisation and deployment issues remained unresolved.
- Map rendering and queue estimation occasionally faced performance limitations.
- Certain advanced features, such as full mobile admin support, were not fully implemented.

Key Learnings:
QueueEase provides valuable lessons for MakerSuite development:

- **Modular API Design:** Well-structured APIs simplify integration of multiple features and external services.
- **Real-Time Data Handling:** Efficiently managing live updates and notifications ensures responsive user experience.
- **User-Centric Features:** Prioritising convenience and usability improves engagement, especially for complex workflows.
- **Analytics Integration:** Collecting and presenting actionable insights can support decision-making and operational efficiency.
- **Cross-Platform Responsiveness:** Designing for multiple device types ensures accessibility and wider adoption.

These learnings are directly applicable to MakerSuite, particularly in integrating multiple marketplace APIs, providing real-time inventory or workflow updates, and delivering an intuitive, responsive interface for makers.

2.6 Conclusions

This literature review section highlighted that the alternative existing solutions to MakerSuite do not address every pain point that this project aims to address as a lightweight ERP web application tailored for makers. However, inspiration can be taken from the features and functionality provided by these existing solutions. Additionally, the review of frontend, backend and database technologies allowed for the final selection of MakerSuite's technology stack: React for the frontend, Django for the backend and PostgreSQL for the database. Overall, this literature review allows for the development of this project to be built upon informed decisions.

3. System Analysis

This section presents the system analysis done for the development of MakerSuite. The purpose of this section is to understand the challenges owners of maker businesses face, identify the requirements for MakerSuite based on these challenges and analyse these requirements to develop an initial systems model.

3.1 System Overview

This subsection aims to define a general, non-technical description of the system from a user perspective. MakerSuite is a lightweight, web-based ERP solution designed specifically for makers who create and sell their own handmade products. It provides a single, integrated platform as a web application accessible through the browser on any device (mobile, tablet, desktop etc.) that helps users manage their business operations and creative workflows in one place.

This system also aims to be flexible and provide support makers with different needs and working environments (e.g. useful both in the studio and in in-person markets).

From a user's perspective, MakerSuite is a central platform where they can organise and streamline workflows that are the most important parts of running their maker business that would otherwise be spread across multiple different channels. This system provides the following functionality for users:

- Allow them to manage product listings across different online marketplaces by providing one central platform where they can create new products, import existing products, manage their product details and perform bulk actions with their product listings.
- Allow them to manage multiple types of inventories and link them together by providing one central platform where they can track the usage, quantity, details of raw materials, tools and finished products.
- Allow them to manage customer communications across different communication channels by providing one central platform that aggregates every message into a single, searchable, easy-to-navigate inbox, enabling templates/ quick replies and allowing them to send real-time synced responses from one platform.
- Allow them to document their creative process while producing products through a product workflow logger where they can record materials used, steps taken, time spend and upload photos.

The system will also provide analytics and reports that add on to the above functionality and provide metrics for their sales trends, material usage and customer engagement.

3.2 Requirements Gathering

This section outlines how the functional and non-functional requirements of MakerSuite were determined, including the identification of key stakeholders and the research activities used to gather data. The aim is to ensure that MakerSuite meets the needs of its intended users.

3.2.1. Stakeholder Identification

The key stakeholders that are involved in the development for MakerSuite are:

- **Maker Business Owners:** Individuals who create and sell their own handmade products. They are the main end-users and their pain points, needs and workflows form the basis of MakerSuite's functional requirements.
- **Potential Maker Business Owners:** These are individuals who wish to create and sell their handmade products. They're a consideration because MakerSuite aims to reduce barriers to entry and provide support for not only established maker business owners, but those who otherwise would have not tried to start their own maker business.
- **Customers Of Maker Businesses:** These are indirect stakeholders, their expectations for maker businesses influence MakerSuite's functional requirements.
- **The Developer:** This is the individual (me!) responsible for designing, implementing and maintaining the system. Technical feasibility, time constraints and resource limitations also influence MakerSuite's functional requirements.
- **Providers Of External API's:** These platforms that provide external API's shape what features are feasible and influence MakerSuite's functional requirements.

3.2.2. Research Activities

A combination of secondary and primary research methods was used to collect requirements for MakerSuite. For secondary research, a literature review of existing solutions, along with relevant articles, blogs, and vlogs from makers, was conducted. Insights from this research informed the project background described in Section 1.1. For primary research, an online survey was selected as the primary method to gather structured feedback from a wide range of maker business owners and potential makers.

This subsection will emphasise **the online survey as the main source of research gathering** used to inform the functional and non-functional requirements of MakerSuite because surveys provide both quantitative data, which can guide feature prioritisation, and qualitative insights into workflows, pain points, and user expectations.

Survey Participants and Recruitment:

To ensure relevant insights, the survey targeted current and aspiring maker business owners. Participants were recruited through:

- Social media groups dedicated to makers and artisans (e.g., Instagram communities)
- Personal contacts and networks of individuals involved in creative businesses

A total of **19 participants** completed the survey, although that's a small number, they're the key demographic and target users for MakerSuite so they can act as a representative sample of the target user group.

Survey Design

The survey (found in Appendix A) was designed to collect insights into the needs and experiences of maker business owners, Google Forms was used as the platform to create and distribute the survey due to its accessibility, ease of use and ability to handle both qualitative and quantitative data. The survey consisted of six sections:

The survey consisted of **six sections**:

- Participant Information
    - Collected basic contact information, including email addresses (optional).
    - Participants could enter their email for a prize draw as an incentive to encourage participation.
- Business Overview
    - Questions about the participant's current or planned maker business.
    - Included business type, products made, sales channels, and size.
- Pain Points and Challenges
    - Explored difficulties faced in running a maker business, including managing listings, inventory, workflows, and customer communications.
    - Explored existing solutions makers use and their struggles with them.
- Feature Suggestions
    - Participants were asked questions specifically for the development of product listings, inventory management, workflow logging, customer communications, and analytics feaures that would best help them with their day-to-day operations.
- Accessibility and Sustainability
    - Gathered insights on accessibility needs, such as interface preferences, assistive technologies used, and potential barriers to using software tools as well as sustainability needs.
- Additional Feedback
    - Open-ended questions allowed participants to provide any further suggestions or comments.

Survey Results

The survey results (found in Appendix B) provide a clear validation of the core features and objectives of MakerSuite while offering deeper insight into the real-world workflows and challenges faced by maker business owners. Responses highlighted recurring pain points in managing multiple platforms, tracking inventory, handling customer communications, and documenting creative workflows. They also underscored the importance of accessibility and flexibility across devices. Collectively, these insights not only confirm the relevance of MakerSuite's proposed features but also reveal gaps in existing tools that the application can address, ensuring it is aligned with the specific needs of its target users.

The survey results include the following key findings:

- Cross-Platform Product Listing Management
    - Many participants found managing separate dashboards across platforms time-consuming. Users expressed strong interest in being able to create, update and sync listings from one place.
- Multiple Inventory Tracking
    - Makers highlighted difficulty tracking both finished products and the raw materials used to create them. Survey responses showed a clear need for:
        - Material-level tracking (e.g. beads, yarn, clay, packaging)
        - Linking materials to specific products
        - Tracking of packaging materials
        - Tracking of leftover materials
        - Alerts for low-stock raw materials

- Centralised Customer Communication
    - Participants described feeling overwhelmed managing messages across Instagram, email, Etsy, TikTok, and more. A unified inbox was identified as highly desirable.
- Workflow Logging
    - Makers wanted a way to document the steps, materials, tools, photos and time involved in creating each product, both for reproducibility and pricing accuracy.
- Analytics and Insights
    - Users requested simple analytics showing sales trends, stock levels, material usage and popular products to help with sustainability, planning and decision making.
- Accessibility and Device Flexibility
    - Survey data showed makers frequently work on different devices depending on context (e.g. mobile during markets, desktop during product uploads). This reinforces the need for a responsive, accessible web interface.

3.3 Requirements Analysis

The purpose of this section is to interpret the findings from the requirements gathering activities and translate them into meaningful insights that inform the initial system specification. While Section 3.2 focused on collecting data from surveys, literature, and existing tools, this section focuses on analysing that information to identify patterns, prioritise user needs, and determine what functionality MakerSuite must support.

3.3.1 Analysis of User Needs

The survey results revealed several recurring pain points and behaviours among makers. Users frequently switch between multiple dashboards, spreadsheets, and social media apps, resulting in fragmented workflows. Many described listing management, material tracking, and customer communication as time-consuming tasks that often overshadow the creative work. Makers also reported that available tools are either too generic or too complex for their needs, making it difficult to maintain consistency in creating products or tracking raw materials. Customer communication across multiple channels was highlighted as overwhelming, with makers expressing the need for a way to avoid missed messages. Overall, these findings demonstrate the need for an integrated solution tailored to the unique operations of handmade product businesses.

3.3.2 Prioritisation of Functional Areas

The collected data was reviewed to determine which features should be considered essential for a minimum viable product and which could be treated as secondary enhancements. Prioritisation was based on frequency of mentions in survey responses, impact on daily workflows, and feasibility within the project timeframe. This analysis identified centralised listing management, multi-inventory tracking, customer communication aggregation, basic analytics, and a responsive and accessible interface as primary requirements, while workflow logging, extended analytics, automated low-stock alerts, and communication templates were considered secondary.

3.3.3 Mapping Pain Points to System Capabilities

Each major pain point identified through the research was translated into a corresponding capability for the system. Difficulties managing multiple marketplace dashboards informed the need for centralised listing management. Manual tracking of materials highlighted the need for a dual inventory management system. Challenges in reproducing products consistently supported the inclusion of workflow logging with material and tool tracking. Overwhelming customer messages justified a unified inbox. Lack of visibility into trends and stock levels reinforced the need for simple analytics. Finally, widespread use of tools not designed for makers emphasised the importance of a flexible, craft-agnostic interface. This mapping ensured that system requirements directly addressed real user needs.

3.3.4 Non-Functional Considerations

The research also highlighted several non-functional needs. Makers frequently use different devices depending on context, reinforcing the need for a responsive, accessible interface. Performance expectations were evident from user frustrations with slow tools, indicating that screens should load quickly and interactions should feel seamless. Marketplace integrations introduce security considerations, particularly regarding safe storage of API tokens and secure authentication. Cost-efficiency was also important due to the small-scale nature of many maker businesses, indicating that the system architecture should remain lightweight and affordable.

3.3.5 Summary of Analysis

The analysis confirmed strong alignment between the pain points expressed by makers and the features proposed for MakerSuite. It validated the project's emphasis on integration, workflow flexibility, ease of use, and support for handmade production processes. These insights informed the functional and non-functional requirements presented in Section 3.4 and provided a clear foundation for system design.

3.4. Initial System Specification

This section defines the initial functional and non-functional requirements for MakerSuite, informed by both primary research (survey results of section 3.2.) and secondary research (alternative existing solutions of section 2.2.). It translates the insights gained from makers' workflows, pain points, and feature preferences into a structured specification that guides the design and development of the platform. Additionally, this section presents the requirements diagram that captures how user needs, system functions, and data persistence requirements interact.

3.4.1. Functional Requirements

These have been divided into primary and secondary functional requirements for the purpose of developing MakerSuite. Each functional requirement has been numbered and labelled as part of a natural language requirement specification.

Primary Functional Requirements

- **FR1. Cross-Platform Listing Management**
  The system must allow users to create, edit, import and manage product listings from multiple marketplaces through a central interface.
- **FR2. Multi-Inventory Tracking**
  The system must track raw materials, finished products, tools and packaging while allowing users to link materials to products
- **FR3. Centralised Customer Communications**
  The system should aggregate incoming messages from different platforms into a single inbox-style interface.
- **FR5. Analytics Dashboard**
  The system must provide summary insights such as sales trends, stock levels and material usage to support planning and sustainability.
- **FR6. Responsive and Accessible User Interface**
  The system must support mobile, tablet and desktop use, and follow accessibility best practices (e.g., WCAG AA guidelines).
- **FR7. User Accounts & Authentication**
  The system must support secure login and management of connected marketplace accounts.

Secondary Functional Requirements

- **FR8. Workflow Logging**
  The system must allow users to document process steps, time, materials, tools and upload photos to support reproducibility and pricing.
- **FR9. Extended Analytics**
  The system could provide more detailed analytics and reports on general sales trends.
- **FR10. Automated Material Deduction & Low-Stock Alerts**
  When a product is created or sold, the corresponding materials should be automatically deducted and low-stock notifications generated.
- **FR11. Quick Replies/ Bulk Actions For Customer Communications**
  Extend the single inbox to allow users to send quick replies based on templates and conduct bulk actions for customer communications.

3.4.2. Non-Functional Requirements

- **NFR1. Performance**
  Core screens should load within 2–3 seconds on an average internet connection.
- **NFR2. Security**
  The system must use secure authentication, encrypted communication and safe storage of marketplace API tokens.
- **NFR3. Accessibility**
  The interface should adhere to WCAG AA guidelines for colour contrast, keyboard navigation and structural markup.
- **NFR4. Reliability & Scalability**
  The system should be able to handle asynchronous tasks, API rate limits and background sync processes without user disruption.
- **NFR5. Maintainability**
  The system should be modular and follow best practices that enable future updates, testing and debugging.

- **NFR6. Cost Efficiency**
  Given the target user demographic (small maker businesses), system architecture must minimise hosting and operating costs.

3.4.3. Requirements Diagram

To organise and visualise the system requirements, the functional and non-functional requirements identified for MakerSuite were translated into a requirements diagram. This diagram maps user interactions, core system functionality, and data persistence relationships in a structured format. It serves as a blueprint for the design and implementation phases by clearly illustrating:

- **User Requirements**: the features and capabilities that end-users expect from MakerSuite, based on survey insights and research.
- **Functional Requirements**: the actions the system must support, such as cross-platform listing management, inventory tracking, workflow logging, and customer communications.
- **Data Persistence Requirements**: the storage and management of data related to products, materials, users, orders, and communications.

This diagram provides an overview of how users, system functionality, and data interact, supporting both development planning and stakeholder understanding.

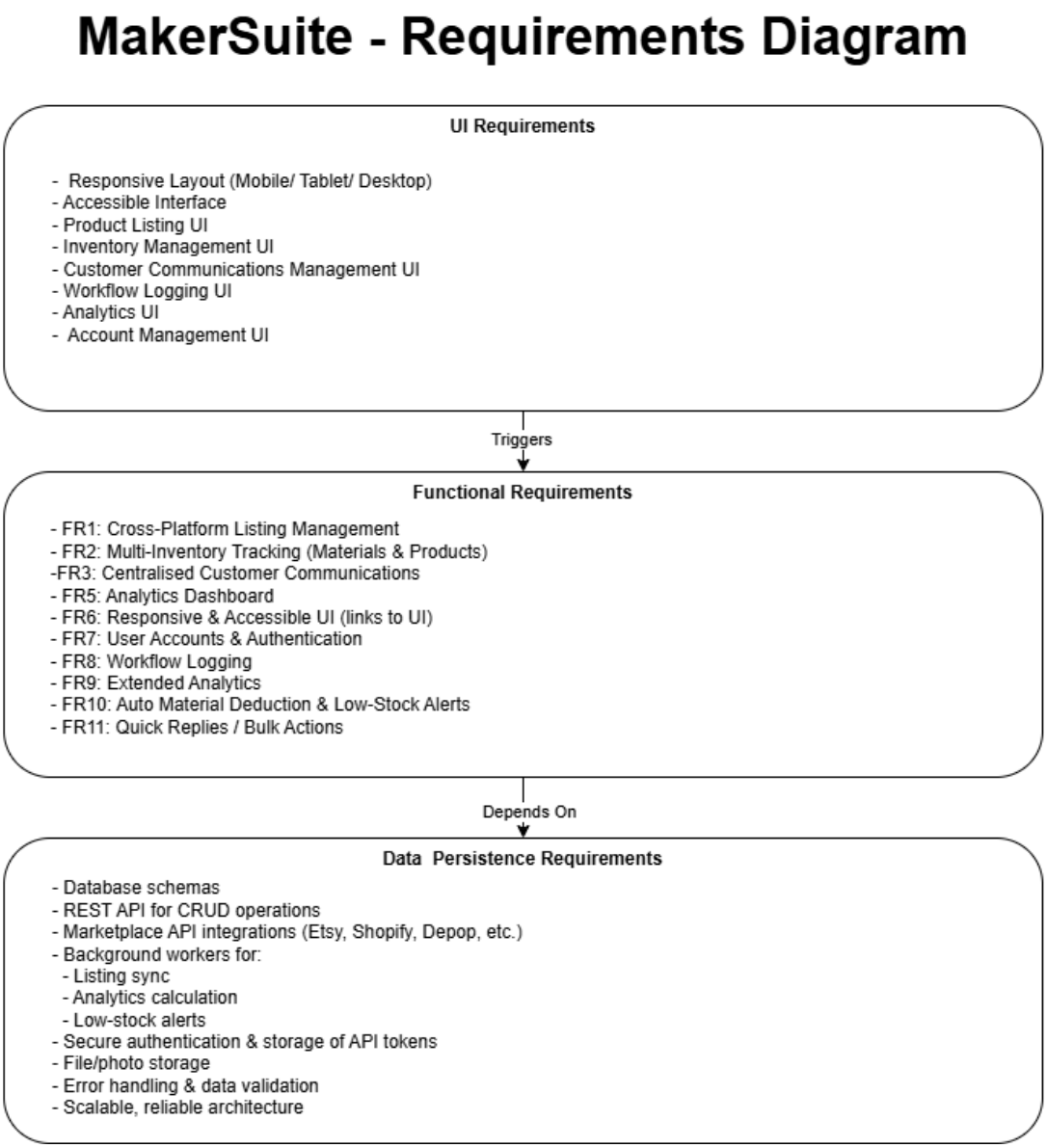

Figure 3.5.1. Diagram Of MakerSuite's Requirements

3.5. Conclusions

The system analysis identified the key functional and non-functional requirements for MakerSuite, informed by both primary user research and secondary evaluation of existing solutions. These insights were translated into a comprehensive requirements diagram that captures the relationships between user needs, system functionality, and data persistence. This diagram provides a clear foundation for the system design phase, ensuring that the development of MakerSuite remains aligned with real maker workflows and the project's intended objectives.

4. System Design

4.1 Introduction

The purpose of this section is to translate the requirements and system specification identified in Section 3 into a detailed design that will guide the implementation of MakerSuite.

The system design must consider the specific limitations of the project. These limitations include the tight project timeline, with a prototype required before March/April, and the fact that the entire system is being developed independently. Therefore, the system design must be high-level and flexible to support those limitations.

The following subsections will:

- Define an appropriate software methodology to develop the system.
- Define an overview of the system by designing the logical architecture.
- Produce a collection of design artefacts (UML diagrams and ER diagrams).

4.2 Software Methodology

I have chosen to use an **Agile-inspired approach** combined with **Feature-Driven Development**. This method supports short, iterative cycles that allow each major feature of MakerSuite (such as listings, inventory, communications and analytics) to be designed, implemented and tested in manageable increments. It offers the flexibility needed to adapt requirements as the system evolves, while ensuring steady progress toward a functional prototype within the project constraints.

4.3 Overview of System

I've chosen to illustrate the overview of the system using a logical architecture diagram that defines the MakerSuite as a modular, three-tier system consisting of a front-end layer, back-end layer and data layer as well as external services. This structure separates concerns between user interaction, business logic and data persistence. Below is the diagram of the logical architecture followed by a breakdown of the system:

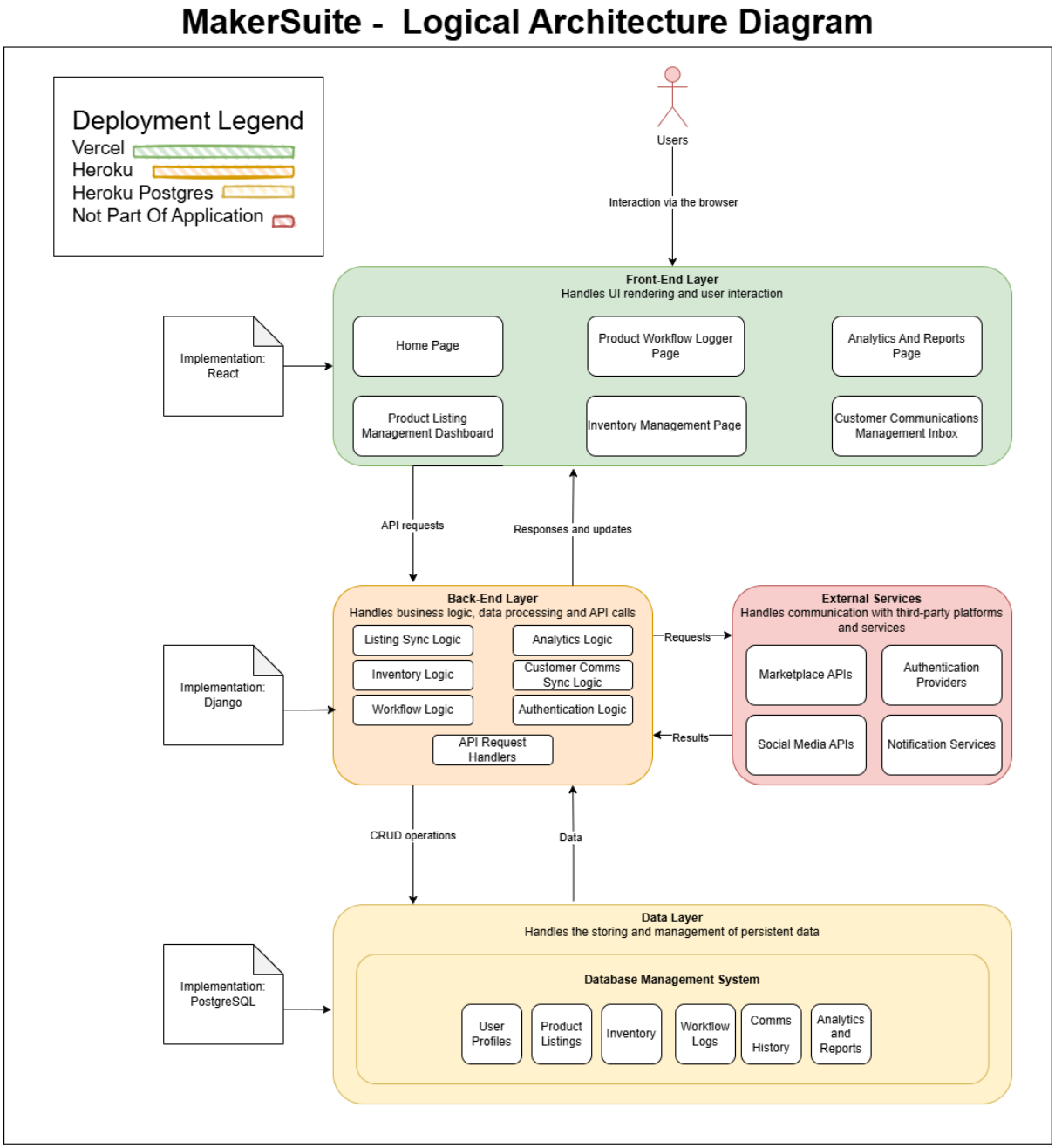

Figure 4.3.1. MakerSuite's Logical Architecture Diagram

The Front-End Layer is responsible for rendering the user interface and managing user interaction through the browser. It includes the primary application pages such as the Product Listing Dashboard, Inventory Management Page, Workflow Logger, Analytics & Reports Page, and Customer Communications Inbox. This layer is intended to be implemented in React and deployed on Vercel, the easiest way to deploy the framework using single-command deployment, automatic-built in SSL and global CDN [23].

The Back-End Layer handles MakerSuite's core business logic, API request handling, and data processing operations. It contains modules for listing synchronisation, inventory tracking, customer communication synchronisation, workflow logging, analytics computation, and authentication. This layer will be deployed on Heroku, as deploying Python apps s simplified with Heroku [24].

The Data Layer manages all persistent data required by the application, including user profiles, product listings, inventory quantities, workflow logs, communication history, and analytics outputs. PostgreSQL is selected as the database management system due to its relational model, support for structured querying, and suitability for MakerSuite's need for reliable data consistency. This layer is deployed on Heroku Postgres, which simplifies deployment and is easily integrated with the back-end being hosted on Heroku .

The system also integrates with external services, such as marketplace APIs, authentication providers, social media APIs, and notification services. These services enable multi-platform listing synchronisation, secure user login, social communication, and automated notifications.

This logical architecture, as well as the implementation and deployment of it, provides a realistic foundation for a prototype while remaining scalable for future development stages.

4.4 Design Of The System

This subsection translates the requirements identified earlier in Section 3.4. into high-level design artefacts that will guide the initial implementation of MakerSuite. The aim here is not to finalise every design decision but produce usable blueprints, the design artefacts produced will act as a guide for prototyping but will not definitely reflect the final system for several reasons: external API constraints, evolving requirements and implementation discoveries. The design artefacts in this subsection will include:

- High-level use case diagrams
- Sequence diagrams
- Database design

4.4.1. High-Level Use Case Diagrams

The following PlantUML diagrams are early, high-level models intended to guide the initial development of MakerSuite, particularly during prototyping. They do not represent the final system, as future refinements may arise due to external API constraints, user feedback or time/resource limitations.

The System Overview Use Case Diagram illustrates all core features of MakerSuite as packages, and with the entities involved as actors. Each package contains a single high-level use case representing the functionality of that module, detailed behaviours of each feature are modelled in separate, module-specific use case diagrams. Each feature is labelled according to the functional requirements defined in Section 3.4.

System Overview Use Case Diagram

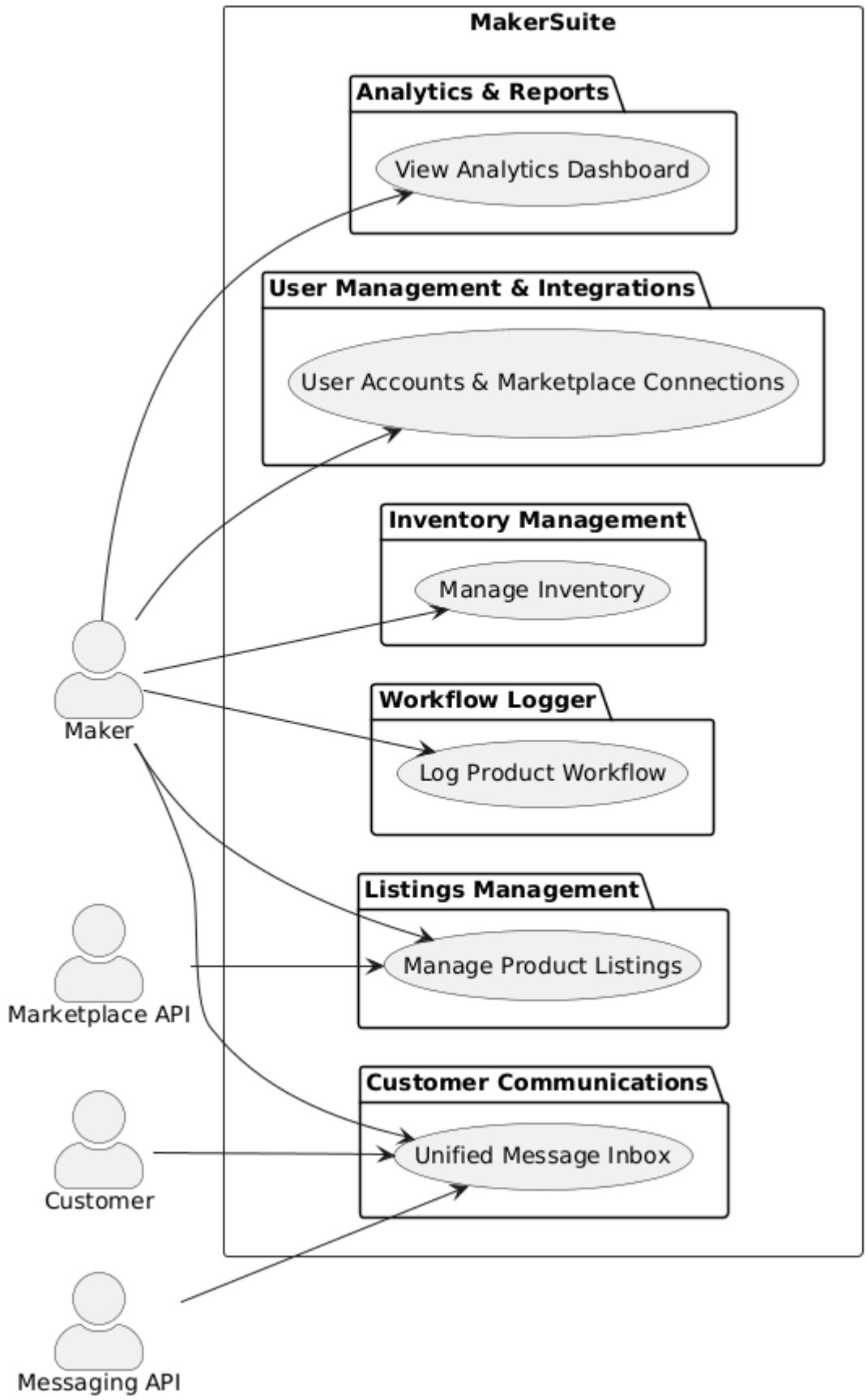

Figure 4.4.1.1. System Overview Use-Case Diagram

Listings Management Use-Case Diagram (FR1)

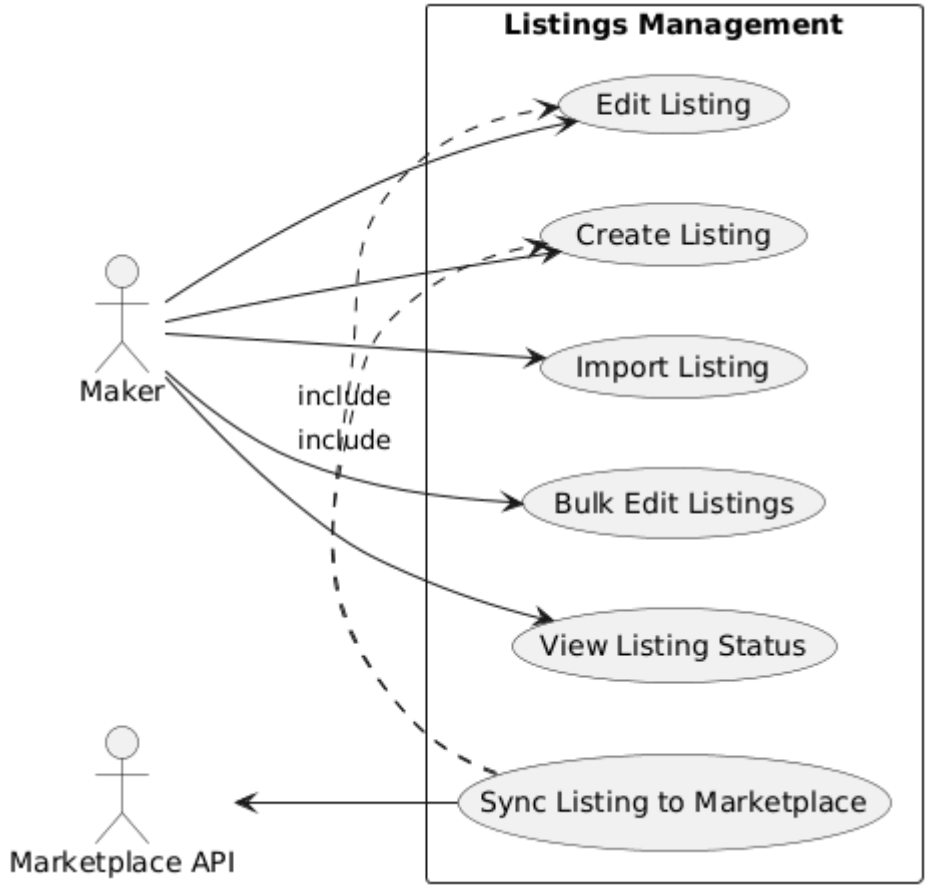

Figure 4.4.1.2. Listings Management Use-Case Diagram (FR1)

Inventory Management Use-Case Diagram (FR2, FR10)

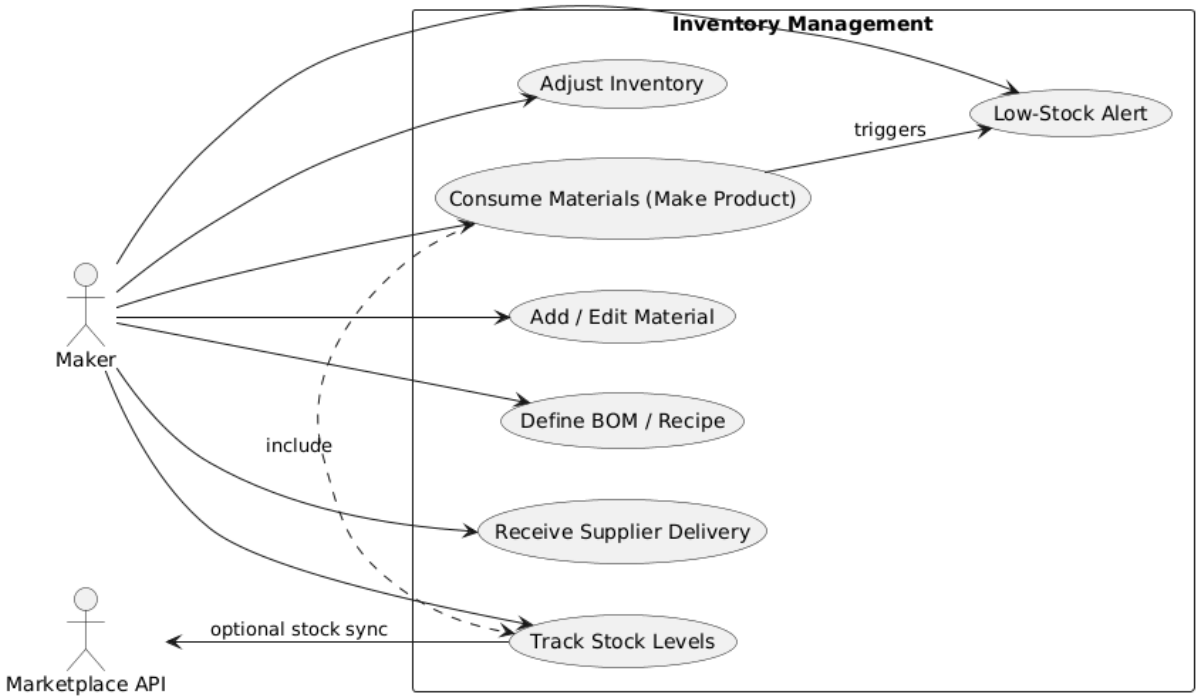

Figure 4.4.1.3. Inventory Management Use-Case Diagram (FR2, FR10)

Customer Communications Management Use-Case Diagram (FR3, FR11)

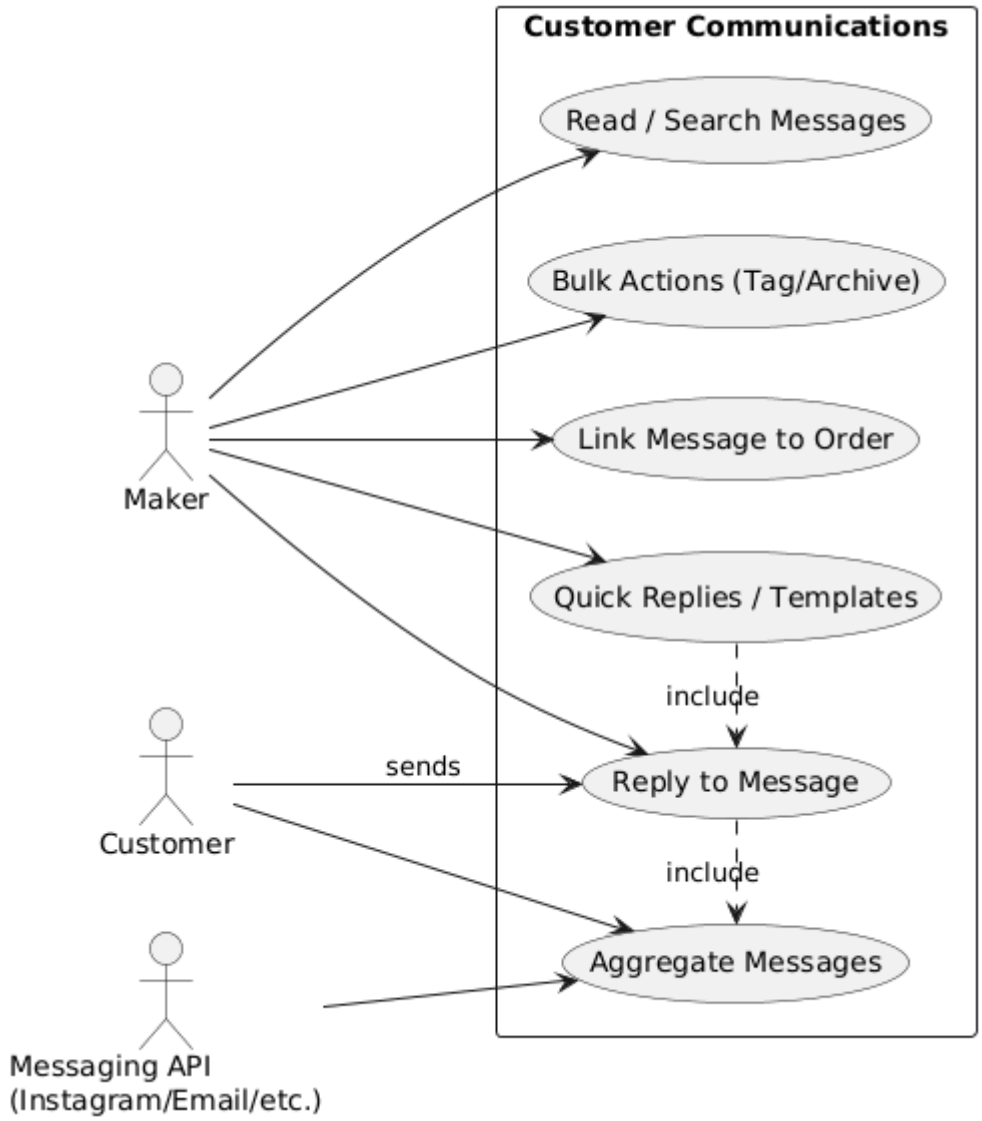

Figure 4.4.1.4. Customer Communications Management Use-Case Diagram (FR3, FR11)

Workflow Logger Use-Case Diagram (FR8)

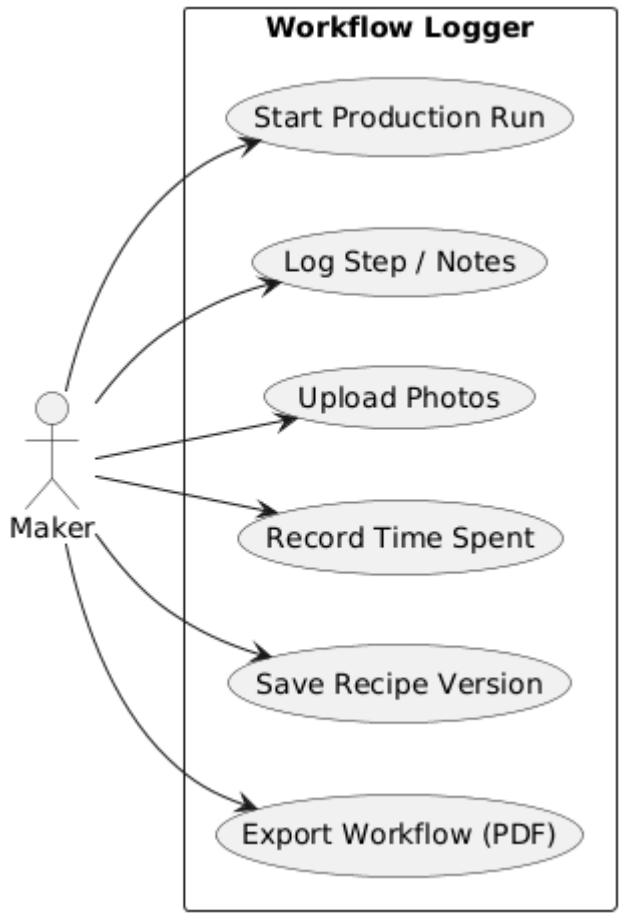

Figure 4.4.1.5. Workflow Logger Use-Case Diagram (FR3, FR11)

Analytics And Reporting Use-Case Diagram (FR3, FR11)

Figure 4.4.1.6. Analytics And Reporting Use-Case Diagram (FR5, FR9)

User Management And Integrations Use-Case Diagram (FR7, Supporting NFRs)

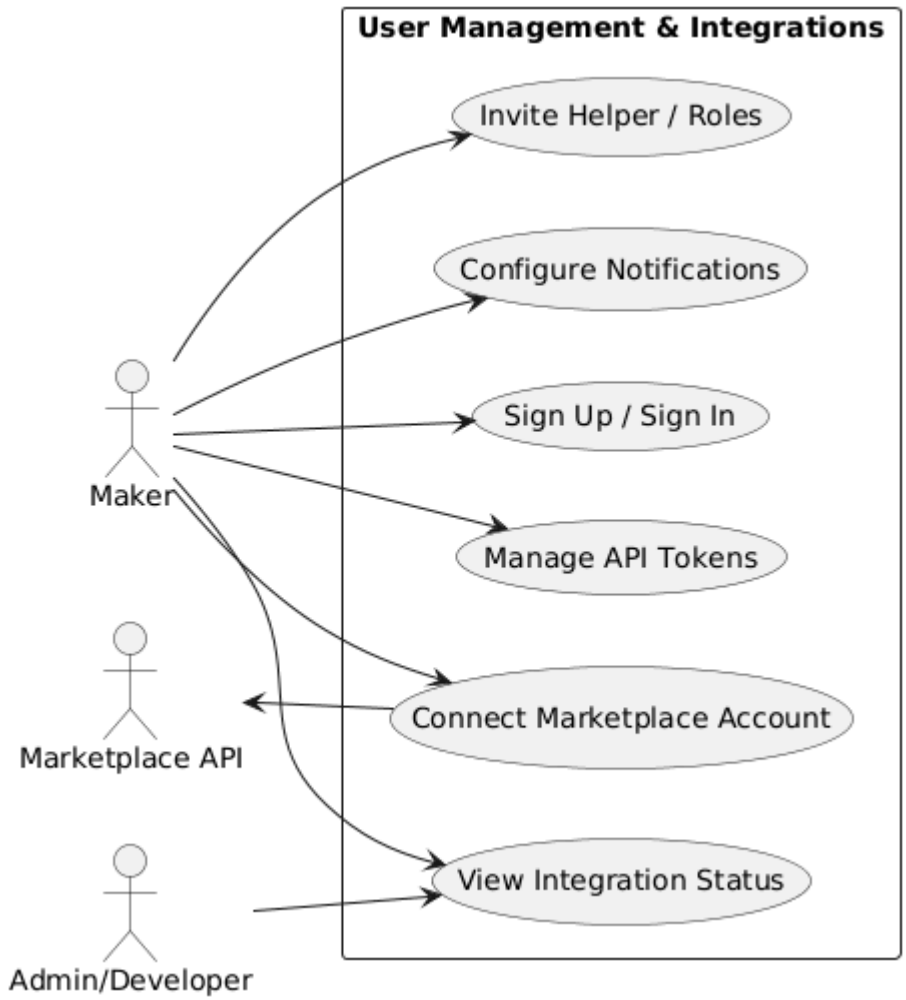

Figure 4.4.1.7. User Management And Integrations Use-Case Diagram (FR5, FR9)

4.4.2. High-Level Sequence Diagrams

The following PlantUML sequence diagrams are early, high-level models that illustrate how the system responds to user actions, communicates with databases and APIs, and executes core functionality such as product listing, inventory management, and customer communications. Sequence diagrams were initially only created for these core features.

Listing Management Sequence Diagram (FR1)

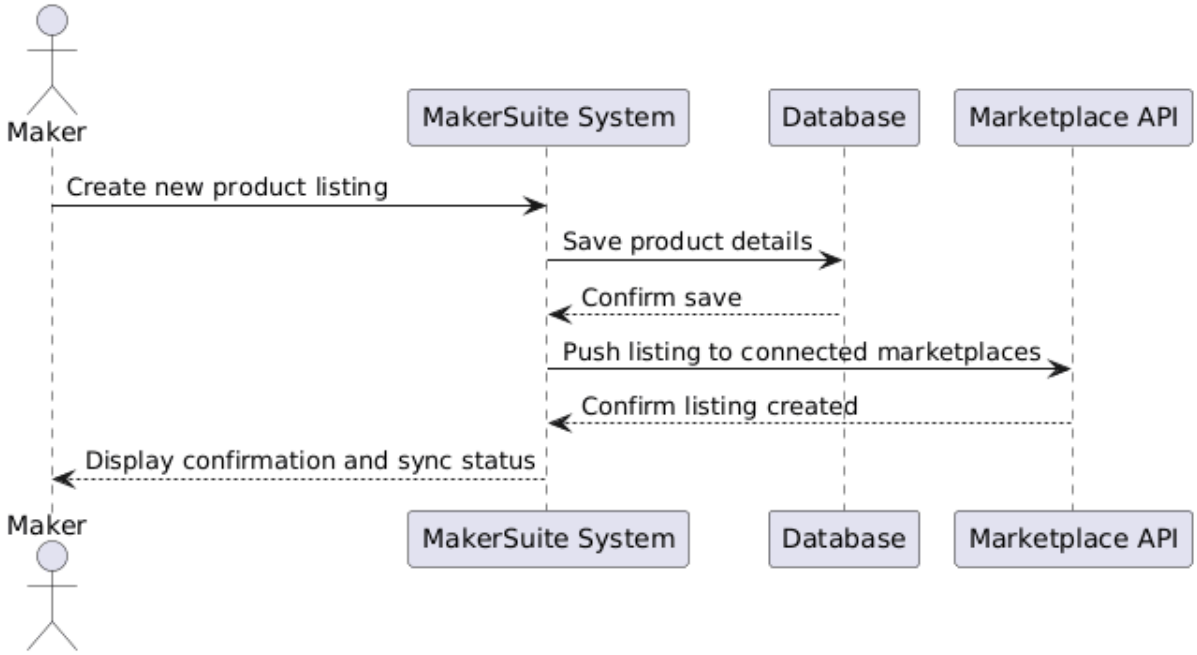

Figure 4.4.2.1 Listing Management Sequence Diagram

Inventory Management Sequence Diagram

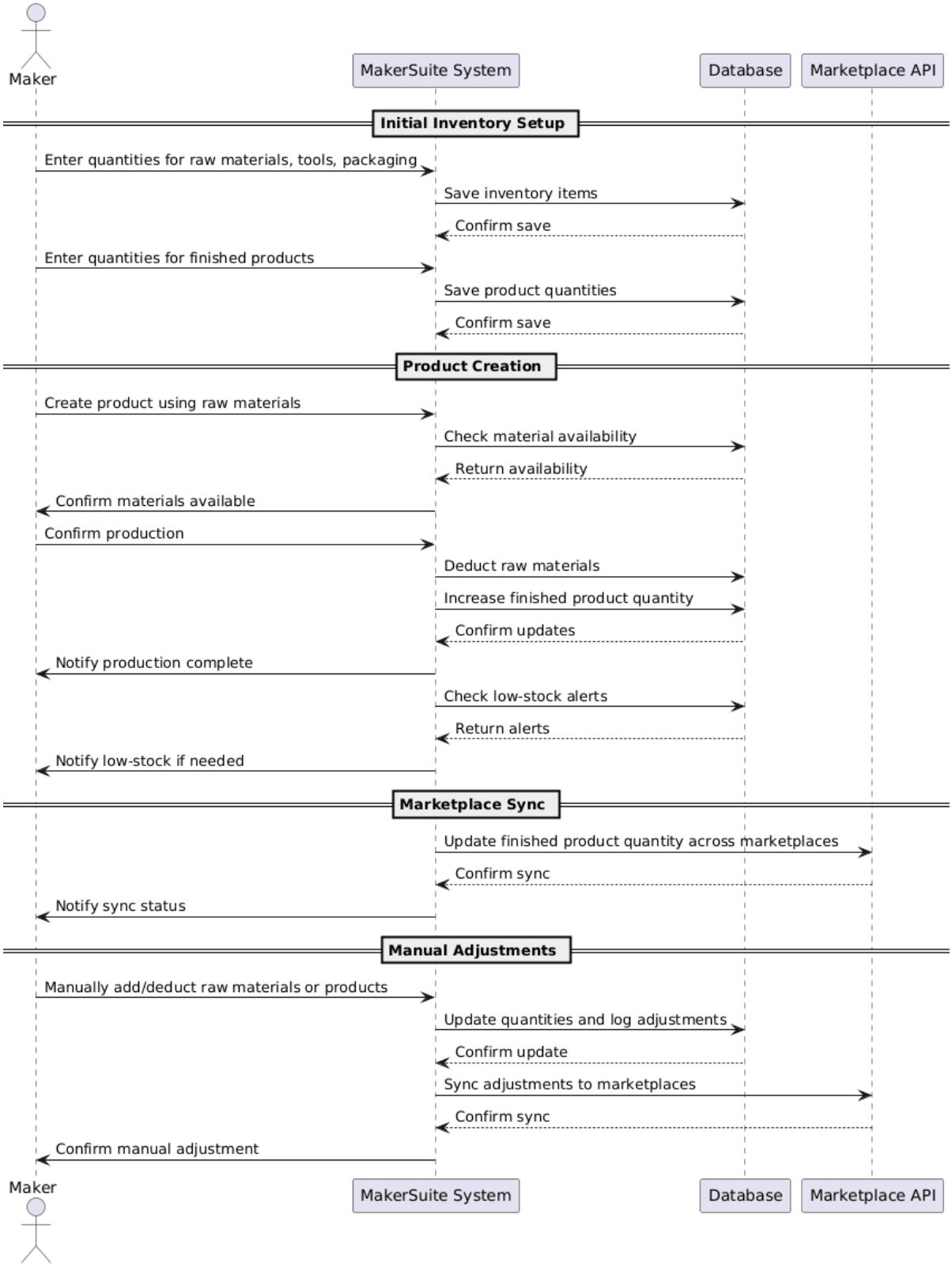

Figure 4.4.2.2. Inventory Management Sequence Diagram

Customer Communications Management Sequence Diagram

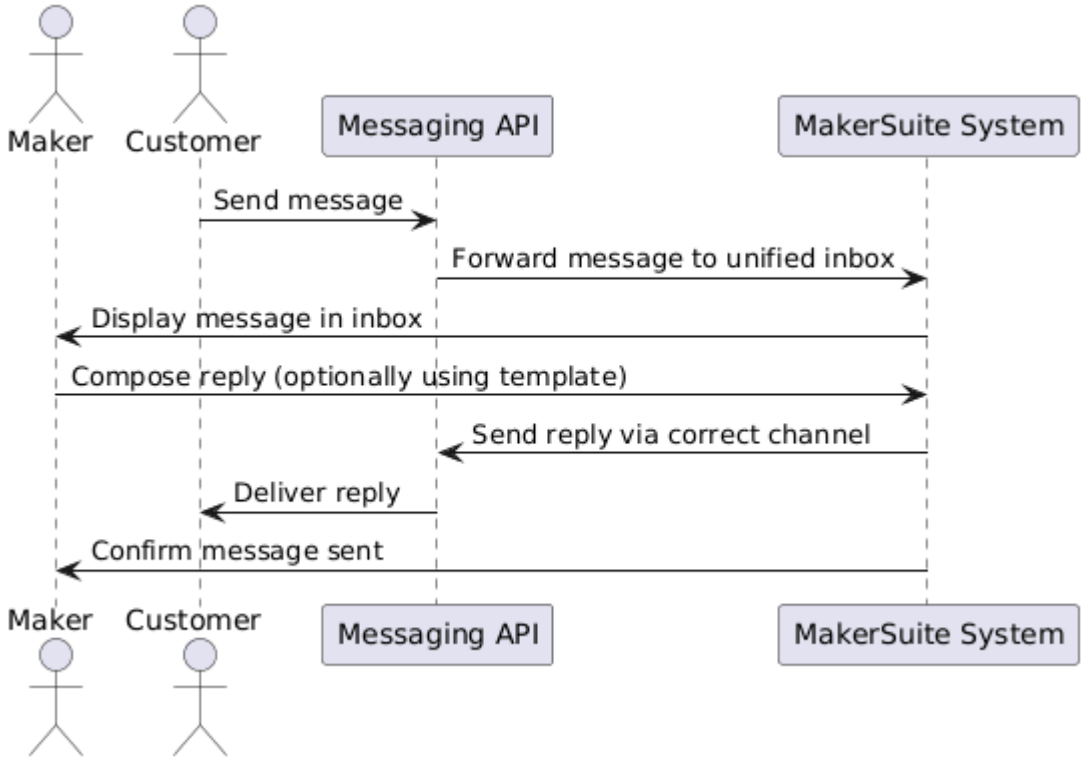

Figure 4.4.2.3 Customer Communications Management Sequence Diagram

4.4.3. Database Design (ER Diagrams)

The following database design and ER diagram represent the initial data model for MakerSuite. This early design supports the core functional requirements defined in Section 3.4 and has been structured to suit PostgreSQL, the selected relational database for the system. The schema emphasises flexibility, normalisation and minimal data duplication, ensuring that tables are scalable and maintainable as the system grows. Relationships have been designed to capture the real-world interactions between products, listings, inventory items, customer messages and external marketplace integrations. As with previous design artefacts, this data model is expected to evolve throughout development as API constraints, performance considerations and user feedback further shape MakerSuite's requirements.

MakerSuite ER Diagram

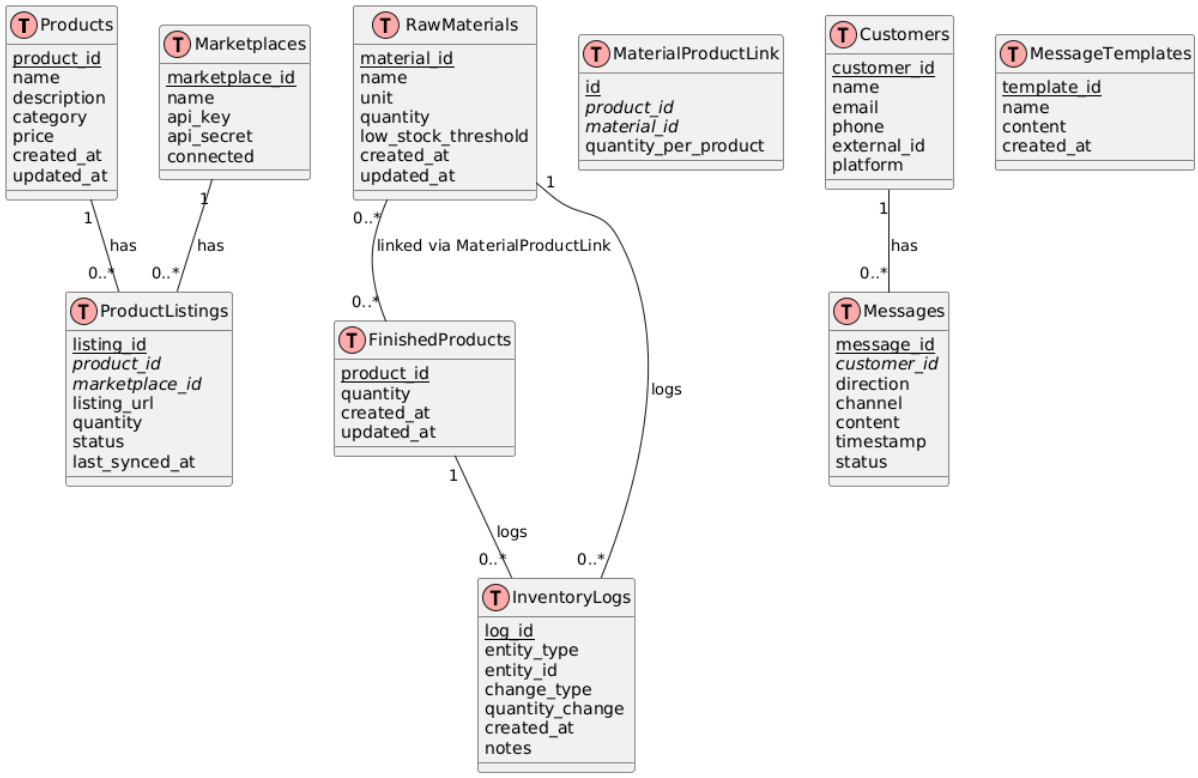

Figure 4.4.1. MakerSuite ER Diagram

4.5. Conclusions

This section translated the requirements identified in Section 3 into a structured, implementation-ready system design for MakerSuite. An agile-inspired, feature-driven-development software methodology was selected to develop the system to reflect the project's constraints, prioritising flexibility and iterative refinement while providing a MVP at each iteration. The logical architecture established the overall structure of the system as well as implementation and deployment options. The UML diagrams and database designs modelled how the system's core features behave and how the key data entities relate to one another

As this project operates under a tight timeline and is developed independently, the design remains intentionally high-level and adaptable. These design outputs form a solid foundation for the upcoming implementation phase, while still allowing for refinement in response to technical constraints, integration challenges and user feedback gathered during prototyping.

5. Testing and Evaluation

5.1 Introduction

This section outlines the testing and evaluation approaches for the major components of the system (as defined earlier in Section 4's logical architecture diagram). This testing and evaluation is crucial to ensure that MakerSuite meets its functional requirements, performs reliably and provides a usable experience for makers. No testing and evaluation has been conducted yet but once feature-driven-development begins, both automated and manual testing methods are to be used on the major component in the logical architecture.

5.2 Plan for Testing

Testing will be carried out across the major components of the system using a combination of unit tests, UI tests, API tests, and integration checks. This will be done using the below methods:

- **Front-End Testing:**
  Selenium/ Robot Framework will be used to automate UI testing, validating that page components, forms, interactions, and workflows behave as expected in the browser. Additional manual exploratory testing will be performed to identify visual or usability issues.
- **Back-End and API Testing:**
  Bruno will be used to test API endpoints, ensuring correct request/response handling, validation, and error behaviour. Each feature module (inventory, listing sync, workflow logging, analytics, and communications) will have targeted API tests.
- **Unit Testing:**
  Core logic for the back-end layer, such as inventory deduction, listing synchronisation, and analytics calculations, will be tested with lightweight unit tests to verify correctness and limit regressions during iterative development.
- **Integration Testing:**
  Combined workflows (e.g., marketplace sync + inventory update) will be validated to confirm that the front-end, back-end and database interact correctly.

A structured test plan will be created using test case tables covering objectives, preconditions, inputs, expected outputs, and pass/fail criteria for each major feature.

5.3 Plan for Evaluation

Evaluation will focus on assessing usability, functionality, and suitability of MakerSuite for its intended users, this will be conducted by the below methods.

- **User Acceptance Testing (UAT):**
  A user group of makers will evaluate early prototypes to determine whether workflows are intuitive and meet their practical needs. Feedback will be used to refine UI layouts, feature priorities, and overall system behaviour.
- **Performance and Reliability Evaluation:**
  API response times, error handling, and data consistency will be monitored to ensure the system behaves predictably under typical usage.
- **Usability Evaluation:**
  Observational feedback and informal interviews or surveys will be used to assess

whether users can complete tasks efficiently, understand the UI, and navigate pages with minimal guidance.

Each evaluation method contributes to validating that MakerSuite functions effectively and aligns with its goal of supporting makers.

5.4 Conclusions

This section established a structured and practical approach for testing and evaluating MakerSuite throughout development. Testing coverage is ensured across all major system components through automated UI testing with Selenium/Robot Framework, API validation using Bruno, and targeted unit and integration tests for backend logic. Evaluation activities, including UAT, performance assessments and usability reviews, ensure that MakerSuite not only functions correctly but also meet the real needs and expectations of its intended users.

6. System Prototype

6.1 Introduction

As this is an interim report, the MakerSuite prototype has not yet been developed. This section outlines the planned approach for building the prototype based on the logical architecture defined in Section 4. This prototype will be developed iteratively using the Agile-inspired, feature-driven methodology described earlier, with the goal of producing a system that meets the requirements defined in Section 3 by integrating each technology chosen in Section 2.

The prototype will be implemented using the selected technology stack: **React** for the frontend, **Django** for the backend and **PostgreSQL** as the database management system. Additional frameworks and tools, such as **Selenium**, **Robot Framework**, and **Bruno**, will support testing during development of the final prototype.

6.2 Prototype Development

Prototype development will begin by implementing the essential modules identified in the logical architecture:

- **Frontend Prototype (React):**
  Initial UI layouts for the Listing Dashboard, Inventory Page, Workflow Logger and Communications Inbox will be created to establish navigation flow and usability foundations.
- **Backend Prototype (Django):**
  Basic API endpoints will be implemented to support CRUD operations for listings, inventory, workflow logs, and user communications. Authentication and external API connectors will be added in later iterations.
- **Database Prototype (PostgreSQL):**
  The ER models designed in Section 4 will be converted into database tables. Early development will focus on core entities such as listings, materials, inventory items and workflow logs.

At this stage of the report, these components are planned but not yet developed.

6.3 Results

Since prototype implementation has not yet begun, no results or test outputs are available at this stage. Once the prototype is developed, the following planned tests as defined in Section 5 will be carried out:

- Automated UI tests using Selenium/Robot Framework
- API tests using Bruno
- Unit and integration tests across backend modules

Results will be documented in the final report.

6.4 Evaluation

Evaluation will take place once the prototype is functional. The following evaluation methods will be applied:

- **User Acceptance Testing (UAT):** Makers will assess whether key workflows meet their needs.
- **Performance Evaluation:** Response times and sync operations will be assessed against non-functional requirements.
- **Usability Evaluation:** Observational testing and UI feedback sessions will be used to analyse navigability and accessibility.

Outcomes of these evaluation activities will be presented in the final report.

6.5 Conclusions

This section outlined the planned approach for developing, testing and evaluating the MakerSuite prototype. While no prototype has been implemented at the interim stage, the framework, development plan and testing methodology are now clearly defined, enabling the next phase of work to focus on building and validating the core system.

7. Issues and Future Work

7.1 Introduction

This section outlines the main issues, risks and uncertainties identified so far in the development of MakerSuite, based on the current state of the project. It also presents future work required to complete the system and a high-level project plan showing how development will progress during the implementation phase.

7.2 Issues and Risks

As MakerSuite is still in the early stages of development, there haven't yet but any different outcomes compared to the expected outcomes but several challenges and uncertainties have been identified that may impact the progress and final delivery of the system:

- **Learning Curve Across the Full Stack:**
  MakerSuite requires the integration of React, Django and PostgreSQL, alongside authentication, messaging pipelines and background syncing. The learning curve involved in combining these technologies into a functional full-stack system may be an issue, particularly as the project is developed independently.
- **Integration of Multiple External APIs:**
  A key requirement of MakerSuite is communication with third-party platforms (e.g., marketplace APIs, messaging APIs). These services often impose constraints such as strict rate limits, complex OAuth flows, incomplete documentation and changing API policies. Each of these introduces uncertainty in the design and may require significant adaptation during implementation.
- **Time Constraints of the Project:**
  The project timeline is tight, with limited development time before the expected prototype deadline. Since the system requires multiple complex features (product syncing, inventory automation, workflow logging, unified communications), there is risk that not all planned functionalities can be completed to full depth within the allocated timeframe.

7.3 Plans and Future Work

To address the identified risks and ensure completion of MakerSuite within the available time, the following plans are proposed:

1. Managing the Learning Curve

- Develop iteratively, starting with core features.
- Maintain internal documentation and modular code for easier integration and debugging.

2. External API Integration

- Prototype API connections early to identify constraints.
- Use abstraction layers and robust error handling to manage changes and rate limits.

3. Time Management

- Prioritize essential features (product syncing, inventory automation) first.
- Set short-term milestones and leverage pre-built libraries to accelerate development.

4. Future Work

- Conduct comprehensive testing and optimize system performance.
- Expand features such as workflow logging and dashboards in later iterations.
- Prepare full documentation and deployment guides for maintainability and scaling.

7.3.1 Project Plan with GANTT Chart

The project plan covers all phases of the project, including research, requirements gathering, analytics and design, implementation, testing/evaluation and submission of the final report. The GANTT chart created based on this project plan separates each phase into different swimlanes, with each swimlane being high level to allow for flexibility in case adaptability is needed for unexpectedly time-consuming tasks.

Therefore, there is one swimlane for the implementation of MakerSuite, but key implementation tasks include the development of back-end and database functionality, integration of external APIs, creation of product listings, the inventory management, customer communications and workflow logging functionality, front-end integration and UX enhancements.

Below is the Gantt Chart for the development of MakerSuite:

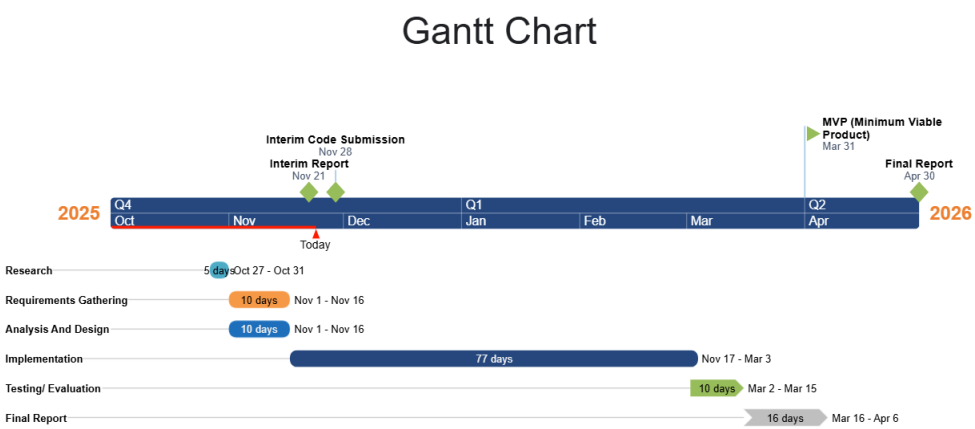

Figure 7.3.1.1. Gantt Chart Of MakerSuite Development

References

[1] C. Anderson, ‘Makers: The New Industrial Revolution’.

[2] ‘(PDF) (Not) Getting Paid to Do What You Love: Gender, Social Media, and Aspirational Work’. Accessed: Nov. 11, 2025. [Online]. Available: https://www.researchgate.net/publication/321127402_Not_getting_paid_to_do_what_you_love_Gender_social_media_and_aspirational_work

[3] A. Chukwuelue, ‘Seven Major Obstacles Small Businesses May Encounter’, Forbes. Accessed: Nov. 12, 2025. [Online]. Available: https://www.forbes.com/councils/forbesbusinesscouncil/2024/07/10/seven-major-obstacles-small-businesses-may-encounter/

[4] ‘10 common manufacturing challenges your small business might encounter’. Accessed: Nov. 12, 2025. [Online]. Available: https://www.brahmin-solutions.com/blog/10-common-manufacturing-challenges-your-small-business-might-encounter

[5] ‘Top 10 Social Selling Platforms and Tools (2025) - Shopify Ireland’, Shopify. Accessed: Nov. 12, 2025. [Online]. Available: https://www.shopify.com/ie/blog/social-selling-platform

[6] J. S. Hui and E. M. Gerber, ‘Developing Makerspaces as Sites of Entrepreneurship’, in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, Portland Oregon USA: ACM, Feb. 2017, pp. 2023–2038. doi: 10.1145/2998181.2998264.

[7] Apple Cheeks, *🌷 How I Organize My Small Business With Notion 🌷 Templates included!*, (Mar. 21, 2021). Accessed: Oct. 12, 2025. [Online Video]. Available: https://www.youtube.com/watch?v=S7PGRQHxBhs

[8] ‘Inventora - Inventora | Inventory System for Makers and Manufacturers | Shopify App Store’. Accessed: Nov. 10, 2025. [Online]. Available: https://apps.shopify.com/inventora

[9] ‘Reviews: Inventora - Inventora | Inventory System for Makers and Manufacturers | Shopify App Store’. Accessed: Nov. 17, 2025. [Online]. Available: https://apps.shopify.com/inventora

[10] N. Pascoe, ‘Features Craftybase Inventory and Manufacturing Software’. Accessed: Nov. 17, 2025. [Online]. Available: https://craftybase.com/features/

[11] ‘Reviews: Craftybase Inventory + Sync - Inventory, stock sync & COGS for small batch brands | Shopify App Store’. Accessed: Nov. 17, 2025. [Online]. Available: https://apps.shopify.com/craftybase

[12] ‘Customer service software’, Front. Accessed: Nov. 17, 2025. [Online]. Available: https://front.com/lp/customer-service-emea

[13] ‘Front Reviews - Pros & Cons’, Front Reviews - Pros & Cons. Accessed: Nov. 17, 2025. [Online]. Available: https://www.joinsecret.com/front/reviews

[14] A. Gaikwad, ‘Why React is a Great Choice for Your First Front-End Framework ✨’, Medium. Accessed: Nov. 18, 2025. [Online]. Available: https://medium.com/@gaikwadashutosh35/why-react-is-a-great-choice-for-your-first-front-end-framework-805c6c9943bd

[15] ‘Getting Started – React’. Accessed: Nov. 18, 2025. [Online]. Available: https://legacy.reactjs.org/docs/getting-started.html

[16] E.-E. I. Technology, ‘React Native for Front-End Developers: Transitioning from Web to Mobile Made Easy’, Medium. Accessed: Nov. 18, 2025. [Online]. Available: https://medium.com/@eitbiz/react-native-for-front-end-developers-transitioning-from-web-to-mobile-made-easy-69476a3ed944

[17] ‘Getting started with Angular - Learn web development | MDN’, MDN Web Docs. Accessed: Nov. 18, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Angular_getting_started

[18] ‘Django’, Django Project. Accessed: Nov. 19, 2025. [Online]. Available: https://www.djangoproject.com/

[19] ‘Flask Tutorial’, GeeksforGeeks. Accessed: Nov. 19, 2025. [Online]. Available: https://www.geeksforgeeks.org/python/flask-tutorial/

[20] K. Patel, ‘NodeJS vs Other Backend Technologies: A Detailed Comparison’. Accessed: Nov. 19, 2025. [Online]. Available: https://www.wedowebapps.com/node-js-vs-other-backend-technologies/

[21] ‘What is PostgreSQL? Introduction, Benefits, Use Cases & More’. Accessed: Nov. 19, 2025. [Online]. Available: https://www.percona.com/blog/what-is-postgresql-used-for/

[22] ‘Why Use MongoDB And When To Use It?’, MongoDB. Accessed: Nov. 19, 2025. [Online]. Available: https://www.mongodb.com/resources/products/fundamentals/why-use-mongodb

[23] ‘Dashboard’. Accessed: Nov. 23, 2025. [Online]. Available: https://vercel.com/solutions/react

[24] D. Baliles, ‘Python’, Heroku. Accessed: Nov. 23, 2025. [Online]. Available: https://www.heroku.com/python/

A) Appendix A: Online Survey Google Form (Help Shape A New Tool For Small Handmade Business Owners)

Help Shape A New Tool For Small Handmade Business Owners

Thanks for stopping by !! ٩(ˊᗜˋ*)و♡
I'm the owner of *With Love, Jeni.* where I make crochet bouquets, beaded jewellery and other handmade trinkets, as well a final-year computer science student! I'm building a **software tool designed specifically for small handmade business owners** to make your day-to-day operations easier.

If you **create and sell your own handmade products** — or hope to — you'd be a huge help!

The tool aims to be one, centralised place where you can manage:

• Listings across multiple marketplaces (Etsy/ Shopify/ Depop).

• Inventory (Products/ Raw Materials/ Tools).

• Customer messages across different channels (Email/ Socials/ Marketplaces).

This survey takes about **5–10 minutes**, and none of the questions are required — share as much or as little as you like.

Thank you so much for your support! ♡

1. **Email (Optional)**

   If you'd like to enter the draw to win a **crochet Miffy plush** (winner announced **Dec 12th, 2025**), you can pop your email here!

   This is just a little thank-you for taking part, your email won't be used for anything else, promise 🤗.

All About You And Your Handmade Business

Help me to understand the type of handmade business you run (or want to run!), what you make, the tools you use and how you currently sell your work.

2. What is your age range?

   Mark only one oval.

   - ◯ 18 or under
   - ◯ 19-24
   - ◯ 25-34
   - ◯ 35-44
   - ◯ 45-54
   - ◯ 55-64
   - ◯ 65+

3. What type(s) of handmade products do you create?

Check all that apply.

- [ ] Jewellery
- [ ] Fibre Arts (Crochet, Knitting, Felting etc.)
- [ ] Ceramics
- [ ] 3D Printing
- [ ] Art/ Illustration
- [ ] Soap/ Candles
- [ ] Other: _____

4. Do you currently sell your handmade products?

Mark only one oval.

- ◯ Yes, actively selling
- ◯ Not yet, but I'd like to
- ◯ No, and I don't plan to
- ◯ Other: _____

5. Where do you/ would you sell your products?

Check all that apply.

- [ ] Etsy
- [ ] Shopify
- [ ] Depop
- [ ] TikTok Shop
- [ ] Facebook Marketplace
- [ ] Personal Website
- [ ] Instagram DM's
- [ ] Word Of Mouth
- [ ] In-person markets/ pop-ups
- [ ] Other: _____

6. How long have you been selling handmade products?

Mark only one oval.

- ◯ Less than 6 months
- ◯ 6-12 months
- ◯ 1-3 years
- ◯ 3+ years

7. What accessibility features do you look for in a web app?

Check all that apply.

- ☐ Dyslexia-friendly features
- ☐ Colour contrast needs
- ☐ Screen reader support
- ☐ Motor accessibility needs
- ☐ None
- ☐ Other: _____

8. How many products do you typically sell per month?

Mark only one oval.

- ◯ 0-10
- ◯ 10-30
- ◯ 30-60
- ◯ 60+

Your Pain Points

Help me to understand what parts of running a handmade business feel confusing, stressful or time-consuming.

9. Which day-to-day operations of running your business do you find challenging?

These can be any part that feel confusing, stressful or time-consuming.

Check all that apply.

- [ ] Creating custom products
- [ ] Recreating products
- [ ] Tracking inventory of products
- [ ] Tracking inventory of raw materials and tools used to create products
- [ ] Managing multiple shop listings
- [ ] Calculating the cost of production and making a profit
- [ ] Managing custom orders
- [ ] Packaging and order fulfilment
- [ ] Messaging customers
- [ ] Being active on social media
- [ ] Marketing
- [ ] Tracking sales trends
- [ ] Staying organised
- [ ] Other: _____

10. What tools do you use to stay organised with your day-to-day operations?

These day-to-day operations can involve managing inventory, managing orders, track customer communications, logging workflows on product creation.

Check all that apply.

- [ ] Spreadsheet
- [ ] Calendar
- [ ] Notebook/ Physical Log
- [ ] Software Tools (e.g. Notion)
- [ ] Other: _____

11. Tell me about how you use the above tools stay organised with your day-to-day operations!

This doesn't have to be a super long answer, I just want to know how you use these tools and how they help you e.g. if you use Notion spreads to track your inventory, orders, ideas etc.

12. What frustrates you the most about your current tools to stay organised with your day-to-day operations?

This can be multiple things e.g. difficulty keeping track of physical notes, time-consuming to manually set up spreadsheets, lack of reminders etc.

13. What software tools would make it easier for you to manage your day-to-day operations?

Help Shape A New Tool To Address Your Pain Points

Help me to understand how I can develop features for listing management, inventory management, customer communications management and product workflow logging in a way that would help you run your small business!

14. Do you, or would you, sell products on multiple different platforms at a time?

This can be in-person (Word of mouth, pop-up markets etc,) or online (Etsy, Depop, Shopify etc.)

Mark only one oval.

- ◯ Yes, I do
- ◯ No, but I'd like to
- ◯ No, I'm not interested

15. If you answered "No, but I'd like to" to the above question, what's stopping you from selling products on multiple different platforms at a time?

Check all that apply.

- ☐ Too time-consuming to manage multiple dashboards
- ☐ Unsure how to start or set up new platforms
- ☐ Hard to keep inventory synced between platforms
- ☐ Worried about making mistakes (overselling, wrong stock counts, slow responses)
- ☐ Other: _____

16. What listing management features would be most useful to you in a software tool?

Check all that apply.

- ☐ Create new listings in one place
- ☐ Automatically sync sold/ out-of-stock products
- ☐ Import existing listings
- ☐ Bulk editing
- ☐ Analytics and reports on sales

17. What would make cross-listing (listing products on multiple different platforms at a time) easier for you?

e.g. Tools that automatically sync inventory across platforms, a way to import existing listings instead of recreating them, a single place to edit product details, automatic price adjustments for different platform fees

18. When creating your products, what types of inventory would you want to track?

Check all that apply.

- [ ] Raw materials
- [ ] Finished products
- [ ] Packaging supplies
- [ ] Tools used in production
- [ ] Time spent/ labour
- [ ] Other: _____

19. Which of these inventory management features would be important to you in a software tool?

Check all that apply.

- [ ] View of current stock of raw materials/ products
- [ ] Notifications when raw materials/ products are low
- [ ] Batch tracking
- [ ] Cost of goods calculation (COGS)
- [ ] Other: _____

20. Would it be useful to be able to link the raw materials used to a product you created?

e.g. For a crochet plush, you could link the type/colour/weight of yarn you used, approximately how much, crochet hook size used, and have this be reflected in your inventory system

Mark only one oval.

- ◯ Yes
- ◯ Maybe
- ◯ No

21. Would a single inbox to organise all customer messages from different platforms be useful?

Mark only one oval.

- ◯ Yes
- ◯ Maybe
- ◯ No

22. Which communication features would you want?

Check all that apply.

- ☐ Templates/ quick replies
- ☐ Saved customer profiles
- ☐ Analytics and reports (response time, message volume)
- ☐ Searchable conversation history

23. Would you use a feature that lets you log your creative process when making each product?

e.g. Log photos, raw materials and tools used, steps taken, time spent
This could help with reproducing the product or just documenting your creative process

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

Accessibility & Sustainability

Help me to understand how I can produce an accessible software tool for you.

24. What devices do you typically use for managing your business operations?

Check all that apply.

- ☐ Phone
- ☐ Tablet
- ☐ Laptop
- ☐ Desktop
- ☐ Other: _____

25. If you face any accessibility-related challenges when using apps/tools, please let me know about them!

_____

_____

_____

_____

_____

26. If you consider sustainability during your day-to-day operations in running your business, please let me know how!

Thank You So Much For Taking Part!

Your input is incredible valuable and will help shape my final year project.
If you provided your email, you're also entered into the draw to win a crochet Miffy plush (winner announced Dec 12th, 2025) 🫶.

27. If you'd like to be contacted to test early prototypes of my software solution, please provide your email 🤗

28. Please let me know if you've any more feedback or feature requests for a software tool designed for small handmade business owners!

B) Appendix C: Online Survey Data (Help Shape A New Tool For Small Handmade Business Owners)

Timestamp	**Email (Optional)**	What is your age range?
11/20/2025 17:14:16	franmagat39@gmail.com	
11/20/2025 18:01:29	valerieshramko@gmail.c	
11/20/2025 18:26:21		
11/20/2025 18:46:17	calvindelporte.1@gmail.c	
11/20/2025 18:48:53	davisjaudzems11@gmail	
11/20/2025 19:01:33		
11/20/2025 19:09:59	c22351646@mytudublin.	

11/20/2025 19:36:39	jackkava@gmail.com	
11/20/2025 20:50:26	kellyobeirne@hotmail.com	
11/20/2025 21:33:57	limantsatiashvili@gmail.c	
11/20/2025 22:02:57	nickbuksha@gmail.com	
11/21/2025 1:08:00	iamsammiyip@gmail.com	
11/21/2025 1:47:06	danellejp563@gmail.com	
11/21/2025 9:07:21	dolrxhive@gmail.com	
11/21/2025 9:08:28		
11/21/2025 10:22:42	theeiceefairy@gmail.com 19-24	
11/21/2025 10:24:05	lidiyazhang321@gmail.co 19-24	
11/21/2025 12:01:25	zophia0808@gmail.com19-24	
11/21/2025 13:27:40	jelena.vk@gmail.com 55-64	

Denim jeans	Not yet, but I'd like to

Fibre Arts (Crochet, Knitting, Felting etc.)	
Fibre Arts (Crochet, Knitting, Felting etc.), Art/ Illust	No, and I don't plan to
Art/ Illustration	Not yet, but I'd like to
Art/ Illustration	Did in the past
Art/ Illustration	Yes, actively selling
Art/ Illustration, miniatured	No, and I don't plan to
Fibre Arts (Crochet, Knitting, Felting etc.), Ceramics	Not yet, but I'd like to
Fibre Arts (Crochet, Knitting, Felting etc.), Art/ Illust	No, and I don't plan to
Jewellery	Yes, actively selling
Photographs	No, and I don't plan to
Jewellery, Art/ Illustration, Soap/ Candles	I use to sell
Art/ Illustration	Not yet, but I'd like to

Not yet, but I'd like to

Instagram DM's	6-12months
Etsy, Word Of Mouth, In-person markets/ pop-ups	
Word Of Mouth	Less than 6 months
Instagram DM's, Word Of Mouth, In-person markets	1-3years
In-person markets/ pop-ups	6-12months
Etsy	Less than 6 months
Etsy, Shopify, Instagram DM's, Word Of Mouth, In-pe	Less than 6 months
TikTok Shop, Personal Website, Instagram DM's, In-	
Etsy, Word Of Mouth	Less than 6 months
Shopify, Personal Website, Instagram DM's, Word Of	1-3years
Depop, Word Of Mouth, In-person markets/ pop-ups	Less than 6 months
Etsy, Instagram DM's, Word Of Mouth, In-person mar	Less than 6 months

0-10

	0-10
	0-10
	0-10
	0-10
	0-10
	0-10

	30-60
	0-10
	0-10
	0-10
	10-30

None	30-60
None	10-30
Colour contrast needs, Screen reader support	0-10
Colour contrast needs	0-10

Tracking inventory of raw materials and tools used Software Tools (e.g. Notion)	

Creating custom products, Recreating products, Ca Spreadsheet, Calendar
Creating custom products, Recreating products, Ca Calendar, Notebook/ Physical Log
Being active on social media, Marketing, Tracking s Notebook/ Physical Log
Managing multiple shop listings, Calculating the co Calendar, Software Tools (e.g. Notion)
Tracking inventory of products, Tracking inventory o Notebook/ Physical Log
Tracking inventory of products, Tracking inventory o Spreadsheet
Tracking sales trends, Staying organised Calendar, Notebook/ Physical Log
Creating custom products, Recreating products, Tra Spreadsheet, Calendar
Managing custom orders, Messaging customers, B Calendar

Creating custom products, Managing multiple shop Notebook/ Physical Log

Tracking inventory of raw materials and tools used Spreadsheet, Notebook/ Physical Log

Tracking inventory of products, Tracking inventory o Calendar, Notebook/ Physical Log, Procreate tbh

Tracking inventory of raw materials and tools used Calendar

Creating custom products, Tracking inventory of pro Spreadsheet, a booking website called acuity

Calculating the cost of production and making a pro Calendar, Notebook/ Physical Log

Managing multiple shop listings, Calculating the co Spreadsheet

Separation in folders	Time consuming

notebook to keep track of inventory	keeping track of different sets of products (e.g. 6 bl
Noting it so I remember when, who and what the or	Sometimes hard to find
On market days I track what I sell in a notebook and	There's a lot of moving parts, it can be difficult to re
list tasks and inventory	not easily accesibld
Use excel to track what I've made	Setting up the spreadsheet can be a hassle
N/A	N/A
tracking orders, inventory	all on different platforms
Write down each order, the price of orders.	Hard to calculate profit and postage.
Calendar - I use to timeblock when to make certain	
Notes app - for having my market application	

	No, but I'd like to
	No, but I'd like to
	No, but I'd like to
counters for each different product	No, but I'd like to
Microsoft project	Yes, I do
If it was all streamlined. Selling, keeping track of stock	No, but I'd like to
inventory/produxt manager	No, I'm not interested
Definitely an inventory tracking function, something	No, I'm not interested
N/A	Yes, I do
if it was all in one	No, but I'd like to
	No, I'm not interested
Maybe spreadsheet, blank document for brainstorm	No, but I'd like to
	No, but I'd like to

Pop up notifications to keep track of inventory

Too time-consuming to manage multiple dashboard Automatically sync sold/ out-of-stock products, Ana

Too time-consuming to manage multiple dashboard	Create new listings in one place, Automatically sync
Hard to keep inventory synced between platforms, W	Create new listings in one place, Automatically sync
	Automatically sync sold/ out-of-stock products, Ana
Too time-consuming to manage multiple dashboard	Create new listings in one place, Automatically sync
na	Create new listings in one place, Automatically sync
	Create new listings in one place, Automatically sync
Unsure how to start or set up new platforms, Want to	Import existing listings, Analytics and reports on sa
Too time-consuming to manage multiple dashboard	Create new listings in one place, Automatically sync
Too time-consuming to manage multiple dashboard	Create new listings in one place, Automatically sync
Too time-consuming to manage multiple dashboard	Create new listings in one place

A single place to edit product details	Raw materials

	Time spent/ labour
easy integration, dashboard	Finished products
Something that also helps update it on social media	Time spent/ labour
If the stock was communicated between the different	Finished products
	Raw materials
A single place to edit would make me more likely to	Finished products
N/A	Packaging supplies, Time spent/ labour
	Raw materials, Finished products, Packaging suppl
Tool that automatically syncs listing on different pla	Finished products, Packaging supplies, Time spent/
	Raw materials, Finished products, Packaging suppl

Notifications when raw materials/ products are low Maybe

	Maybe
View of current stock of raw materials/ products	Maybe
View of current stock of raw materials/ products	Yes
View of current stock of raw materials/ products	Maybe
View of current stock of raw materials/ products	Maybe
Cost of goods calculation (COGS)	Yes
View of current stock of raw materials/ products	Yes
Batch tracking, Cost of goods calculation (COGS)	Yes
View of current stock of raw materials/ products	Yes

Cost of goods calculation (COGS)	Maybe
View of current stock of raw materials/ products	Yes

View of current stock of raw materials/ products, N Yes

View of current stock of raw materials/ products, C Yes

Cost of goods calculation (COGS) Yes

Notifications when raw materials/ products are low Maybe

Maybe	Saved customer profiles
Maybe	
Yes	Templates/ quick replies, Analytics and reports (res
Yes	Searchable conversation history
Yes	Templates/ quick replies, Saved customer profiles,
Yes	Searchable conversation history
Yes	Templates/ quick replies

Yes	Templates/ quick replies, Saved customer profiles,
Yes	Saved customer profiles
Yes	Templates/ quick replies, Saved customer profiles,
Yes	Saved customer profiles, Analytics and reports (res
Yes	Templates/ quick replies, Saved customer profiles,
Maybe	Templates/ quick replies, Analytics and reports (res
Yes	Templates/ quick replies, Saved customer profiles,
Yes	Templates/ quick replies, Analytics and reports (res

Yes	Saved customer profiles, Searchable conversation h
Yes	Saved customer profiles, Searchable conversation h

Yes	Phone, Laptop, Desktop
	Phone, Laptop
Yes	Phone, Tablet, Desktop
Yes	Phone, Laptop, Desktop
Maybe	Phone
Yes	Phone, Laptop
Yes	Phone
Yes	Phone
Yes	Laptop
Yes	Phone, Desktop

Maybe	Phone, Tablet
Yes	Phone, Laptop
Yes	Phone, Tablet
Yes	Phone, Laptop
Maybe	Phone, Tablet, Laptop
Yes	Laptop
Maybe	Laptop

Tracking waste materials / using it for another proje

Online storefronts, and what option would be the be	Part time business, but the ordering in packaging in
	no
	Would like to keep packaging as environmentally fri
N/A	N/A
some tabs are hard to find, no clear path on the site	

	Using sustainable brands of yarn, reducing plastic u
No.	I would be interested in operating sustainably.

If you'd like to be contacted to test early prototypes Please let me know if you've any more feedback or	
franzliquit@gmail.com	
calvindelporte.1@gmail.com	
davisjaudzems11@gmail.com	
	Thank you I hope your project goes well 😗🦎💕
jackkava@gmail.com	
kellyobeirne@hotmail.com	☺💗
limantsatiashvili@gmail.com	
dolrxhive@gmail.com	I wish you luck for this project & maybe come back
lidiyazhang321@gmail.com	
zophia0808@gmail.com	Maybe a sheet that logs the amount of material nee
jelena.vk@gmail.com	

