G2U - a second life for every treasure

Course ID.: CPE-334

Submitted By-

Chawit Pimapansri	(ID: 65070503411)
Sorawit Tonpitak	(ID: 65070503438)
Jutamas Kaewchuenchai	(ID: 65070503444)
Nichaporn Manachaiprasert	(ID: 65070503446)
Thanakit Chokbunsuwan	(ID: 65070503448)
Arita Tragulmalee	(ID: 65070503470)
Yuil Tripathee	(ID: 65070503480)
Tom Medhi Pannier	(ID: 67540460025)

Submitted To-

Department of Computer Engineering
in partial fulfillment of the requirements
for the completion of
CPE-334 Software Engineering course.

Supervised by-

Dr. Natasha Dejdumrong
Associate Professor
Department of Computer Engineering
King Mongkut's University of Technology Thoburi (KMUTT)
Bangkok-10140, Thailand
1/2024

Abstract

G2U – A Second Life for Every Treasure is an developing e-commerce platform dedicated to promoting refurbishment and resale of second-hand consumer items. In response to growing environmental and economic challenges, G2U aligns with the United Nations' SDG Goal 12 - Sustainable Production and Consumption, advocating for responsible consumption and production. The platform serves as a marketplace for buying, selling, and organizing giveaways for refurbished products, addressing needs of eco-conscious consumers in Southeast Asian and European markets.

Using a Lean Startup approach, G2U steadily improves based on user feedback to refine its offerings and ensure scalability. With secure transactions, streamlined user management, and robust payment and shipping systems, the platform is positioned to lead in sustainable e-commerce. Market trends highlight a growing demand for sustainable, eco-friendly products, particularly in regions with strong repair policies and a focus on sustainability. As G2U grows, it will apply state of art testing methodologies like A/B testing and stress testing to optimize user experience and enhance platform performance.

Current version emphasizes the platform's iterative growth, capturing user feedback, and sustainability while incorporating the core elements of Lean Startup.

Keywords: Lean Startup, MVC, Software Engineering

Terms, Acronyms, and Abbreviations

Keyword	Description
ETA	Electronic Transactions Act (Thailand)
GDPR	General Data Protection Rule (EU)
MVC	Model View Controller web architecture
MVP	Minimum Viable Product
PDPA	Personal Data Protection Act (Thailand)
SEA	South East Asia
VAT	Value Added Tax (also called GST in some jurisdictions)
VPC	Virtual Private Cloud

Contents

Listings

List of Figures

Part I

Project Description

Chapter 1
Introduction

1.1 Background and Motivation

As of 2024, we stand in the face of turbulent geopolitics, frequent extreme weather events, and escalating living cost all around the world. And we chose to make a small step today towards a greater impact tomorrow. Unreasonable consumer demands is believed to be one of the leading factors that is keeping us behind on our goal of developing sustainable societies and planet. UN SDG goal 12 promotes sustainable consumption and production patterns, ensuring efficient use of natural resources.
This inspired us to found G2U. We are developing on online e-commerce platform where every treasure has a find its second life, to a new owner. Finding the gap in this nascent niche market, we intend to incorporate lean startup model to gain market knowledge & feedback to the maximum. Then, our agile engineering team will capitalize on the market input swiftly to capture the sizable market share.

1.2 Market study

In global outlook, the refurbished electronics market is projected to experience significant growth from $48.29 billion in 2022 to $94.10 billion by 2030, at a compound annual growth rate (CAGR) of 12%.
This highlights an expanding demand for more sustainable and affordable alternatives in electronics, driven by both economic and environmental causalities.

1.2.a SEA Market

Here is the outline for SEA Market:

Cultural Thriftiness: Thai consumers have a strong inclination toward affordability, making value-for-money products highly popular.

Ecological Consciousness: Over recent years, there has been a growing awareness among Thai consumers about sustainability.

Affordability and Sustainability: he twin goals of affordability and sustainability are guiding factors in the growing adoption of refurbished electronics in Thailand.

1.2.b EU Market

We see a different outlook in the EU market:

Affordability in Tough Economic Times: The EU has seen economic pressures, especially in the wake of the COVID-19 pandemic and rising inflation caused by war in Ukraine.

Eco-labeling and Tax Incentives: Some EU countries have introduced eco-labeling schemes for refurbished products, allowing consumers to easily identify environmentally friendly options.

Repair and Reuse Policies: The EU has also introduced policies that encourage the repair and reuse of electronic devices. The Right to Repair initiative, for example, aims to ensure that consumers can easily repair their products, thus extending their lifecycle and reducing the need for new purchases.

1.3 Scope of work

G2U is an innovative online platform designed to simplify the buying, selling, and donating of second-hand reusable items, with a strong focus on affordability and environmental sustainability. The platform differentiates itself from competitors like Lazada and Shopee by emphasizing eco-conscious practices and promoting a circular economy.
Concisely, we have segregated the scope of work into the following deliverables:

1. Interactive online platform.
2. Project documentation (diagrams, reports, presentation).
3. Information gathered from user feedback.

Chapter 2
Project Management

2.1 Incremental funding methodology

We use this technique to delegate high level business decisions. Each improvements to app that involves cash flow has to go through internal rate of return test for that particular incremental investment.
We intend to utilize lean startup principles to minimize upfront investments, iterate quickly, and validate assumptions with real customer feedback. The main goal of this model is to develop a sustainable cash flow model for scaling. [1]
Instagram's acquisition by Facebook for $1 billion with just 13 full-time employees highlights the potential for scalability and profitability with a lean team. Ad revenue benchmarks provide context for monetization strategies: [2]

CPC: $0.5–$3.5

CPM: $2–$20

SPE: $0.03–$0.08

2.2 Agile Method with Kanban Tool

Our team decided to use a blend of Scrum and Kanban tactics under Agile methodology. So far, it has been rewarding to adapt with necessary changes. Going forward, we have continued to stay with this plan. The particular details around the project management are sufficiently provided in other formats of project submission.

Part II

Requirements

Chapter 3
Requirements Elicitation

3.1 Elicitation Techniques

To validate our hypothesis with market needs and quantize the requirements, we applied the elicitation techniques as follows.

- Data Analysis
- Market Study
- Observation
- Prototyping and User Review

3.2 Stakeholders

Direct stakeholder acting in software include:

1. Customers (Buyers and Sellers)
2. Payment Gateway

Here are the indirect stakeholder that have influence over software development, but not the direct actors in the system:

1. Donation and Charity Campaigns
2. Authorized Resellers
3. Advertisement Companies
4. Shipping providers

3.3 Use Case Analysis

Initially, 3.1 had issues regarding the resemblance of DFD with the use case diagram. In the initial stages of engineering, we kept data flow analysis out of SDLC loop as it is redundant to sequence diagram (below). We can effectively model both system behavior, reaction (in terms of data changes and transfers) much effectively across our implementation team. Later to maintain the documentation, this work pattern was reverted and DFD diagram now is consistent to our use case diagram.

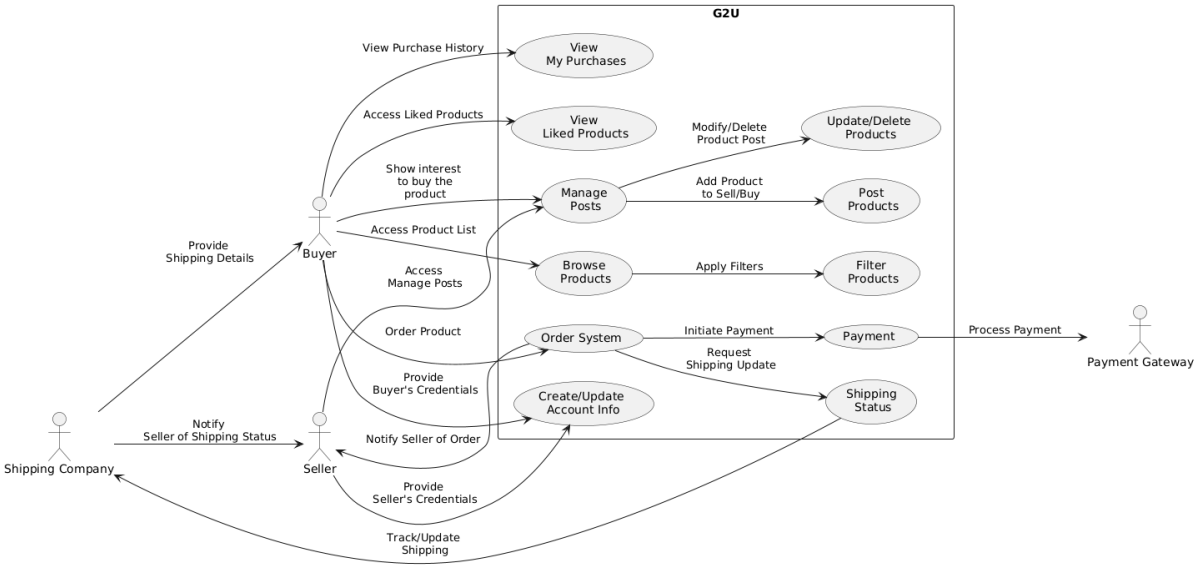

Figure 3.1: Latest version of use case diagram after feedback driven iteration

Upon final correction, we decided to incorporate data flow diagram as we modify our flow of application according to the user stories. Here, user stories are dictated by each individual use cases. Therefore, the data flow diagram which is created for the overall understanding of the system is maintained to be consistent with our changes in the use case diagram with the best of our abilities.
The use cases that are linked to preceding use case are well explained in the level 2 category of data flow diagram below. We attached it this way so that overall system is easily readable in the level 1 of DFD and finer details is based on conditions if the actor chose to access the particular small usecases (say Filter Products).

3.4 System Analysis - Data Flow

Figure 3.2 and 3.3 represents data flow diagram in level 0 and 1 respectively.

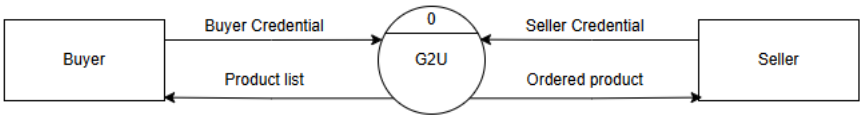

Figure 3.2: Data Flow Diagram - Level 0

And here comes the detailed DFD in level 2, this is separated into multiple parts starting from figure 3.4.

3.5 Functional Design

Here are some functional requirements prior to user's evaluation and feedback resolution. This based on market validation of our brainstormed hypothesis.

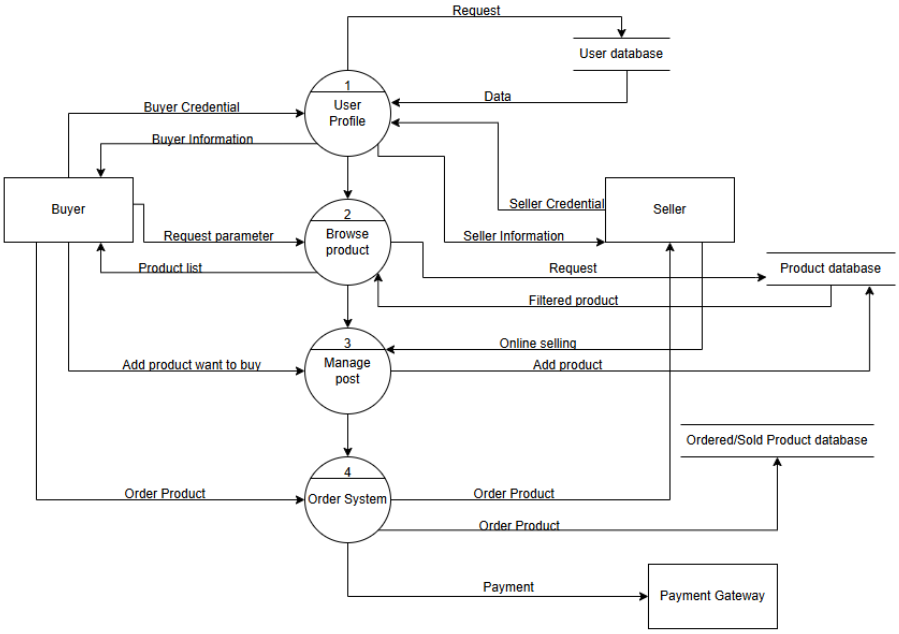

Figure 3.3: Data Flow Diagram - Level 1

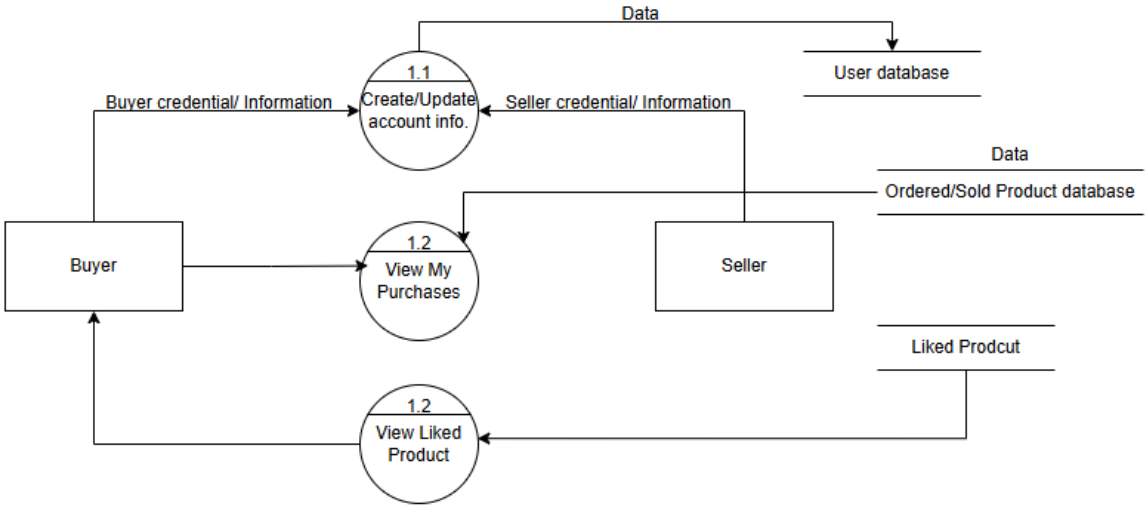

Figure 3.4: Data Flow Diagram - Level 2 part 1

1. Internal escrow system to tackle scam issues.
2. User management.
3. Payment and shipping.

During our application of Agile methodology, we can use these use case to track the progress based on user stories. For example, Browse Product can be a user story. Therefore, we can track the progress of certain specific elements from requirement to coding and document in this manner.
This is the tentative use case diagram (figure 3.1) which evolved as a result of validating our hypothesis throughout continuous user feedback and CI/CD practice.

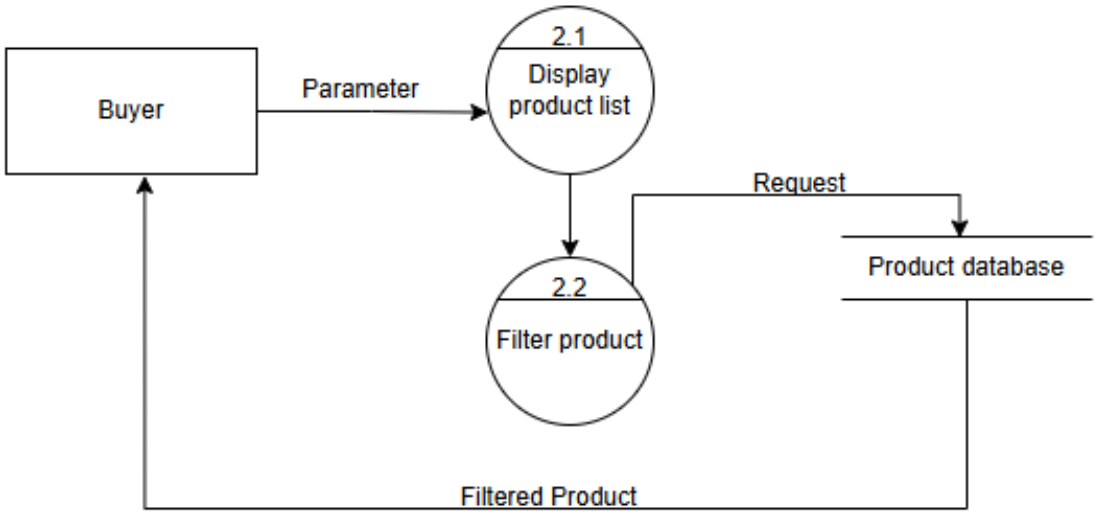

Figure 3.5: Data Flow Diagram - Level 2 part 2

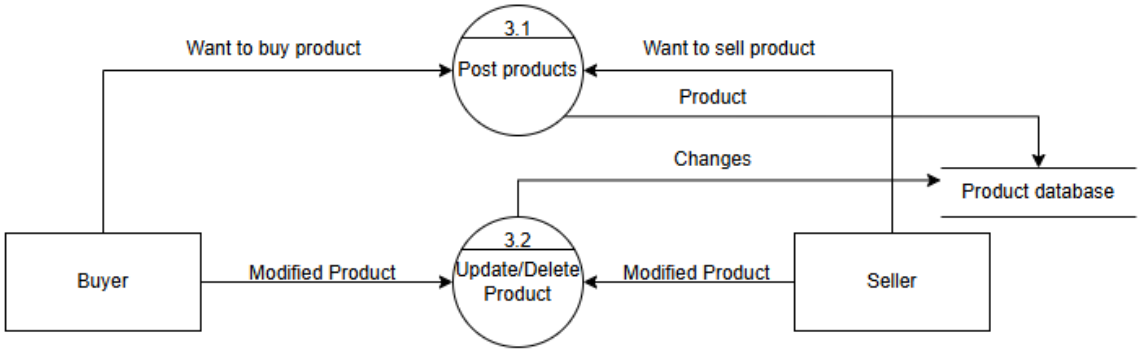

Figure 3.6: Data Flow Diagram - Level 2 part 3

3.6 Other Non-functional requirements

Scalability 1000 concurrent users, with minimal architectural changes.

Availability 99.5% uptime (43.8 hours of annual downtime)

Security Oath 2.0 with 256-bit encryption, implement VPC rules on database access.

Usability checker 3 testing sessions with 5-10 users in MVP phase.

Performance $200ms$ for 50th percentile on server, $100ms$ per query on database without performance degradation.

3.6.a Mandated constraints

The system must remain cost-effective during initial development and deployment, adhering to the constraints of a lean startup model. Expenses should prioritize scalability and maintainability, with a focus on minimizing fixed costs during the early stages. Therefore, incremented

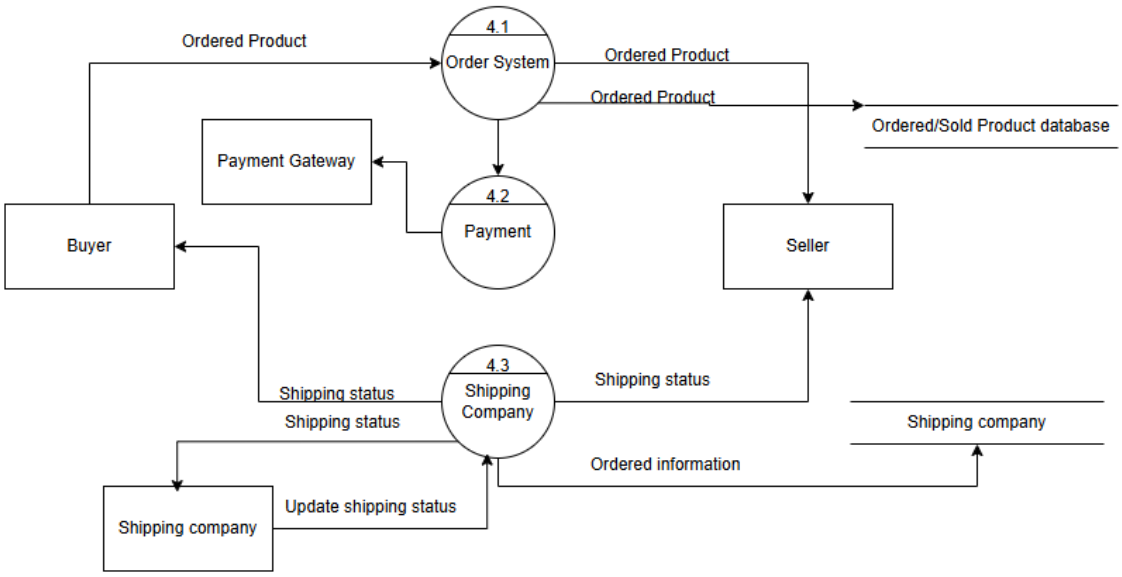

Figure 3.7: Data Flow Diagram - Level 2 part 4

funding methodology allows us do financial checks over every incremental investment per feature addition or improvement.

3.6.b Regulatory compliance

Briefly here are the regulatory issues we have encountered so far in the Thai business environment:

ETA ETA compiles data security requirements.

PDPA safeguards user data.

Tax compliance facilitate tax invoicing which is important for us and especially our B2B customers.

In the European side,

GDPR implements data protection rules especially in EU jurisdictions.

E-commerce directives safeguards customer protection.

Fiscal obligations to comply with local laws, including VAT collection and reporting.

Part III

Design and Development

Chapter 4
Systems Analysis and Design

4.1 Software Analysis

4.1.a Class Diagram

The rational behind this implementation is to have fastest and the most iterable base for our back-end. It blends the OOAD concept to adapt with relational design that is readily deployed on PostgreSQL database.
Figure 4.1 is the class diagram in our current system. The full image is available in the documentation repository, submitted along this project.

4.1.b Sequence Diagram

The rational behind our sequence diagram is simple. We want to visualize and communicate how every machine components interacts in terms of interaction and data transfer in sequence of individual operations.

4.1.b.1 Limitations

Unconventional implementation: Coding syntax such as including SQL as part of modeling app sequences. This is not part of best practice as specified in the UML specifications. However, it serves us more effectively as the means of communication for different technical sub-teams (front end, back-end and the deployment team). Our rationale behind this was to address the knowledge gap our current development team has. For the further updates, we intend to move up to the standard specification as knowledge gap is shortened and the back-end team is ready to receive requirements inputs using the top level description.

Less flexibility to stack changes For the current timeline, we chose the back-end stack to adapt customer needs as quickly as possible. However, if the requirements changes results in change of technical stack (for example, we switch from HTTP REST & SQL interface to GraphQL &gRPC); our specification for interaction sequences will be obsolete. The SQL-inspired specification cannot address changes to other modern day No-SQL schema (such as document store, column store). Therefore, if we change the database structure the specifics in the sequence diagram regarding SQL interface should be changed.

Figure 4.1: Class Diagram

4.2 Systems Design

4.2.a Demonstration model

Figure 4.8 is a minimal base on how our system is deployed and serving the user currently. We estimate to handle the peak of 10000 transactions per month comfortably in this setup. However, it might not meet the high up-time requirements as this server is currently hosted on our residential premises.

4.2.b Full scale production model

Upon reaching the designated transaction volume, we intend to migrate our application to a dedicated cloud service. Figure 4.9 suggests the tentative plan that allows us to deploy in

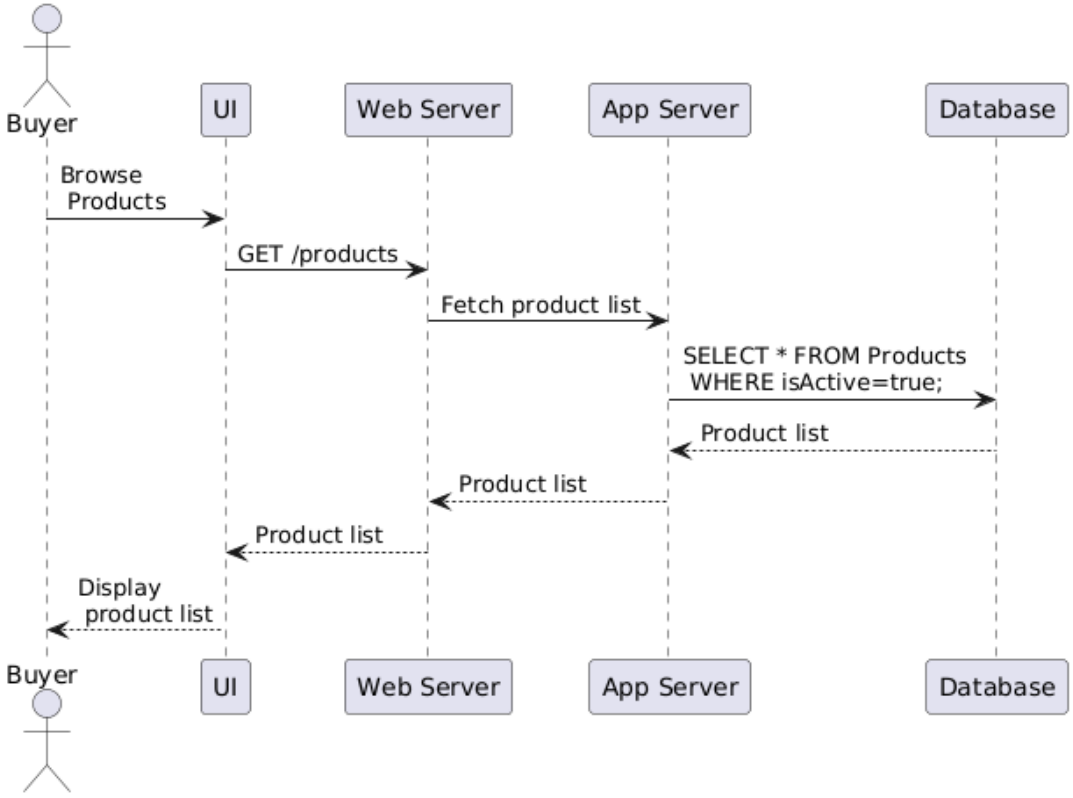

Figure 4.2: Sequence diagram for Use-case: Browse Products

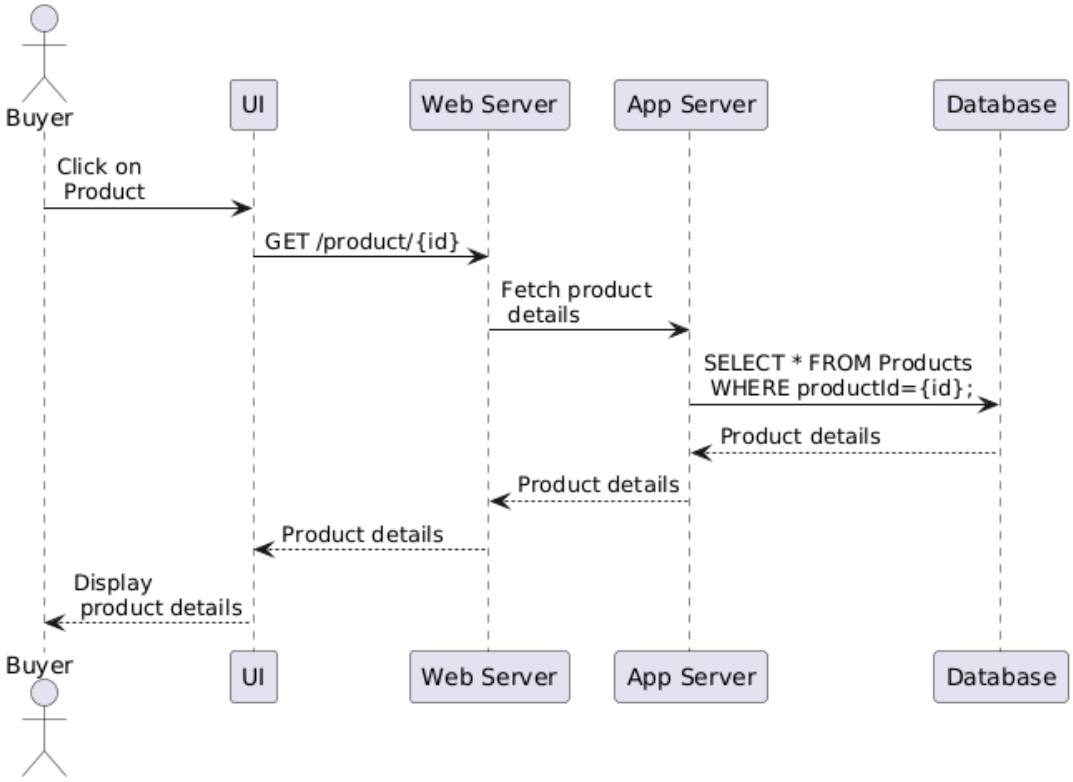

Figure 4.3: Sequence diagram for Use-case: View Products

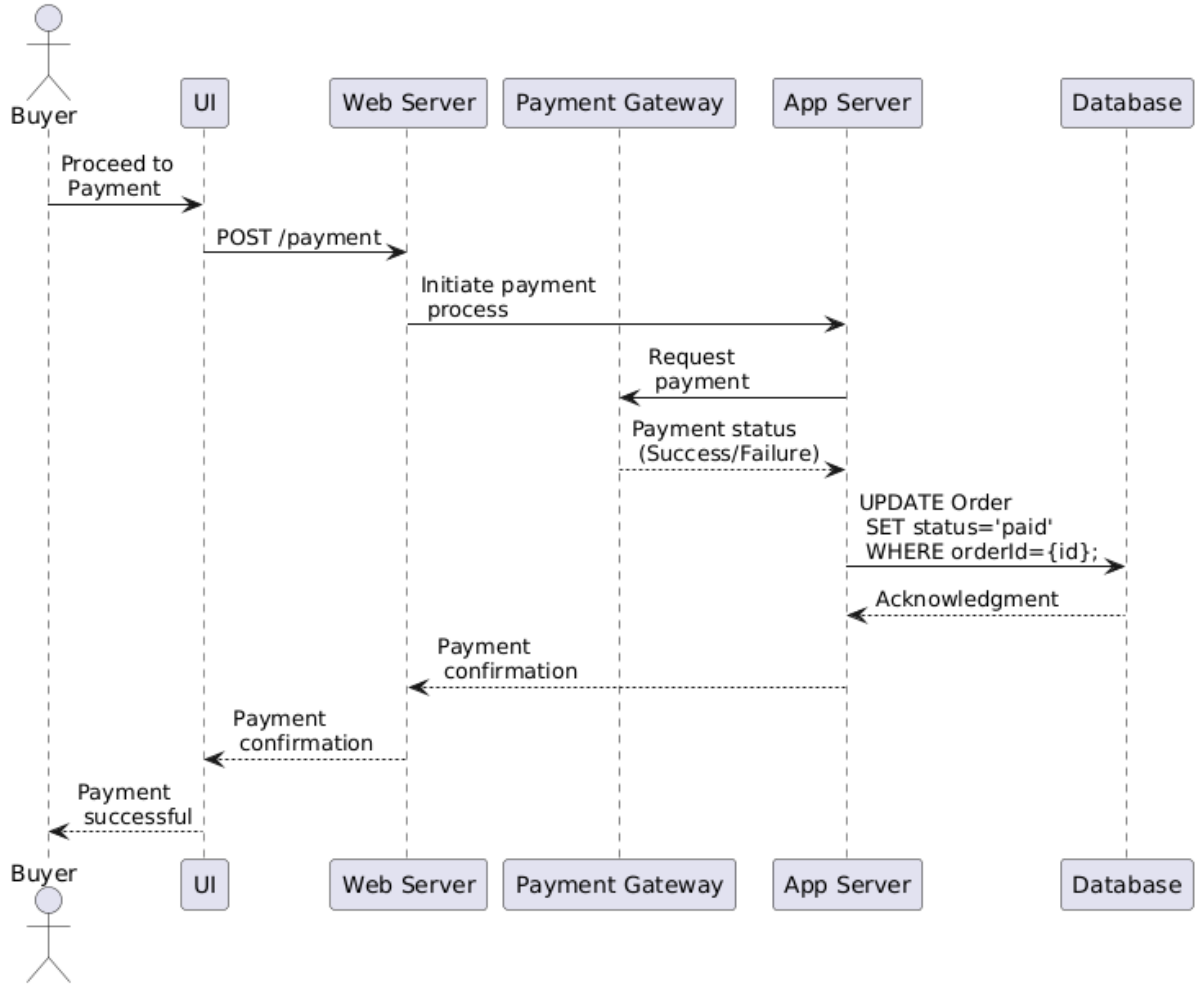

Figure 4.4: Sequence diagram for Use-case: Pay

cloud and scale.

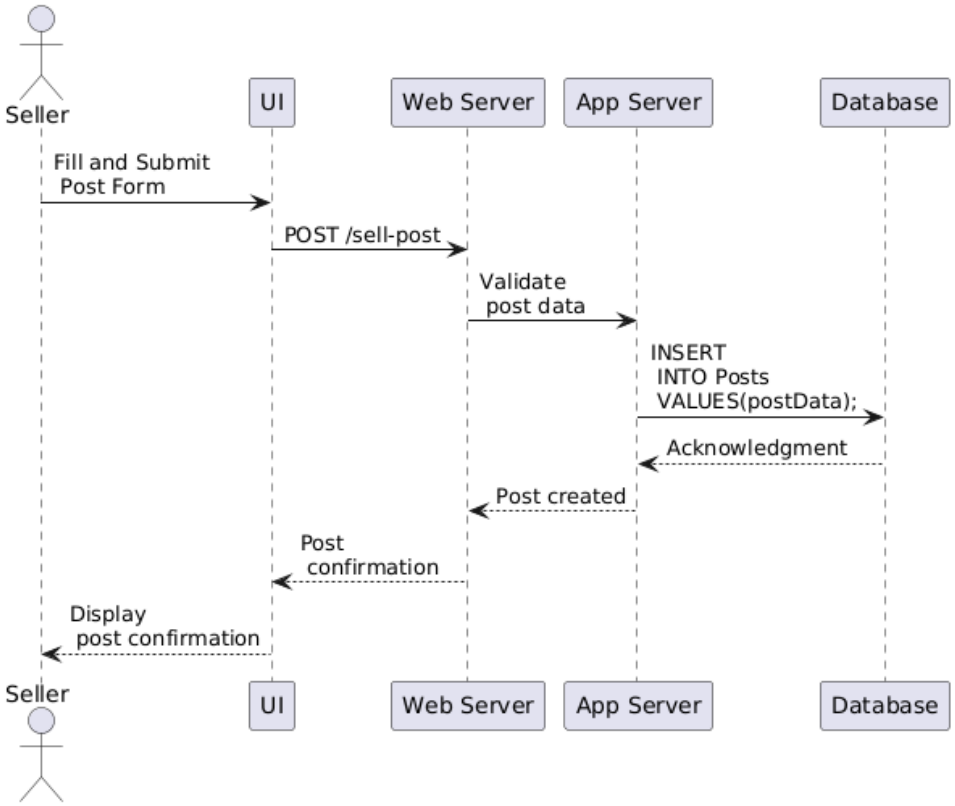

Figure 4.5: Sequence diagram for Use-case: Post new products

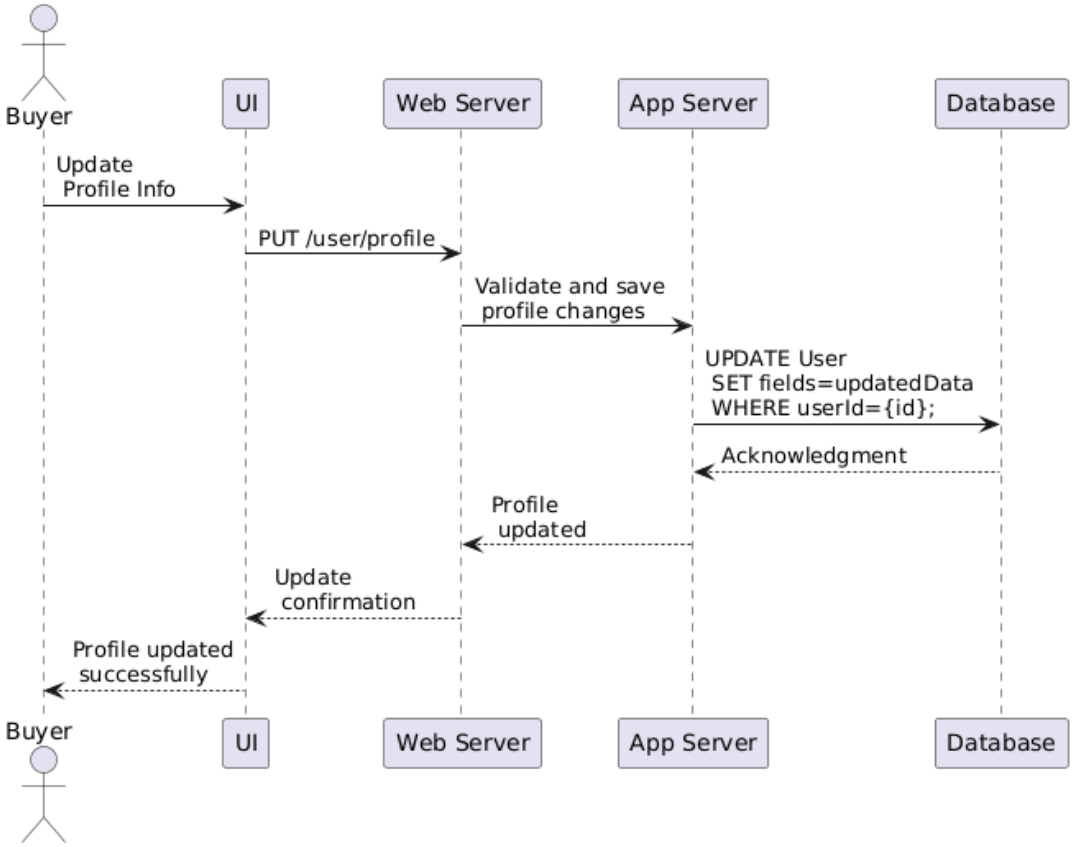

Figure 4.6: Sequence diagram for Use-case: Edit profile

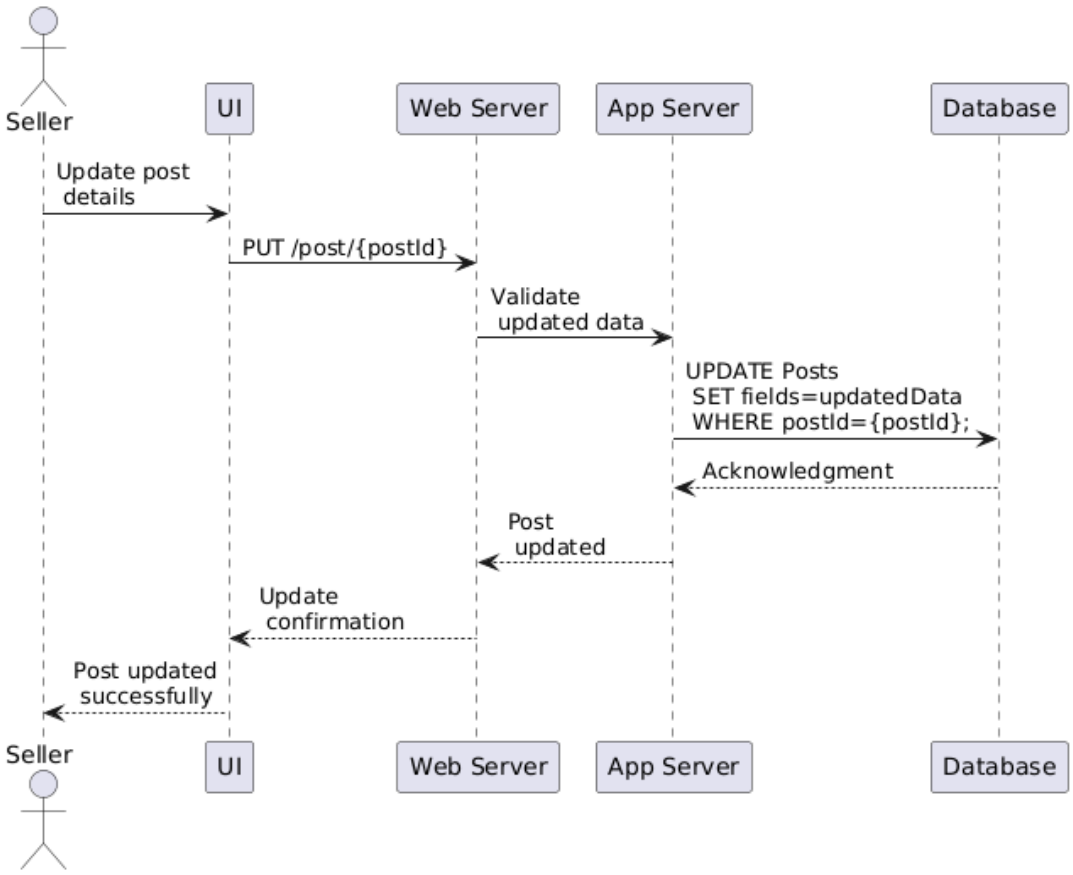

Figure 4.7: Sequence diagram for Use-case: Edit product post

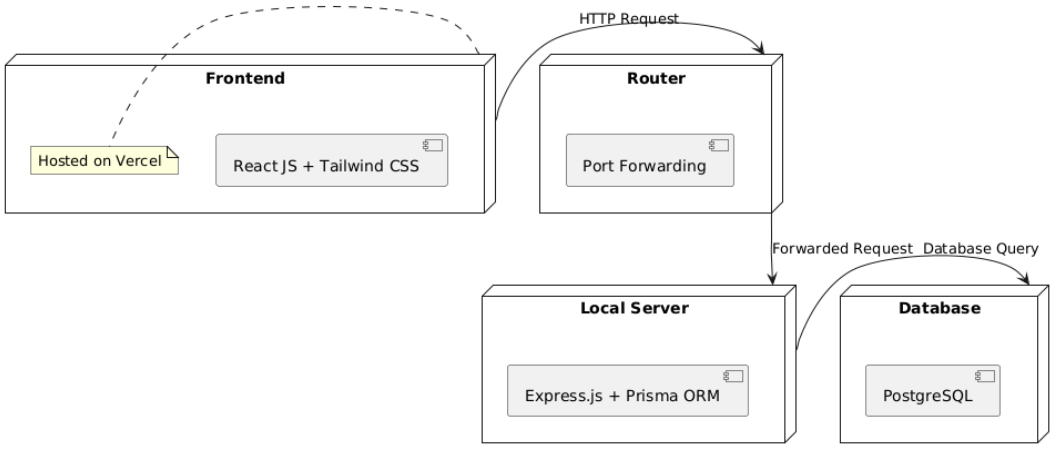

Figure 4.8: Development phase components layout

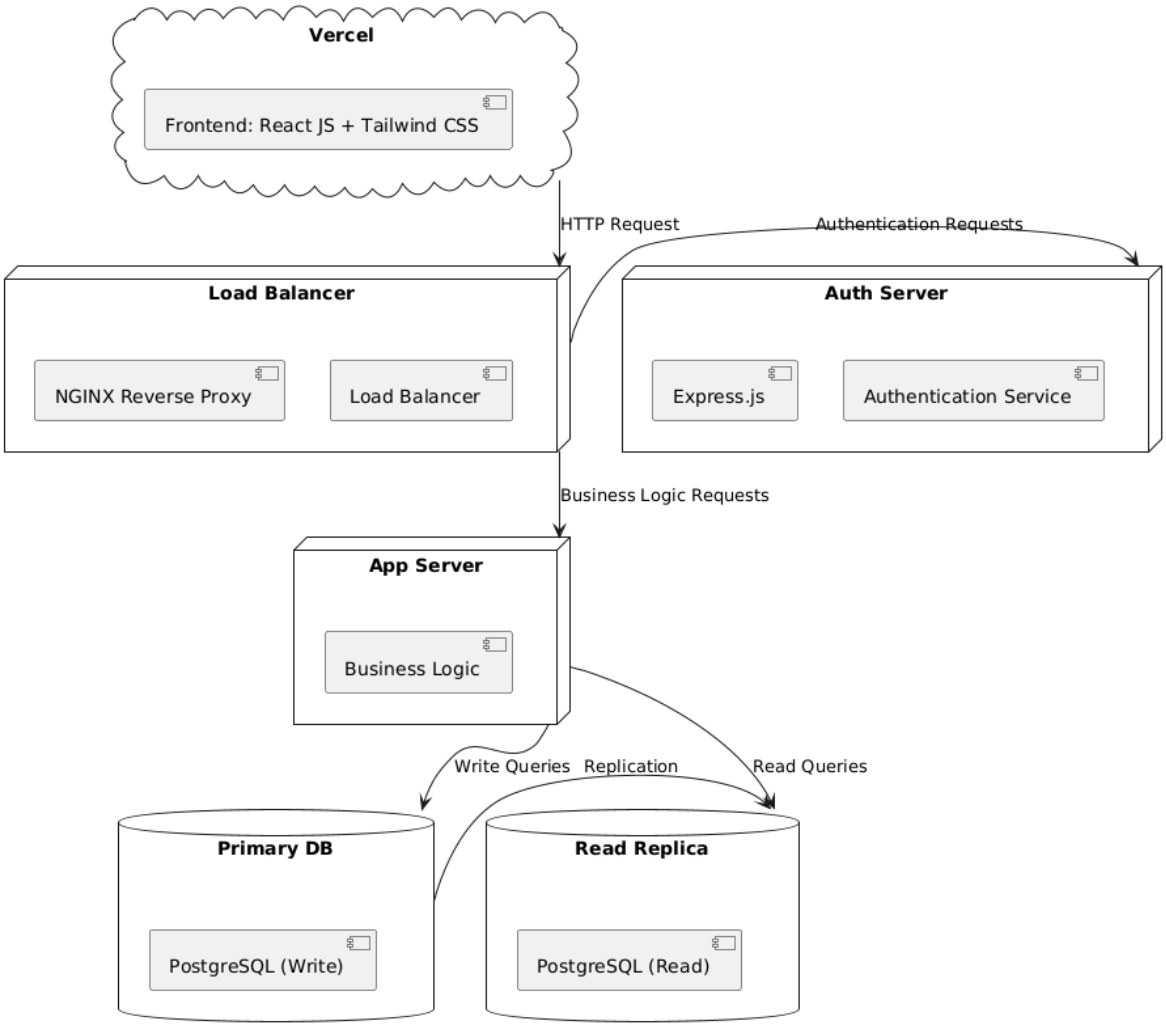

Figure 4.9: Deployment phase components layout

Chapter 5
Implementation

5.1 Prototyping

High fidelity design tools (Miro and Figma) were used through the entirety of the project. The public links and deliverables will be provided alongside this report.

5.2 Coding

Front end team implements code in the React components based architecture.

Listing 5.1: React Based Components example (View for MVC architecture)

```
export default function ProductCard({ product, layoutType = "default" }) {
    const handleImageError = (e) => {
        e.target.src = "/pic/default.jpg" // Fallback image
    }
    const navigate = useNavigate()

    // Function to mark a product as favorite
    const markAsFavorite = async (productId) => {
        try {
            const userId = parseInt(localStorage.getItem("userId"), 10)
            // Replace with actual userId logic
            console.log(userId)
            console.log(productId)
            const response = await axios
            .post("http://chawit.thddns.net:9790/api/users/like"
            , {
                productId,
                userId,
            })
            console.log("Product marked as favorite:", response.data)
            // Optionally update the UI or state to reflect the favorite status
        } catch (error) {
            console.error("Error marking product as favorite:", error)
        }
    }
//...
```

Back end team implements code in MVC (Model View Controller) model. Prisma ORM allows use to model our class diagram and update changes without need for having to configure

individual SQL commands by themselves.

Listing 5.2: Model for MVC architecture

```
model Auction {
    auctionId  Int         @id @default(autoincrement())
    userId     Int
    productId  Int
    startPrice Int
    minimumBid Int
    start      DateTime // Use DateTime for start time
    end        DateTime // Use DateTime for end time
    User       User        @relation(fields: [userId], references: [userId])
    AuctionLogs AuctionLog[] // Add this line for the relation
}
```

The view is produced by the React component. And here is the example for the controller:

Listing 5.3: Controller for MVC architecture

```
exports.createProduct = async (req, res) => {
    const {
        productName,
        categoryId,
        productDescription,
        productImage,
        userId,
        price,
        condition,
    } = req.body;

    // ...

    return res.status(201).json(product);  // ...
```

5.3 Systems Integration

Postman was the platform where both front end and back end team use to validate new API changes.
Here's an example for request and response on curl:

Listing 5.4: Example for API integration

```
# cURL Request
curl -X GET https://api.example.com/data \
-H "Authorization: Bearer YOUR_TOKEN" \
-H "Content-Type: application/json"

# cURL Response
{
    "status": "success",
```

```
    "data": {
        "id": 1,
        "name": "Example Data",
        "description": "This is an example response"
    }
}
```

Part IV

Test and Evaluation

Chapter 6
Evaluation of Outcomes

6.1 Testing Methodologies

So far, testing code correctness has been least of the issue. There we will continue performing higher level tests to confirm user's perception of our app. Here are the few methods we found appropriate:

A/B Testing We consider this as the most important test to retain user's feedback. If we give only one interface for user to test, users perceive this as an open ended questionnaire. It makes the participants in the test anxious and fall back to their previous experiences with the similar platform. But, if we provide the tester with two set of alternatives, it is quite easier to make comparative evaluation. Click up rate test compared in different applications helps us to exploit the things users are not willing to semantically express in the test participation (verbally or in written).

Integration Testing Such tests allow testing multiple parts of the software system as the group. We perform this test to see if two modules are interfacing well together, as this is an internal test. This test sets an important precedence before we bring our platform for review with the external users.

Stress Testing This is on the technical side and least relevant for now. Since our user base is small, this test has no use so far till now. But, in the coming future we might have to go through vertical and horizontal integration. We have to know how well the app can handle the load and where optimizations need to be made.

6.2 Results

We have observed A/B testing being tried and tested in by multiple tech companies. Facebook and YouTube gradually roll out their test features and set it as optional. Now, only thing left is to wait and watch user's behavior, especially which service do the users want to stay longer on. The classic example being Apple rolling their flagship MacBook Pro lineup with touch-bar and phasing it out after facing criticism.
We were able to replicate such methodologies my letting users make comparative evaluation between our previous MVP and iterative successor version of our platform. This test also blends with agile method which allows improvements in our platform with the least overhead. We have discovered this method to be the most effective in our test experience and it provides significantly better actionable information that lets the improvement of our application.

Chapter 7
Conclusion

7.1 Discussion

Aligning to the lean startup model, we allowed ourselves to effectively adopt agile engineering workflow. We set the correct success criteria to obtain customer feedback as much as possible, rather than following our prejudice of how an e-commerce platform should be.
With the flexible team setup and feedback driven development, we gladly announce on this date, that we have achieved to develop and deploy an MVP (minimum viable product), that is good enough for closed testing within trusted circles.

7.2 Future Work

In the days to come, we intend to keep improvising our platform based on user feedback from various domains. In summary, we will rewards users to educate us on what they want and then deliver exactly that.
In summary, our technical road map for FY 2024/25 is presented as below. The goal of the technical team is to reward the participants (as much as possible) to retain and maintain the customer base.

Q4 2024 Iterate the existing version of our application to adhere with security and business standards for European market.

Q1 2025 Improvise the existing version of our application to get market readiness to launch in Thai market.

Q2 2025 Integrate AI language model to improve localization and User Experience.

As for our business road-map, our goal is to sustain customer development process and keep getting credible feedback. As long as we have credible feedback, we are in the upper hand of business.

Q1 2025 participate in marketing festivals, book event launches, and university open house as supporting sponsors.

Q2 2025 experiment and launch giveaway platform and include influencers to participate (those with good follower base on Instagram and Tiktok).

Q3 2025 launch lottery quest to maintain customer retention.

7.3 Recommendation

7.3.a Opportunities

We are grateful to achieve these key distinctive learning outcomes, which is hard to achieve otherwise.

Secrets to quality software We learned (in a somehow hard way) that better code can make program better, but not a better software. We consider this to be the most important learning outcome. Until the software serves the intended need of user in the most effective way possible, software engineering processes should undergo necessary revisions by capturing user feedback with as much as actionable information as possible.

Software Engineering Economics A software engineering projects requires positive cash flow to survive and sustain. We explored and discovered several ways the project can be sustained in a long run with positive cash flow. This is crucial to us because we're going to explore the solution in a blue ocean environment. In such cases, the rules to deliver business values are not yet established, neither standardized. This requirement is quite relaxed in some environment though (like the companies who receive certain fees to complete outsourcing contracts). For us, optimum way to sustain our cash flow were scarce. However, lean startup came into rescue. It allows us to create a testing ground, while giving us space to test and validate new business models with optimum resource utilization.

Synergy When we practice certain discipline for long time in quest of being a subject matter expert in a specialized domain (say server side development), it is quite easy to get tempted and fall into trap of being a possible single point of failure. This is a very risky move because in a short run, we might observe a fast progress. In a long run, lack of presence of that stakeholder alone can jeopardize the engineering operation. Therefore, we found a healthy practice to have each individual team members get a slight generalized knowhow of their few peers. If need be, any potential crisis due to the absence of that developer can be mitigated before the project faces an existential risk.

7.3.b Challenges

If we had to do it over again, we learned that following challenges should be solved before the implementation phase begins:

Line of communication We have when programming team members are unfamiliar to each other due to some reasons (such as unmatched learning curve). Front end and back end teams have different backgrounds that shape their idea of integrating two ends together. There are some workable solutions to try out such as add a personnel specializing in full stack development. Role of such person would be to develop interfaces in the back end part and sharing data directly to the front-end team as they like. In this case both parties can maintain consistency when there is changes to schema or business logic.

However, this luxury might not be available all the times. Therefore, a common documentation such as shared OpenAPI specification [3] is found to be better alternatives. In order follow this approach, either team (front end or back end) has to put their desired specification in a shared document. Every time there is a breaking changes from either team, it has to go through this document first prior to code changes.

Retaining quality user's feedback This is perpetually existing challenge in teams of all tiers (from startups to established enterprises). User testing is easier said than done. Because of the competitive market, developer teams have dilemma in terms of trade offs for confidentiality before their app release versus information they should publicly release. Businesses generally can not follow academic style survey methodologies due to various regulatory and technical reasons. This led us to conclude that we always need to explore ways to understand and track user appeal to our features.

References

[1] Steve Blank. Everything about lean startup in 12 minutes, July 2024. URL https://www.youtube.com/watch?v=G-wwOK4X0lc.

[2] Investor Relations at Meta (Previously Facebook). Press release - facebook to acquire instagram, April 2012. URL https://investor.fb.com/investor-news/press-release-details/2012/Facebook-to-Acquire-Instagram/default.aspx.

[3] swagger.io. Openapi specification version 3.0, December 2024. URL https://swagger.io/specification/.