

Palestine Polytechnic University



College of Information Technology & Computer Engineering

Software Project

"قايض"

Project Team

Ahmed Abujheisha

Moath Rabai

Mohammed Hurob

Project Supervisor

Dr. Diya AbuZeina

This project has been submitted to the College of Information Technology and Computer Engineering as a partial fulfillment of the requirements for the Bachelor's degree in Computer Science.

Hebron, May 2023

إهداء

الحمد لله الذي هدانا لهذا وما كنا لنهتدي لولا ان هدانا الله

إلى من لا توفيهـم الكلمات والحروف حقهم في البر والإحسان، إلى من رضا الله في رضاهم وما توفيقنا وسر نجاحنا إلا بهم، إلى من كلهم الله بالهيبة والوقار، إلى من علمني العطاء بدون انتظار، إلى من تحمل اسماءهم بكل افتخار، نرجو من الله ان يمد بأعماركم لتروا ثمارا بعد طول انتظار، وستبقى كلماتكم نجوما نهتدي بها اليوم وفي الغد وإلى الابد والدينا

العزيزين

إلى من رافقونا طوال السنين وشاركونا الفرح والألم، إلى معنى الحب و الحنان والتفاني، إلى بسمة الحياة وسر الوجود إلى من كان دعائهم سر نجاحنا وحنانهم سر تقدمنا، إلى أمهاتنا، إلى من هم أقرب إلينا من روحنا، إلى من شاركونا حضن الألم وبهم نستمد عزتنا و إصرارنا إخوتنا الأعزاء، إلى من أنسنا في دراستنا وشاركنا همومنا تنكراً و تقديراً أصدقائنا، إلى من ساندنا وشجعنا طوال مسيرتنا الدراسية جميعاً، أهدي ثمرة هذا الجهد المتواضع وفاء، إلى حاملي لواء النور والسائرين في دربهم بإخلاص كل أساتذتي وجميع طالب العلم إليهم

شكر وتقدير

قال رسول الله -ﷺ: " من لا يشكر الناس لا يشكر الله "

نحمد الله سبحانه وتعالى على إحسانه وتوفيقه على ما أسداه علينا من نعم لا تعد ولا تحصى ونصلي ونسلم على نبي هذه الأمة الذي جاء رحمة للعالمين. وبعد، فإننا نتقدم ببالغ شكرنا وتقديرنا للدكتور المشرف ضياء ابو زينة على ما قدمه من نصح وإرشاد لإنجاز المشروع، ونشكر جميع الأساتذة المحترمين الذين رافقونا في رحلة العلم ولم يدخروا جهداً، كما نشكر جميع أصدقائنا وزملائنا الذين كانوا لنا خير مساعد ومعين

وأخيراً، نسأل الله دوام فضله، ونرجو أن يكون من نتاج هذا الجهد المتواضع بعض العلم الذي ينتفع به، وأن يكون هذا العمل خطوة متواضعة لطريق أكثر إثراء لمزيد من الأعمال

Abstract

Most of our transactions are now made through the Internet and well-known websites, which has improved our quality of life and saved people time and effort. The ability to access any desired item online and have it delivered from any location makes the world feel like a little village. The idea is to post your pre-owned items on our website so that someone else who needs them can trade them for something of comparable value rather than discarding them. The project aims to develop an online platform that facilitates the exchange of used items between users, promoting sustainable and eco-friendly practices. The system includes a user registration process and an interface that displays items available for exchange, suitable for a diverse range of individuals. Users can select items they wish to trade and initiate the exchange process through the website's interface. The website owner manages a database containing information about users and the items available for exchange. Successful implementation of the project can encourage responsible and sustainable consumption habits, minimizing waste and benefiting both society and the environment.

المخلص

تتم معظم معاملاتنا الآن عبر الإنترنت والمواقع الإلكترونية المعروفة، مما يحسّن نوعية حياتنا ويوفر الوقت والجهد للناس. إمكانية الوصول إلى أي عنصر مرغوب فيه عبر الإنترنت واستلامه من أي مكان يجعل العالم يبدو وكأنه قرية صغيرة. تقتضي الفكرة نشر المنتجات المستعملة على موقعنا الإلكتروني حتى يتمكن شخص آخر يحتاجها من استبدالها بشيء آخر يناسبه بدلاً من التخلص منها. يركز المشروع على نظام التبادل الإلكتروني. يهدف المشروع إلى تطوير منصة إلكترونية تُيسر تبادل المنتجات المستعملة بين المستخدمين، وتشجع على الممارسات المستدامة والصدقية للبيئة. يتضمن النظام عملية تسجيل المستخدمين وواجهة تعرض المنتجات المتاحة للتبادل، والتي تناسب مجموعة متنوعة من الأفراد. من خلال واجهة المستخدم التي تعرض المنتجات، يستطيع المستخدمون التفاعل مع بعضهم البعض في المشروع عن طريق التفاوض على تبادل ما يملكونه مقابل ما يملكه مستخدم آخر. يستطيع المستخدمون اختيار المنتجات التي يرغبون في تبادلها والبدء في عملية التبادل عبر واجهة الموقع الإلكتروني. يدير صاحب الموقع قاعدة بيانات تحتوي على معلومات عن المستخدمين والمنتجات المتاحة للتبادل. يمكن أن يحفز النجاح في تنفيذ هذا المشروع على تعزيز العادات الاستهلاكية المسؤولة والمستدامة، وتقليل النفايات وتحقيق فوائد للمجتمع والبيئة. يسعى هذا المشروع إلى تعزيز ثقافة التبادل والمشاركة بين الأفراد، مما يعزز التعاون والتكافل ويدعم المبادئ الأخلاقية والاجتماعية. يتماشى المشروع مع رؤية التنمية المستدامة ويساهم في الحد من الإهدار والتلوث.

Table of Contents

Contents

| | |
|--|-------------------------------------|
| إهداء..... | I |
| شكر وتقدير | II |
| Abstract..... | III |
| الملخص | IV |
| Table of Contents..... | V |
| List of table | VII |
| List of figures | IX |
| Chapter 1: Introduction..... | 1 |
| 1.1 Overview | 1 |
| 1.2 Project Idea | 1 |
| 1.3 Motivation and Importance | 2 |
| 1.4 Scope of the Project | 2 |
| 1.5 Goals to be attained by Quaid: | 2 |
| 1.6 Alternatives..... | 3 |
| 1.7 Timeline/ Project Scheduling | 4 |
| 1.8 Project Description | 6 |
| 1.9 Context Diagram..... | 6 |
| 1.10 Overview of the Document..... | 7 |
| Chapter 2: Requirements Specification | 8 |
| 2.1 Introduction..... | 8 |
| 2.2 Actors of the system | 8 |
| 2.3 System Requirements | 8 |
| 2.4 Use-Case Diagram | 11 |
| 2.5 Description of the system requirements for the project | 13 |
| 2.6 Sequence diagram | 22 |
| | Error! Bookmark not defined. |
| 2.7 Overview of the chapter..... | 25 |
| Chapter 3: System Design | 26 |
| 3.1 Introduction..... | 26 |
| 3.2 Alternative | 26 |
| 3.3 General structure of the System | 28 |
| 3.4 Models..... | 29 |

| | |
|---|----|
| 3.5 Controllers | 31 |
| 3.6 Database Tables | 34 |
| 3.7 Views..... | 38 |
| Chapter 4: Implementation | 42 |
| 4.1 Introduction..... | 42 |
| 4.2 Programming Languages and Technologies..... | 42 |
| 4.3 Implementation Approach | 43 |
| 4.4 Backend Development..... | 47 |
| 4.5 Frontend Development | 49 |
| 4.6 Overview of the Chapter | 51 |
| Chapter 5: Testing and Validation | 53 |
| 5.1 Introduction..... | 53 |
| 5.2 Testing Methodologies..... | 53 |
| 5.3 Validation Techniques | 62 |
| 5.4 Overview of the Chapter | 63 |
| Chapter 6: Future Plans | 65 |
| 6.1 Introduction..... | 65 |
| 6.2 System Enhancements and Feature Expansion | 65 |
| 6.3 Scalability and Performance Considerations..... | 66 |
| 6.4 Collaboration and Feedback Channels | 67 |
| 6.5 Overview of the Chapter | 67 |
| References:..... | 69 |

List of table

| Type: | Name: | Number of page: |
|-------------|----------------------|-----------------|
| Table 1.7.1 | Gantt Chart | 4 |
| Table 1.7.2 | Gantt Chart | 5 |
| Table 1.7.3 | TABLE KEY | 5 |
| Table 2.5.1 | Log in | 13 |
| Table 2.5.2 | Search for products | 14 |
| Table 2.5.3 | View product | 15 |
| Table 2.5.4 | Exchange products | 16 |
| Table 2.5.5 | Delete products | 17 |
| Table 2.5.6 | Upload product | 18 |
| Table 2.5.7 | Edit profile | 19 |
| Table 2.5.8 | Send report | 20 |
| Table 2.5.9 | Log out | 21 |
| Table 3.6.1 | Database Table | 34 |
| Table 3.6.2 | User Table | 35 |
| Table 3.6.3 | Product Table | 35 |
| Table 3.6.4 | Category Table | 35 |
| Table 3.6.5 | Report Table | 36 |
| Table 3.6.7 | products_image Table | 36 |
| Table 3.6.8 | Requests_offer Table | 36 |

| Type: | Name: | Number of page: |
|--------------|------------------------------|-----------------|
| Table 3.6.9 | user_cart Table | 36 |
| Table 3.6.10 | request Table | 37 |
| Table 3.6.11 | requests_communication Table | 37 |
| Table 3.6.12 | Reports_message Table | 37 |
| Table 5.2.1 | Test: Add New User And Login | 56 |
| Table 5.2.2 | Test: Add New Product | 57 |
| Table 5.2.3 | Test: Get All User Product | 58 |
| Table 5.2.4 | Test: Create New Request | 59 |
| Table 5.2.5 | Test: Get All Users | 60 |
| Table 5.2.6 | Test: Approve Product | 61 |

List of figures

| Type: | Name: | Number of page: |
|--------------|---|-----------------|
| Figure 1.9.1 | Context Diagram | 6 |
| Figure 2.4.1 | Use-Case Diagram | 12 |
| Figure 2.6.1 | Sequence diagram for deleting an account | 23 |
| Figure 2.6.2 | Sequence diagram for exchanging a product | 24 |
| Figure 3.2.1 | Layered Architecture | 27 |
| Figure 3.3.1 | MVC | 28 |
| Figure 3.4.1 | Class diagram | 30 |
| Figure 3.5.1 | Admin Controller | 32 |
| Figure 3.5.2 | User Controller | 32 |
| Figure 3.5.3 | Database mapping | 33 |
| Figure 3.7.1 | Login | 38 |
| Figure 3.7.2 | Signup | 38 |
| Figure 3.7.3 | Upload product | 39 |
| Figure 3.7.4 | Product Detail | 39 |
| Figure 3.7.5 | Product list | 40 |
| Figure 3.7.6 | Home Page | 41 |
| Figure 4.4.1 | An Overview of the System's Directory Hierarchy | 49 |
| Figure 4.5.1 | Implementing Responsive Design with Media Queries | 50 |
| Figure 4.5.2 | API Requests for Data Retrieval from Backend | 50 |

Chapter 1: Introduction

1.1 Overview

According to estimates, there are more than 1.7 billion websites. However, the figure changes daily. The internet is quite large, and 4.5 billion people use it to engage online. All praise goes to the digital revolution and our quick progress in moving our operations online. The change did not happen overnight; instead, it happened gradually.

The advent of visually oriented web browsers in the 1990s marked the beginning of the World Wide Web era for users. Since then, web technology has grown exponentially, and the trend toward web development is currently at its height.

Through our endeavor, we hope to assist people in gaining from one another. After setting up a personal account, a user logs in to the Qaied website to view the items on the public page and to exchange items with other users.

1.2 Project Idea

Qaied is a website that enables individuals to exchange new or used items without using money. When two clients agree to exchange their items through our system, they will exchange their contact information to discuss the details of the item exchange, including how and where to deliver the items to each other. Qaied does not provide delivery services, and users are responsible for arranging item delivery themselves.

1.3 Motivation and Importance

Nowadays, it is common for people to wish to replace electronic items or devices but need clarification about how to do so, whether to buy a new one or dispose of the old one. Electronics, for instance, contain chemicals that could leak harmful substances, making it difficult to dispose of them without harming humans and the environment. Alternatively, keeping them within the house without utilizing them takes up space. In light of this, we decided to build Qaied to allow users to exchange what they do not want for what they do want.

1.4 Scope of the Project

It should be understood that the project is not meant for only certain people in the community of Palestine; it is a website open to anyone within its borders, regardless of nationality. Registrants will be able to view the items that are currently available on the site in order to decide whether to exchange them with the owner of the item.

1.5 Goals to be attained by Qaied:

Qaied aims to facilitate the exchange of items over the internet, offering several advantages for users and the environment. It is accessible to all individuals residing in Palestine, regardless of their location. By using the Qaied system, users can exchange any items they own with others, thereby reducing waste, promoting resource efficiency, and contributing to a cleaner environment.

What we expect to be achieved by Qaied:

1. Provide an easy and convenient way for users to obtain secondhand items without the need to purchase new ones, thereby saving money and reducing the consumption of new resources.
2. Encourage sustainable practices by promoting the exchange and reuse of items, which can lead to a significant reduction in waste and environmental dumping.
3. Assist low-income households in obtaining items they need without having to pay money, which can help to reduce financial burdens.

1.6 Alternatives

We decided on a website for several reasons:

1 - Accessibility: A website can be accessed from any device with an internet connection, making it easy for users to access our platform from anywhere at any time.

2 - Ease of use: Websites are user-friendly and easy to navigate, which is essential for our target audience.

3 - Cost-effectiveness: The cost of developing a website is lower than other options such as developing a mobile application.

4 - Simplicity of updating and modifying: Websites are easier to update and modify than mobile applications. This is important since we plan to make continuous improvements and updates to the platform based on user feedback and changing needs.

Furthermore, a website offers several advantages over a mobile application:

- The web application is less complicated than a mobile application, making it easier for users to understand and use.
- Updating and modifying a web application is more straightforward than a mobile application, which requires users to download updates.
- Websites can be accessed on any device with an internet connection, while mobile applications require users to download and install the application.

1.7 Timeline/ Project Scheduling

We shall adhere to the software development life cycle to accomplish the goals. Furthermore, these are the duties that we must complete, as stated in table 1.1. Table 1.2 also includes the number of weeks needed to complete each assignment.

Gantt Chart

| Task number | Task name | The time required in weeks |
|-------------|---|----------------------------|
| 1 | System definition and planning | 5 |
| 2 | Determine the Project requirements | 4 |
| 3 | Description of the project Requirements | 4 |
| 4 | System design | 5 |
| 5 | System development and programming | 7 |
| 6 | Integration and system testing | 3 |
| 7 | Documenting the website | Along the working period |

Table 1.7.1 Gantt Chart

Gantt Chart

| | FIRST SEMESTER | | | SECOND SEMESTER | | |
|---|----------------|------|-------|-----------------|------|-------|
| WEEKS | 2-6 | 7-10 | 11-14 | 2-6 | 7-13 | 13-15 |
| System definition and planning | | | | | | |
| | | | | | | |
| Determine the Project Requirements | | | | | | |
| | | | | | | |
| Description of the project Requirements | | | | | | |
| | | | | | | |
| System design | | | | | | |
| | | | | | | |
| System development and programing | | | | | | |
| | | | | | | |
| Integration and system testing | | | | | | |
| | | | | | | |
| System documentation | | | | | | |
| | | | | | | |

Table 1.7.2 Gantt Chart

| | |
|-----------|----------------------------------|
| TABLE KEY | |
| | Estimate time to finish the task |
| | Real time to finish the task |
| | Holiday between semesters |

Table 1.7.3

1.8 Project Description

Our concept is to develop Qaied, a website that facilitates the exchange of machinery, electronics, and other items among users. Users can create an account to browse items available on the website. The website will be built using front-end software technologies like React JS and back-end software technologies like Node JS.

1.9 Context Diagram

A context diagram, also known as a Level O Data Flow Diagram, is a high-level view of a Data Flow Diagram. It illustrates the flow of information between the system and external entities. Business analysts commonly use this tool to gain an understanding of the system's scope and constraints required for a project. The context diagram only provides a high-level overview of the system, and the underlying structure's intricate details are hidden.

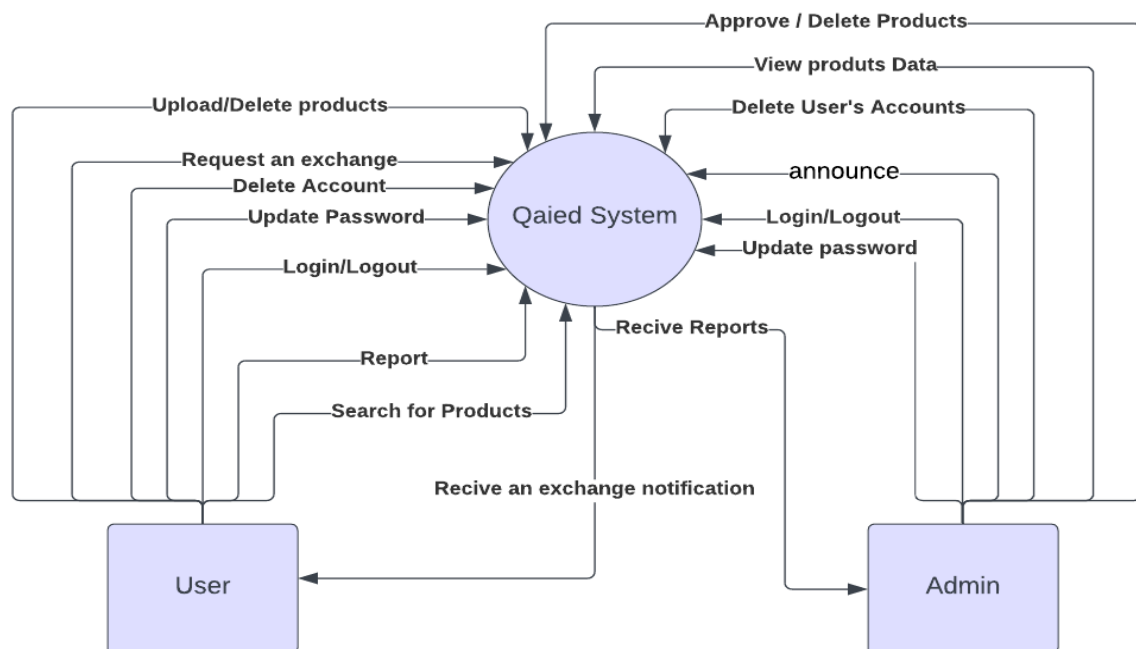


Figure 1.9.1 Context Diagram

1.10 Overview of the Document

Chapter 1 provides an overview of the project, including the idea, motivation, scope, goals, and alternatives. The chapter starts by discussing the large scale of the internet and the evolution of web technology. It then introduces the idea of Qaied, a website that enables individuals to exchange items without using money. The chapter explains the motivation behind the project, which is to promote sustainable consumption and reduce waste. It also discusses the scope of the project and its goals, such as assisting low-income households. The chapter then presents the decision to use a website instead of a mobile application and provides a timeline for the project's completion. Finally, the chapter concludes with a discussion of the project's context diagram and its usefulness for understanding the system's information flow.

Chapter 2: Requirements Specification

2.1 Introduction

This chapter focuses on the requirements specification for Qaied. It begins with an overview of the current system in which Qaied will operate, including its benefits and drawbacks. The chapter then outlines the project's goals, which have been established in the previous chapter.

The requirements analysis and description of Qaied will include the use model and context model of the system, as well as a detailed analysis of its requirements. This analysis will provide a brief overview of the requirements necessary for the successful implementation of the project.

2.2 Actors of the system

1. Admin: An administrator who manages the Qaied system.
2. User: A registered user of the Qaied system who can exchange products with other users.

2.3 System Requirements

One of the most crucial steps in the process of building a project is gathering and assessing the requirements for it.

The work to be done by the system's representatives will be broken down into manageable tasks in an integrated manner that satisfies its primary functions.

The system requirements are split into two categories:

1. Functional requirements
2. Nonfunctional requirements

Functional requirements for the project:

They are specifications that specify the features of each system function.

Functional requirements:

1. Functional requirements for the system Admin

1. A registered admin can log in to Qaied and gain admin privileges.
2. An admin can change their password.
3. An admin can announce important news and updates to all users.
4. An admin can delete users' accounts.
5. An admin can log out of Qaied.
6. An admin can view data related to the products being exchanged.
7. An admin can approve products to be listed for exchange.
8. An admin can delete listed products.
9. An admin can receive messages and reports from users.

2. Functional requirements for the system User

- 1.A registered user can log in to Qaied..
- 2.A user can change their password.
- 3.A user can report inappropriate content or behavior from other users or products.
- 4.A user can delete their own account.
- 5.A user can log out of Qaied.
- 6.A user can request to exchange their own product with another user's product.
- 7.A user can upload a product to be exchanged with another user's product.
- 8.A user can delete their own listed products.
- 9.A user can receive notifications about their exchanges.
- 10.A user can search for products to exchange with.

Non-Functional requirements for the project:

These requirements specify the characteristics of the system that are not directly related to its functions but are necessary for the system's overall effectiveness.

Non-Functional Requirements include:

1. **User-friendliness:** The system should have a user-friendly interface that is easy to navigate, with clear instructions and error messages to minimize mistakes and confusion.
2. **Reliability:** The system should be highly reliable, with minimal downtime or technical issues that could negatively impact users.
3. **Scalability:** The system should be designed to handle a large number of users and data, and should be easily scalable to accommodate future growth.
4. **Security:** The system should have robust security measures in place to protect user data and prevent unauthorized access or data breaches.

Domain requirements for the project:

1. The system should be designed to be simple and user-friendly, with an intuitive interface that allows users to easily navigate and utilize its features.
2. The system should be accessible at all times, with high availability and minimal downtime to ensure that users can access it whenever they need to.
3. The system should be designed to be easily updatable and adaptable to evolving needs, with a modular architecture that allows for seamless updates and upgrades as necessary.
4. The system should be designed with the understanding that its users may have varying levels of computer literacy, and should be simple enough to be used by anyone with the bare minimum of computer-related knowledge.
5. The system should be designed to serve all users in Palestine

2.4 Use-Case Diagram

The use-case diagram provides a high-level view of the system's functionality from the perspective of its users, focusing on what the system should do rather than how it should do it. The use-case diagram will include actors and use cases and will be designed to be simple and easy to understand.

Actors:

- Admin: Has access to all system functionalities and can perform administrative tasks.
 - User: Can access the system to exchange products with other users.
1. Login: Both Admin and User can log in to the system.
 2. Change password: Both Admin and User can change their passwords.
 3. Announce: Admin can post announcements for users.
 4. Delete account: Both Admin and User can delete their accounts.
 5. Logout: Both Admin and User can log out of the system.
 6. View exchanged data: Admin can view data of all exchanges that have taken place.
 7. Approve products: Admin can approve products uploaded by users to be listed.
 8. Delete products: Admin and User can delete their own products.
 9. Receive messages/reports: Admin can receive reports, and User can receive notifications.
 10. Report user/product: User can report other users or products.
 11. Request exchange: User can request to exchange products with another user.
 12. Upload product: User can upload a product to be exchanged with another product owned by another user.
 13. Delete own product: User can delete their own product.
 14. Search for products: User can search for products in the system.

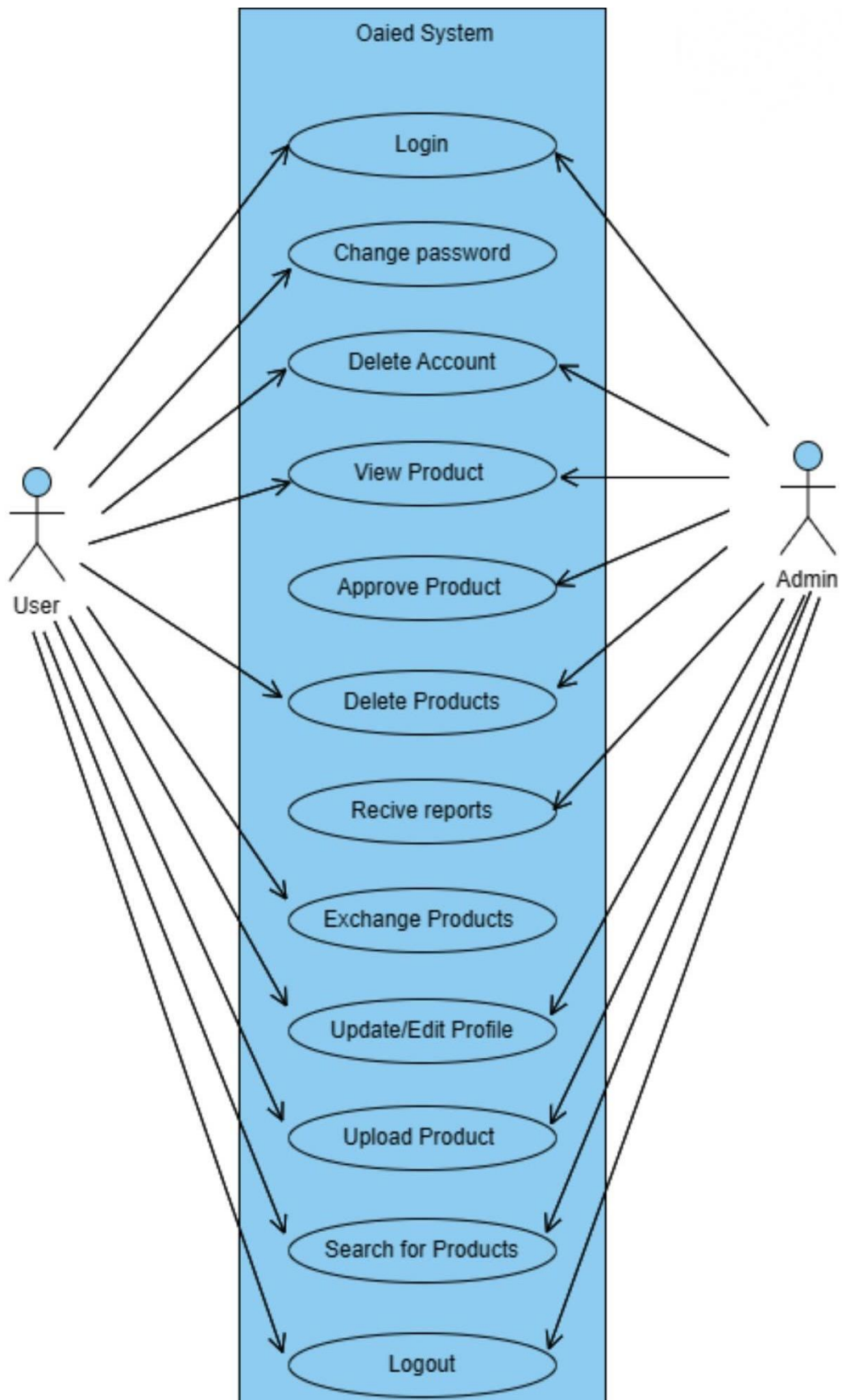


Figure 2.4.1 Use-Case Diagram

2.5 Description of the system requirements for the project

| | |
|------------------------|--|
| Use Case | Log in |
| Primary Actor | Admin, User |
| Goal In Context | To authenticate the user or administrator and grant access to the system |
| Precondition | The user or administrator has a registered account in the system |
| Trigger | The user or administrator navigates to the login page and enters their credentials |
| Scenario | <ol style="list-style-type: none"> 1. The user or administrator navigates to the login page of the system 2. The user or administrator enters their credentials (username and password) into the login form 3. The system verifies the user's or administrator's credentials by checking them against the database 4. The system grants access to the user or administrator if the credentials are correct |
| Exceptions | <ol style="list-style-type: none"> 1. If the user or administrator enters an invalid username or password, the system denies access and displays an error message. 2. If the user or administrator's account is inactive or blocked, the system denies access and displays an error message. |

Table 2.5.1 Log in

| | |
|------------------------|---|
| Use Case | Search for products |
| Primary Actor | Admin, User |
| Goal In Context | To search for products in the system and find relevant products that match the user's needs. |
| Precondition | The user is logged into the system and has access to the search feature. |
| Trigger | The user initiates a search by entering search criteria into the search form. |
| Scenario | <ol style="list-style-type: none"> 1. The user navigates to the search page of the system. 2. The user enters relevant search criteria, such as keywords or product categories, into the search form. 3. The system searches its database for products that match the search criteria. 4. The system displays a list of products that match the search criteria. 5. The user can view the details of a product by clicking on it in the list. 6. The user can further refine the search results by applying filters or sorting options. |
| Exceptions | <ol style="list-style-type: none"> 1. If the user enters invalid or incomplete search criteria, the system may not be able to find any matching products. 2. If the system encounters technical issues, such as database errors or network failures, the search feature may not function correctly. |

Table 2.5.2 Search for products

| | |
|------------------------|--|
| Use Case | View product |
| Primary Actor | Admin, User |
| Goal In Context | To view the details of a product in the system |
| Precondition | The user or administrator is logged in to the system or service and has access to view the product |
| Trigger | The user or administrator clicks on a product in a list or searches for a specific product |
| Scenario | <ol style="list-style-type: none"> 1. The user or administrator navigates to a page that displays a list of products or performs a search for a specific product. 2. The user or administrator clicks on a product in the list or selects a search result. 3. The system retrieves the details of the selected product from its database. 4. The system displays the details of the product, including its name, description, and any available options or variations. 5. The user or administrator can add the product to their shopping cart or wish list or initiate a purchase or exchange for the product. |
| Exceptions | <ol style="list-style-type: none"> 1. If the user or administrator does not have permission to view the product, the system will display an error message. 2. If the selected product is no longer available, the system will display a message indicating that the product is out of stock or no longer listed. 3. If there is an error retrieving the product details from the database, the system will display an error message. |

Table 2.5.3 View product

| | |
|------------------------|--|
| Use Case | Exchange products |
| Primary Actor | User |
| Goal In Context | Exchange a product with another user |
| Precondition | The user is logged in to the system or service and has a product that is eligible for exchange |
| Trigger | The user initiates the exchange process by navigating to the exchange page and selecting a product for exchange |
| Scenario | <ol style="list-style-type: none"> 1. The user goes to the exchange page in the system 2. The user selects a specific product for exchange 3. The system displays the details of the specified product and asks the user to send a notice to the product owner to replace it 4. The system checks if the other user has a product that they are willing to exchange and sends a notification to them 5. If the other user agrees to the exchange, the system updates the database to reflect the exchange and notifies both users |
| Exceptions | If the product is not available for replacement, the system will display an error message. If the other user declines the exchange, the system will notify the first user and cancel the exchange. |

Table 2.5.4 Exchange products

| | |
|------------------------|---|
| Use Case | Delete products |
| Primary Actor | Admin, User |
| Goal In Context | To delete one or more products from the system |
| Precondition | The user or administrator is logged in to the system or service and has permission to delete products |
| Trigger | The user or administrator initiates the delete process by selecting one or more products for deletion |
| Scenario | <ol style="list-style-type: none"> 1. The user or administrator selects one or more products for deletion. 2. The system prompts the user or administrator to confirm the deletion. 3. The system prompts the user or administrator to confirm the deletion 4. If the user or administrator confirms the deletion, the system removes the selected products from its database and updates any related records or accounts |
| Exceptions | <ol style="list-style-type: none"> 1. If the user or administrator does not have permission to delete products, the system will display an error message. 2. If the selected products have dependencies or relationships with other records, the system will prompt the user or administrator to resolve these dependencies before deleting the products. |

Table 2.5.5 Delete products

| | |
|------------------------|--|
| Use Case | Upload product |
| Primary Actor | Admin, User |
| Goal In Context | To user or allow admin to add a new product to the system |
| Precondition | The user or admin has the necessary permissions to add a product |
| Trigger | The user or admin initiates adding a new product by navigating to the appropriate page or form on the system |
| Scenario | <ol style="list-style-type: none"> 1. The user or admin accesses the product upload page or form 2. The user or admin enters the required information about the product, including the name, description, and any relevant images or documents 3. The user or admin submits the form 4. The system checks the information for accuracy and completeness 5. The product is added to the system and made available for the exchange if the information is correct |
| Exceptions | If the user or admin does not have the necessary permissions to add a product, the system will display an error message and will not allow the product to be added. |

Table 2.5.6 Upload product

| | |
|------------------------|--|
| Use Case | Edit profile |
| Primary Actor | Admin, User |
| Goal In Context | To allow the user or administrator to update their profile information in the system |
| Precondition | The user or administrator is logged in to the system or service and has permission to edit their profile information |
| Trigger | The user or administrator initiates the edit process by navigating to the account settings page and selecting the option to edit their profile |
| Scenario | <ol style="list-style-type: none"> 1. The user or administrator navigates to the account settings page of the system 2. The user or administrator selects the option to edit their profile 3. The system displays the current profile information for the user or administrator 4. The user or administrator edits the profile information as desired and submits the changes 5. The system verifies that the new profile information is valid and meets the system requirements 6. The system updates the profile information in its database and any related records or accounts |
| Exceptions | If the user or administrator does not have permission to edit their own profile, the system will display an error message. If the new profile information does not meet the system requirements, the system will display an error message and prompt the user or administrator to make the necessary corrections. |

Table 2.5.7 Edit profile

| | |
|------------------------|--|
| Use Case | Send report |
| Primary Actor | admin |
| Goal In Context | To generate and send a report from the system |
| Precondition | The admin is logged in to the system or service and has permission to generate and send reports |
| Trigger | The admin initiates the report generation and send process by navigating to the report page and selecting the options for the report |
| Scenario | <ol style="list-style-type: none"> 6. The administrator navigates to the report page of the system 7. The administrator selects the options for the report 8. The system generates the report based on the selected options 9. The system prepares the report for sending 10. The administrator reviews the generated report 11. The administrator enters the recipient's email address 12. The administrator clicks on the "Send" button 13. The system sends the report to the specified recipient's email address |
| Exceptions | If the administrator does not have permission to generate and send reports, the system will display an error message. If there is an issue with sending the report, such as an invalid email address or network connectivity problem, the system will display an error message and prompt the user or administrator to resolve the issue. |

Table 2.5.8 Send report

| | |
|------------------------|--|
| Use Case | Log out |
| Primary Actor | Admin, User |
| Goal In Context | To log out of the system |
| Precondition | The user or administrator is logged in to the system |
| Trigger | The user or administrator initiates the log out process by clicking the log out button or link |
| Scenario | <ol style="list-style-type: none"> 1. The user or administrator clicks the log out button or link 2. The system terminates the user's or administrator's session and logs them out of the system or service 3. The system redirects the user or administrator to the login page 4. The user or administrator is presented with a confirmation message indicating that they have been successfully logged out |
| Exceptions | If the system is experiencing technical issues, it may be unable to log the user or administrator out. In such cases, the system may display an error message informing the user or administrator about the issue and advise them to try again later |

Table 2.5.9 Log out

2.6 Sequence diagram

The sequence diagram, as an interaction diagram, depicts the interactions and order of actions among a set of components. It is a valuable tool for software engineers and business experts to understand the specifications of a new system or describe an existing procedure within the context of this project.

In the context of Qaied, the sequence diagram provides a visual representation of the flow of interactions between the actors, such as Admin and User, and the system components. It illustrates how the different functionalities and use cases are invoked and executed in a coordinated manner.

By utilizing the sequence diagram, stakeholders can gain insights into the precise order of actions and the overall behavior of the system. This helps in analyzing the system's functionality, identifying potential issues or improvements, and ensuring that the desired interactions are properly implemented.

1.1 Sequence diagram for deleting an account (User - Admin)

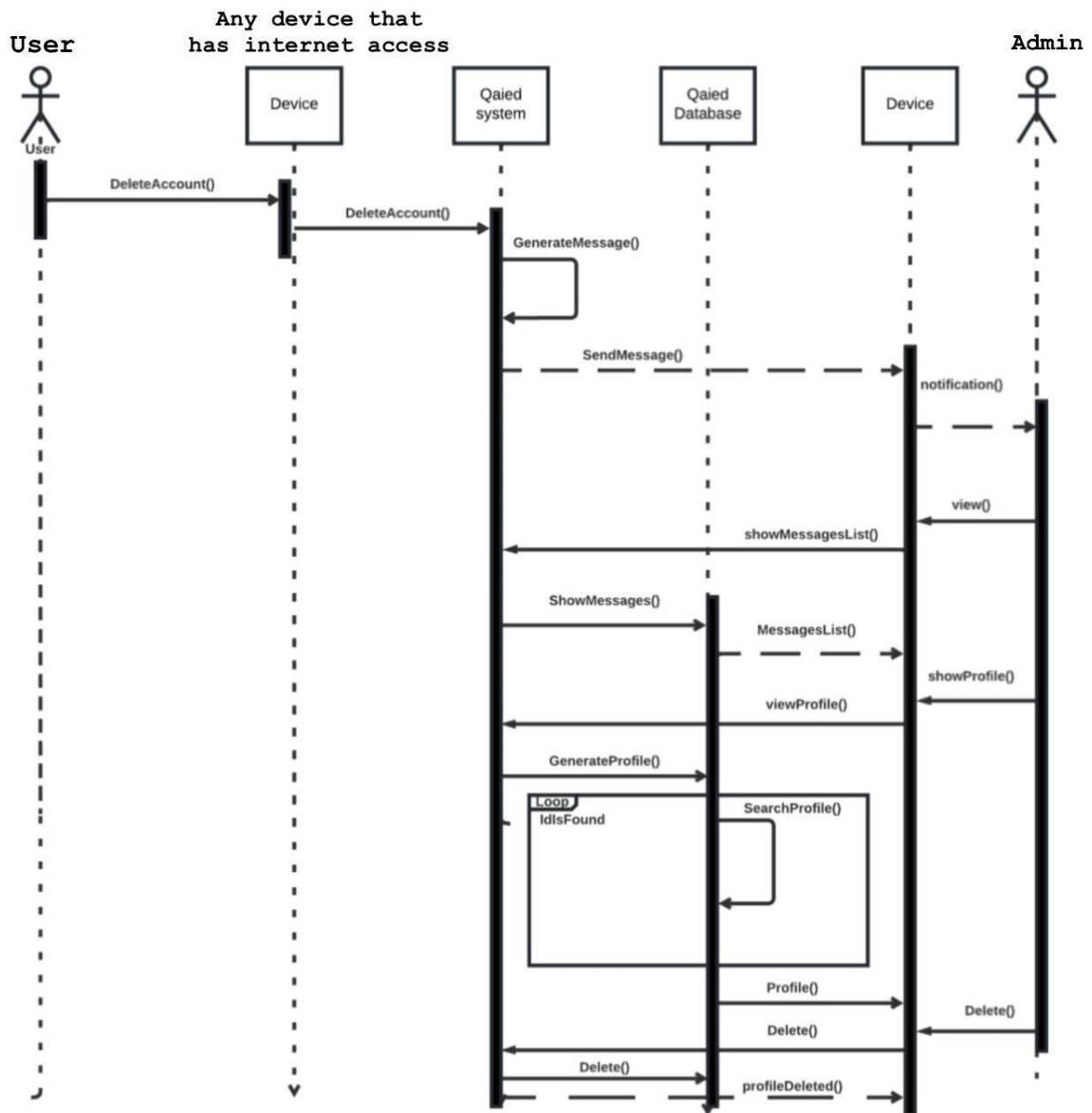


Figure 2.6.1 Sequence diagram for deleting an account

1.2 Sequence diagram for exchanging a product (User-User)

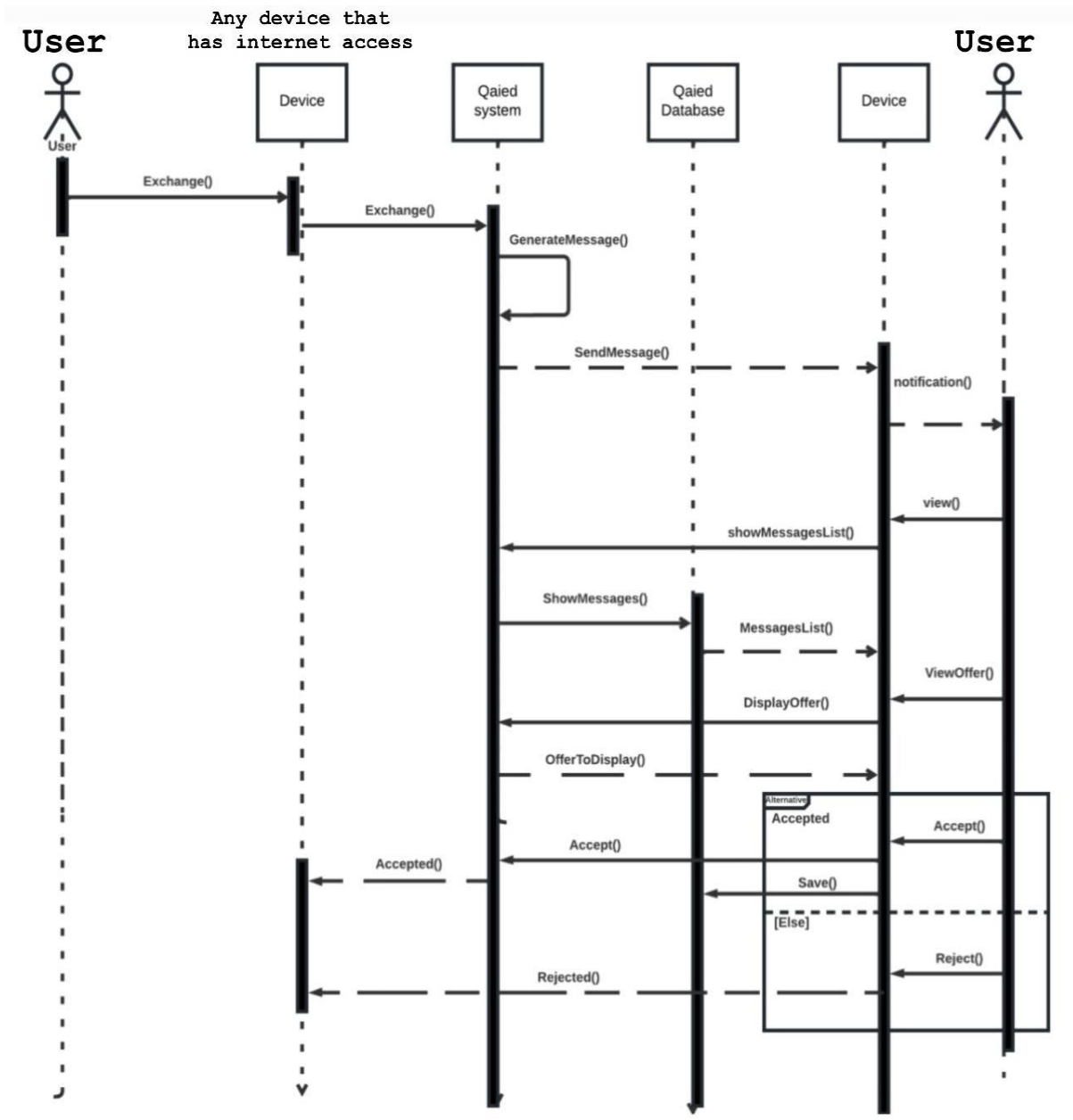


Figure 2.6.2 Sequence diagram for exchanging a product

2.7 Overview of the chapter

Requirements Specification provides a comprehensive analysis of the requirements for the Qaied system. It covers both functional and non-functional requirements, focusing on the roles of the Admin and User actors. The chapter includes a use case diagram illustrating the system's functionality and two sequence diagrams: one for deleting an account (User-Admin) and another for exchanging a product (User-User).

The sequence diagram for exchanging a product depicts the interaction between two users in the system. It showcases the sequence of actions involved in initiating and completing a product exchange process. This diagram helps in understanding the flow of events and the communication between users during the exchange.

Additionally, the chapter highlights the importance of user-friendliness, scalability, reliability, and security as non-functional requirements. These requirements ensure that the system is easy to use, always accessible, and capable of serving all users in Palestine. The chapter emphasizes the need for regular updates to meet evolving needs and acknowledges that users only require basic computer knowledge to utilize the system effectively.

Chapter 3: System Design

3.1 Introduction

Chapter 3 focuses on the system design for the Qaied project. It delves into the detailed parts and components of the system, providing clear explanations and drawings to aid the programmer in understanding and building the system effectively. The design phase also takes into account user preferences, ensuring that the system's design aligns with user expectations through user interfaces and database considerations. This chapter serves as a bridge between the requirements analysis and the implementation phase, providing a blueprint for development that meets requirements and enhances user experience.

3.2 Alternative

In our exploration of system architectures, we evaluated two options: Layered Architecture and Model-View-Controller (MVC).

- **Layered Architecture :**

Layered Architecture divides the system into separate layers, each representing a component or subset of the system. However, our research revealed that this architecture is not suitable for websites and is better suited for desktop and mobile

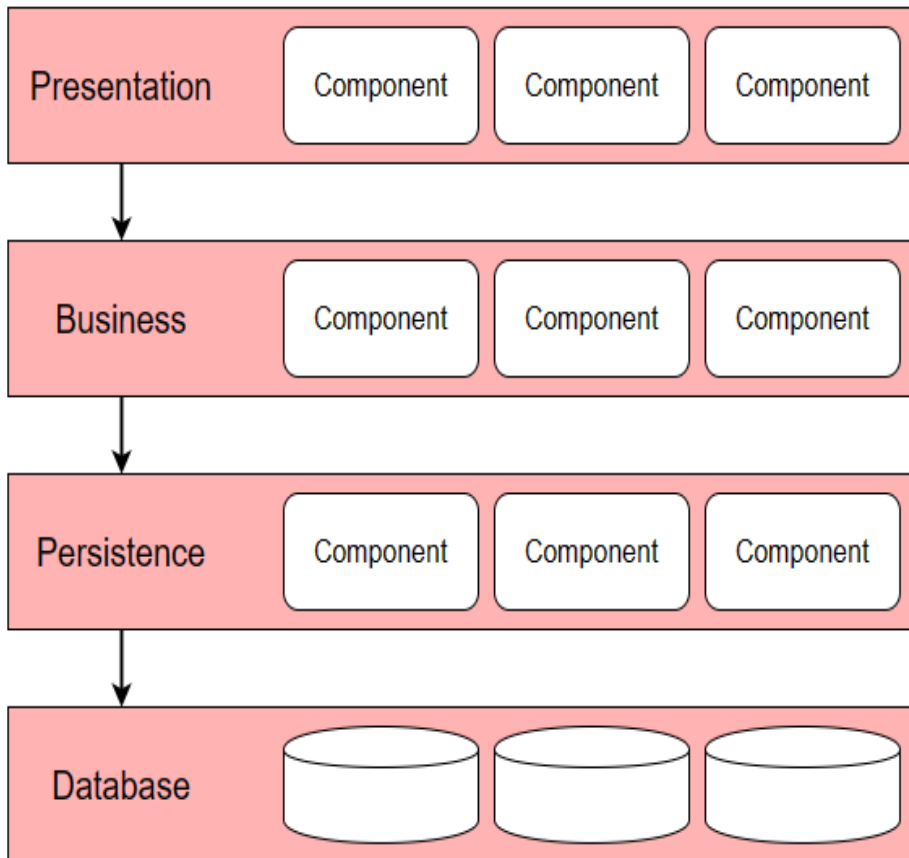


Figure 3.2.1 Layered Architecture diagram.

Through research into layered architecture, we discovered that websites should not use it and are better suited for desktop and mobile apps (web).

- **Model–view–controller**

After considering both Layered Architecture and MVC, we concluded that MVC is a more suitable choice for the Qaied system. MVC offers several advantages, including:

1. Independence of Components: Each component in MVC is independent, enabling individual testing and simplifying code maintenance and development.
2. Clear Function Specification: MVC allows for the specification of specific functions for each component, enhancing clarity and organization.

3. Seamless Data Updates: With MVC, data can be updated without impacting the user view of the system, ensuring a smooth and uninterrupted user experience.

3.3 General structure of the System

In the MVC structure, the system is divided into three main components:

- Model: The Model represents the data and business logic of the system. It encapsulates the data and provides methods for accessing and manipulating it.
- View: The View is responsible for the presentation and user interface of the system. It displays the data from the Model and interacts with the user to capture input or display information.
- Controller: The Controller acts as an intermediary between the Model and the View. It handles user input, updates the Model accordingly, and updates the View to reflect any changes in the data.

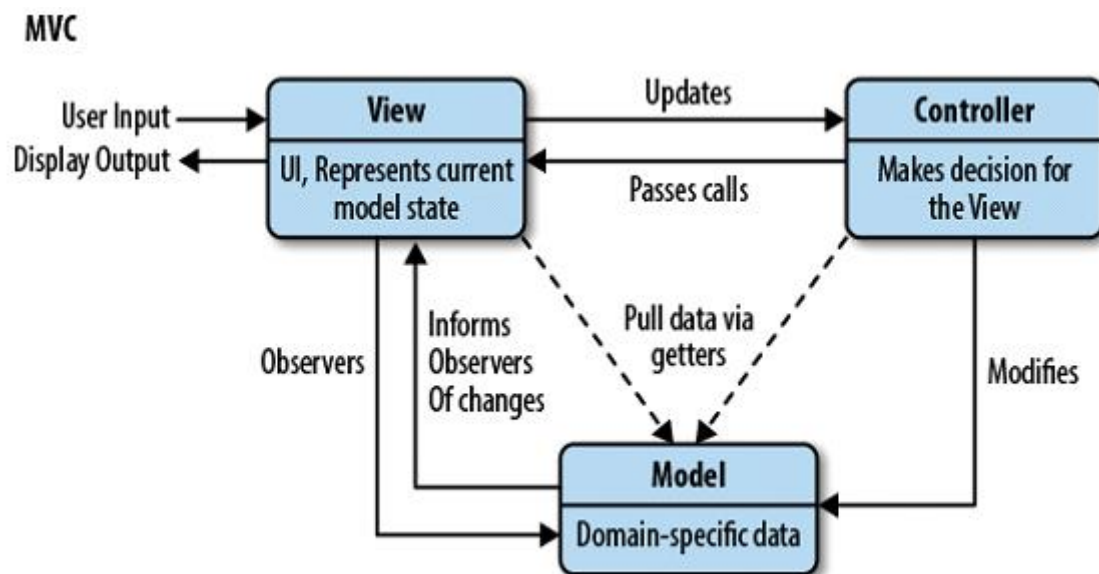


Figure 3.3.1 MVC diagram

Following is a description of the system's components by its structure.

The Model component of the system is responsible for managing data and interacting with the database. It encompasses various categories and handles operations such as retrieving and saving data to and from the database. The Model component plays a crucial role in the overall system structure as it serves as the core of the program.

3.4 Models

The model layer of our system consists of several primary elements, as depicted in the following class diagram.

Class diagram model:

The class diagram represents the fundamental models that are part of the model layer in our project, adhering to the MVC architectural pattern.

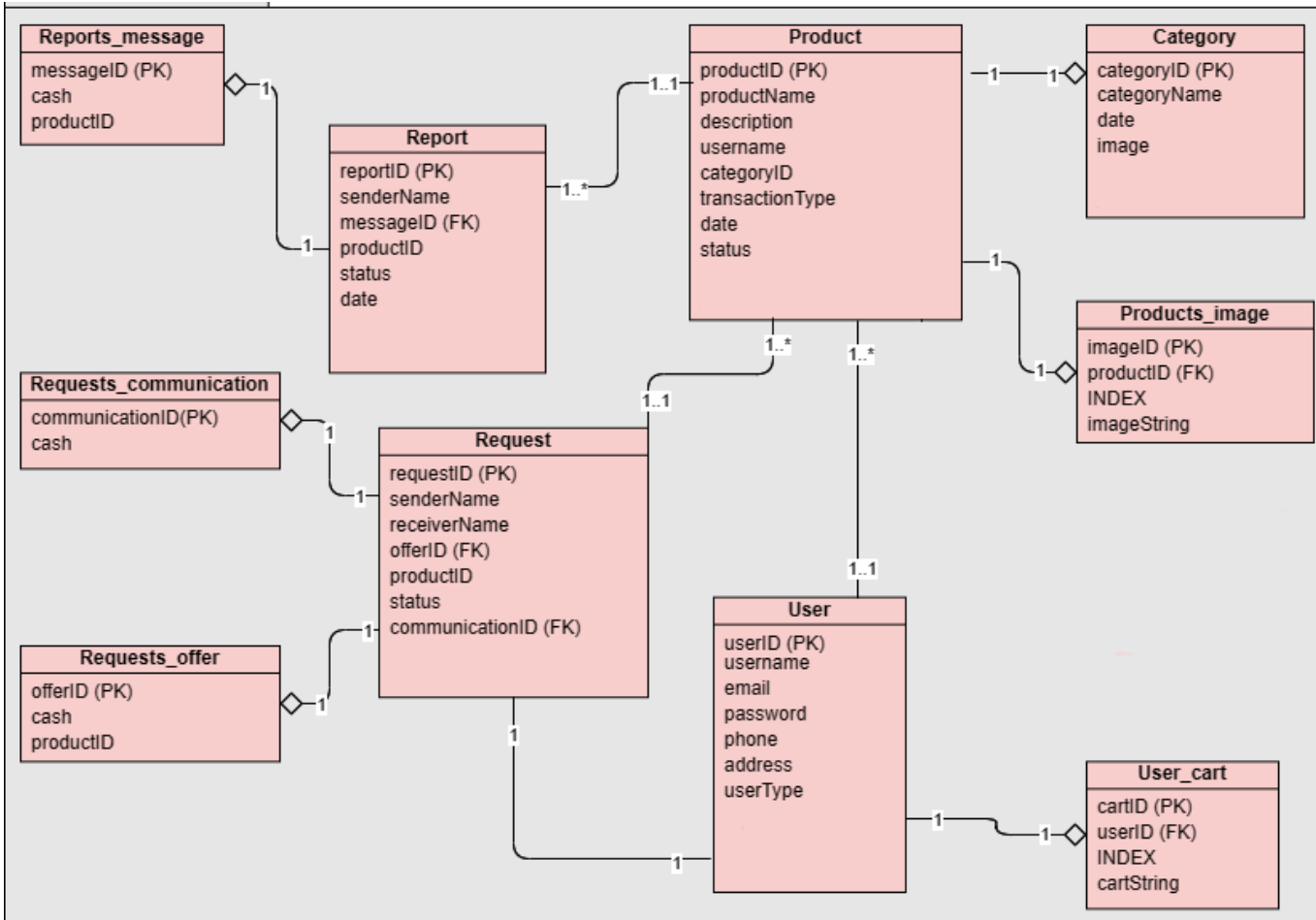


Figure 3.4.1 Class diagram

3.5 Controllers

This component plays a crucial role in the system as it acts as a mediator between the user and the system. It receives user inputs from the interfaces (Views), processes and validates those inputs, and communicates with the model to perform the necessary actions. The controller ensures that the user's inputs are correctly handled and interpreted by the system.

By following the rules provided and the given information, the controllers in the system can be designed to:

- Receive user inputs and validate them according to the defined rules and requirements.
- Interact with the model to perform operations such as creating, updating, and deleting data.
- Retrieve data from the model based on user requests.
- Apply business logic and rules to process user inputs and system responses.
- Handle errors and exceptions, providing appropriate feedback to the user.
- Communicate with the interfaces (Views) to display relevant information and receive user feedback.

Overall, the controllers component serves as the central coordinator, ensuring the smooth flow of information and actions between the user and the system.

- Admin Controller

| Admin Controller |
|--|
| login() deleteUser() deleteProduct() CreateCategory() displayCategory() removeReborts() fetchUnapprovedProducts() getAllProducts() getAllUsers() logout() |

Figure 3.5.1 Admin Controller

- User Controller

| User Controller |
|---|
| login() deleteUser() deleteProduct() updateProduct() uploadImage() MyProducts() signup() getAllProducts() UpdateProfile() logout() |

Figure 3.5.2 User Controller

Database System Description:

The database tables that will be generated are shown in the following diagram.

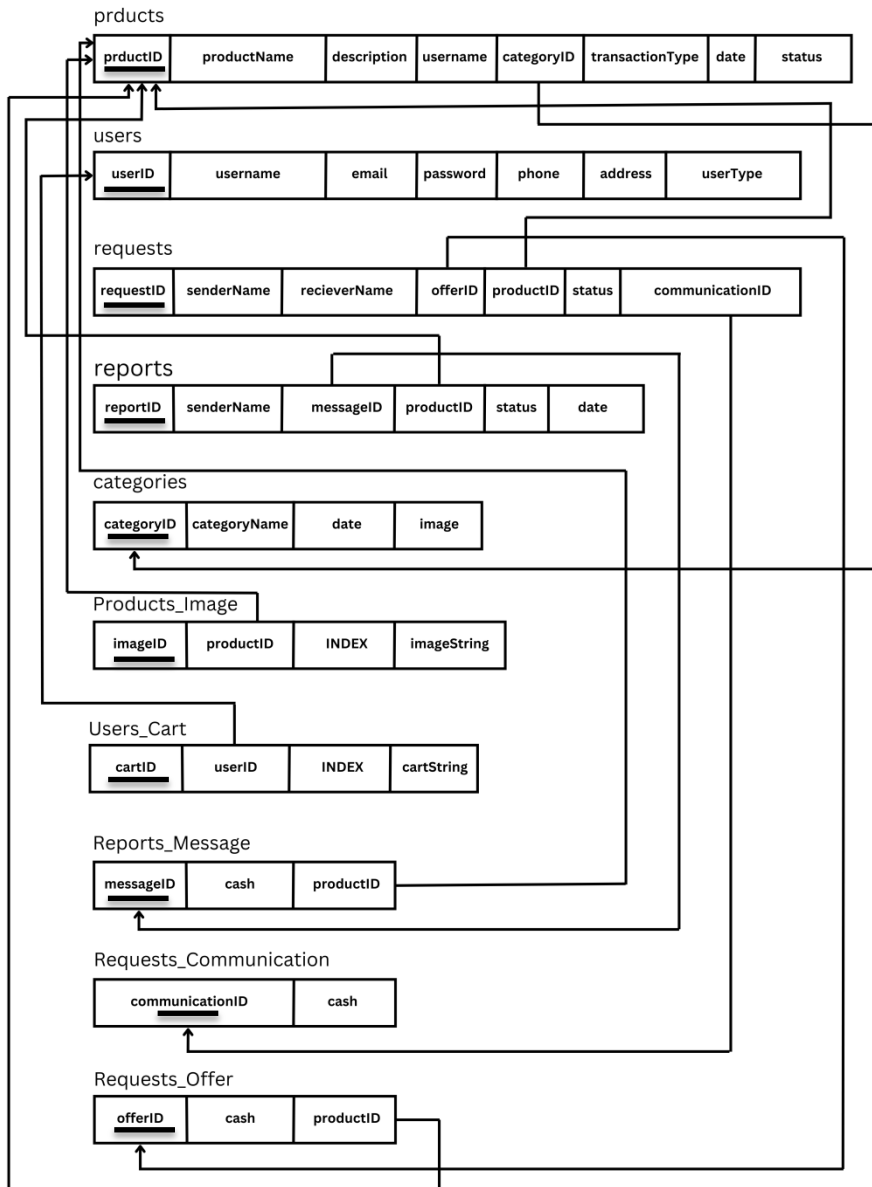


Figure 3.5.3 Database mapping

3.6 Database Tables

The system is connected to a database consisting of several tables connected by relationships. In this section, the database's information about the system's inputs and the links between its tables will be used to explain the system's many components.

Table 3.6.1 Database Table

| Name | Description |
|------------------------|--|
| User | Store information about each user of the system |
| Product | Store general information about the products available for exchange in the system. |
| Report | Store general information about the reports generated in the system. |
| Request | Store information about the requests made in the system. |
| Reports message | Store information about each message associated with a report in the system.. |
| Requests communication | Store information about the communication associated with user requests in the system. |
| Requests offer | Store information about any notifications that are sent to users. |
| Category | Store information about the categories of products |
| Products image | Store information about the images associated with products. |
| User cart | Store information about the user's cart. |

Table3.6.2 User Table

| field | Data Type | NULL | Length | Description |
|-----------------|-----------|------|--------|----------------------------------|
| userID | INTEGER | | 24 | Unique identifier for each user. |
| username | varchar | NO | 20 | user's name |
| email | varchar | NO | 50 | User's email |
| password | int | NO | 30 | user's password |
| phone | varchar | NO | 18 | user's mobile number |
| address | varchar | NO | 20 | User's address |
| userType | varchar | NO | 9 | User type |

Table 3.6.3 Product Table

| field | Data Type | NULL | Length | Description |
|------------------------|-----------|------|--------|--|
| ProductID | int | | 24 | unique identifier for each userproduct |
| ProductName | varchar | NO | 20 | Product name |
| description | varchar | NO | 300 | information about the user's product |
| username | varchar | NO | 20 | Name of the user |
| CategoryID | int | NO | 24 | Identifier for category |
| transactionType | varchar | NO | 6 | Type of transaction |
| date | date | NO | 24 | Date when the record was created |

Table 3.6.4 Category Table

| field | Data Type | NUL L | Length | Description |
|---------------------|-----------|-------|--------|----------------------------------|
| CategoryID | int | | 24 | Identifier for category |
| categoryName | varchar | NO | 20 | Name of Category |
| date | date | NO | 24 | Date when the record was created |
| image | varchar | NO | 64 | Image of Category |

Table 3.6.5 Report Table

| field | Data Type | NULL | Length | Description |
|------------|-----------|------|--------|-----------------------------------|
| reportID | int | | 24 | unique identifier for each report |
| SenderName | varchar | NO | 20 | Name of sender |
| messageID | int | NO | 24 | unique identifier for message |
| productID | int | NO | 24 | unique identifier for product |
| status | varchar | NO | 6 | Status of report |
| date | date | NO | 24 | Date when the record was created |

Table 3.6.7 products_image Table

| field | Data Type | NULL | Length | Description |
|-------------|-----------|------|--------|----------------------------------|
| imageID | int | | 24 | unique identifier for each image |
| productID | int | NO | 24 | Product id |
| index | varchar | NO | 24 | Number of image |
| imageString | varchar | NO | 64 | Name of image |

Table 3.6.8 Requests_offer Table

| field | Data Type | NULL | Length | Description |
|-----------|-----------|------|--------|----------------------------------|
| offerID | int | | 24 | unique identifier for each offer |
| cash | int | NO | 24 | Secondary data of request offer |
| productID | int | NO | 24 | unique identifier for product |

Table3.6.9 user_cart Table

| field | Data Type | NULL | Length | Description |
|--------|-----------|------|--------|---------------------------------|
| cartID | int | | 24 | unique identifier for each cart |
| userID | int | NO | 24 | unique identifier for user |
| index | varchar | NO | 24 | Number of products in cart |

Table 3.6.10 request Table

| field | Data Type | NULL | Length | Description |
|-----------------|-----------|------|--------|-------------------------------------|
| requestID | int | | 24 | unique identifier for each request |
| senderName | varchar | NO | 20 | Name of sender |
| receiverName | varchar | NO | 20 | Name of receiver |
| offerID | int | NO | 24 | unique identifier for offer |
| productID | int | NO | 24 | unique identifier for product |
| status | int | NO | 6 | Status of request |
| communicationID | int | NO | 24 | unique identifier for communication |

Table 3.6.11 requests_communication Table

| field | Data Type | NULL | Length | Description |
|-----------------|-----------|------|--------|--|
| communicationID | int | | 24 | unique identifier for each communication |
| cash | varchar | NO | 300 | Secondary data of request |

Table 3.6.12 Reports_message Table

| field | Data Type | NULL | Length | Description |
|-----------|-----------|------|--------|--|
| messageID | int | | 24 | unique identifier for each communication |
| cash | varchar | NO | 300 | Secondary data of report |
| productID | int | NO | 24 | unique identifier for product |

3.7 Views

- Login

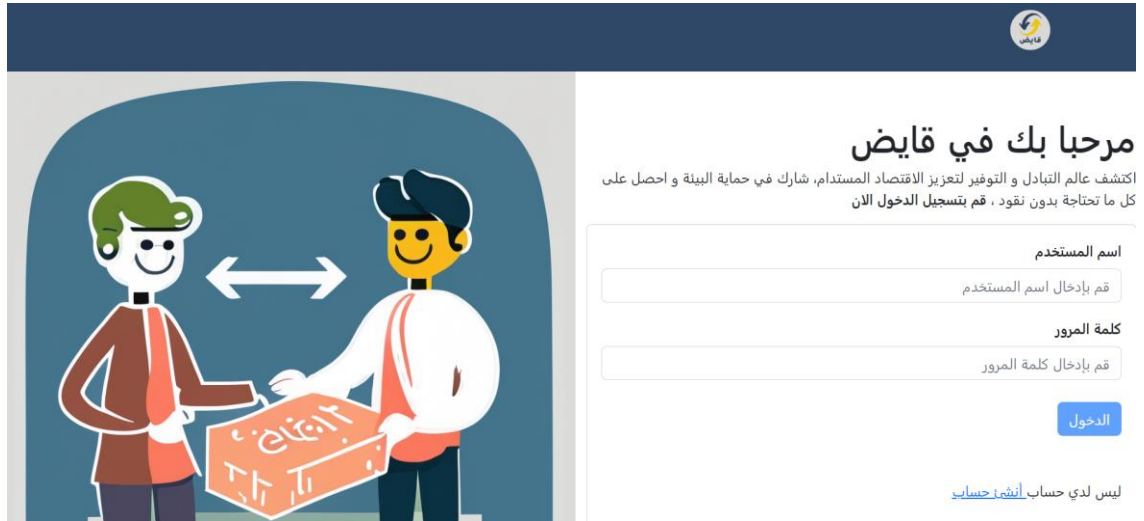


Figure 3.7.1 View Login

- Signup

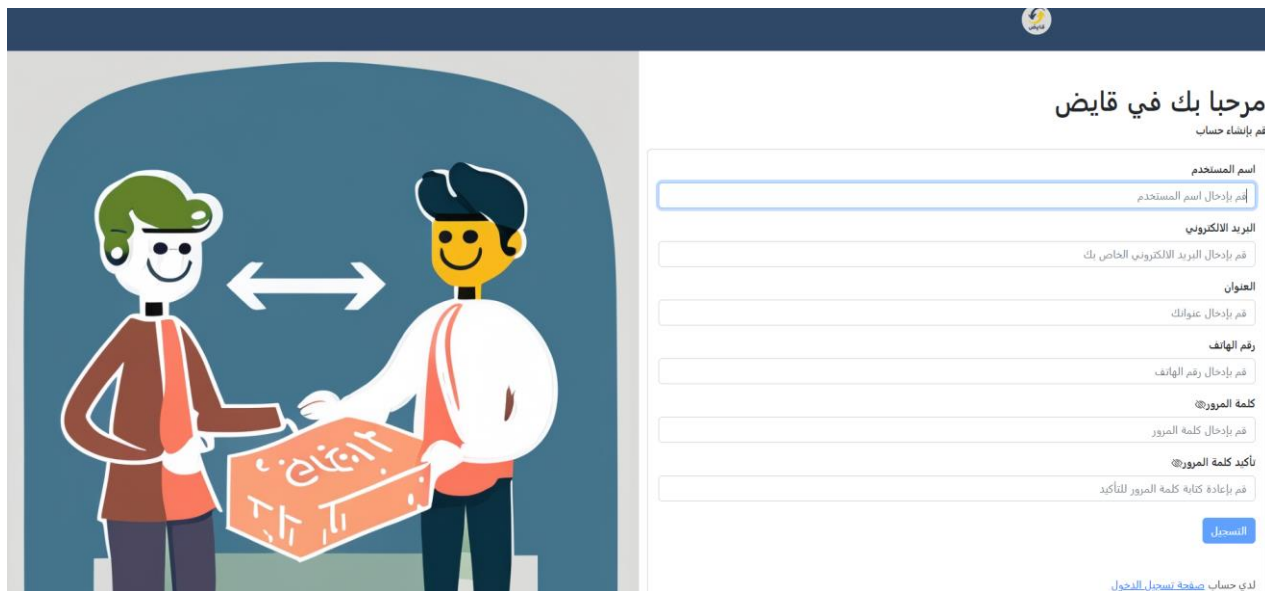
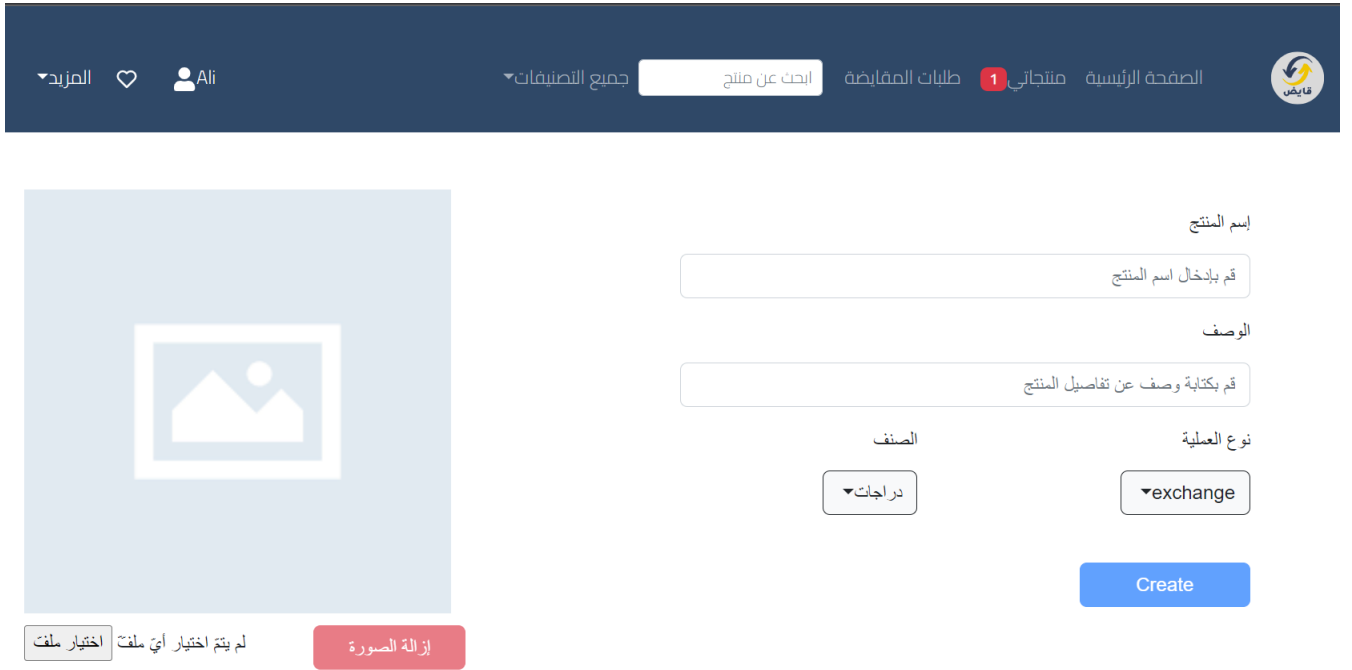


Figure 3.7.2 Signup

- Upload product



المزيد Ali الصفحة الرئيسية منتجاتي 1 طلبات المقايضة ابحث عن منتج جميع التصنيفات

اسم المنتج
قم بإدخال اسم المنتج

الوصف
قم بكتابة وصف عن تفاصيل المنتج

الصف دراجات

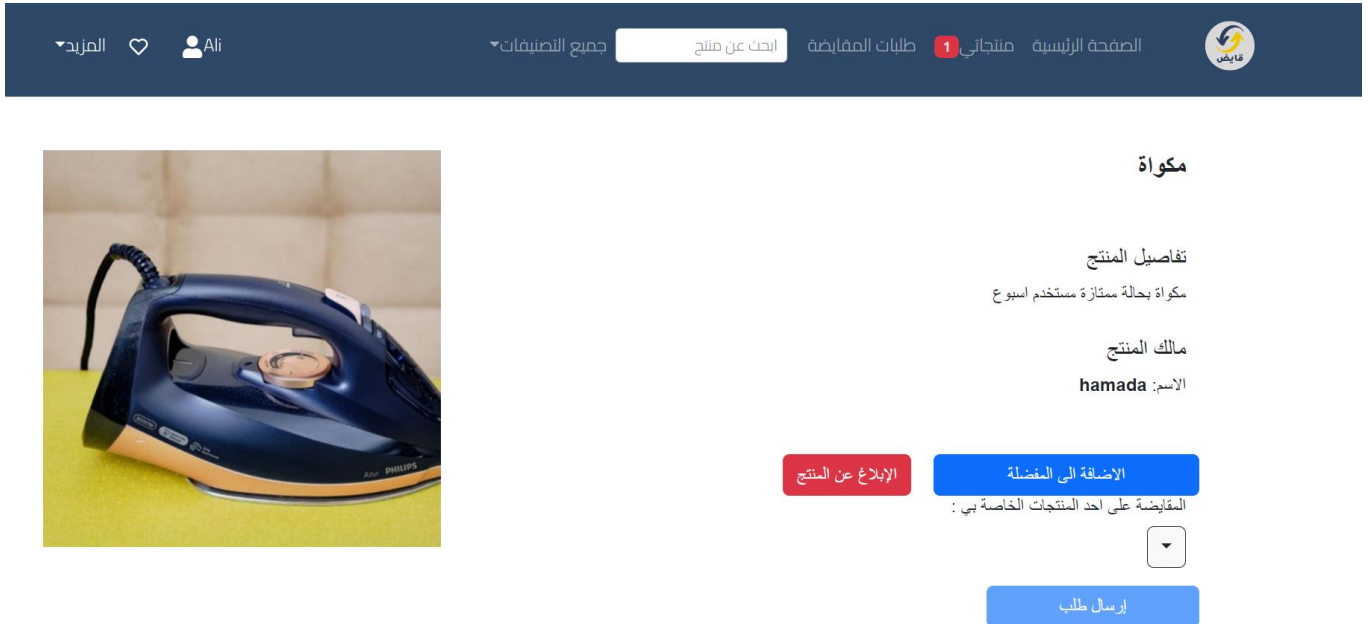
نوع العملية exchange

Create

اختيار ملف لم يتم اختيار أي ملف إزالة الصورة

Figure 3.7.3 Upload product

- Product Detail



المزيد Ali الصفحة الرئيسية منتجاتي 1 طلبات المقايضة ابحث عن منتج جميع التصنيفات

مكواة

تفاصيل المنتج
مكواة بحالة ممتازة مستخدم اسبوع

مالك المنتج
الاسم: hamada

الإبلاغ عن المنتج

الإضافة إلى المفضلة
المقايضة على أحد المنتجات الخاصة بي:

إرسال طلب

Figure 3.7.4 Product Detail

- Product list

إضافة منتج

all

المنتجات الخاصة بك (2)



| | |
|--|---|
| <p>كاميرا nikon</p> <p>تاريخ النشر: 22/5/2023</p> <p>Sold</p> <p>إزالة عرض</p> |  |
| <p>كتاب</p> <p>تاريخ النشر: 22/5/2023</p> <p>Active</p> <p>إزالة تعديل عرض طلبات</p> |  |

Figure 3.7.5 Product list

- Home Page

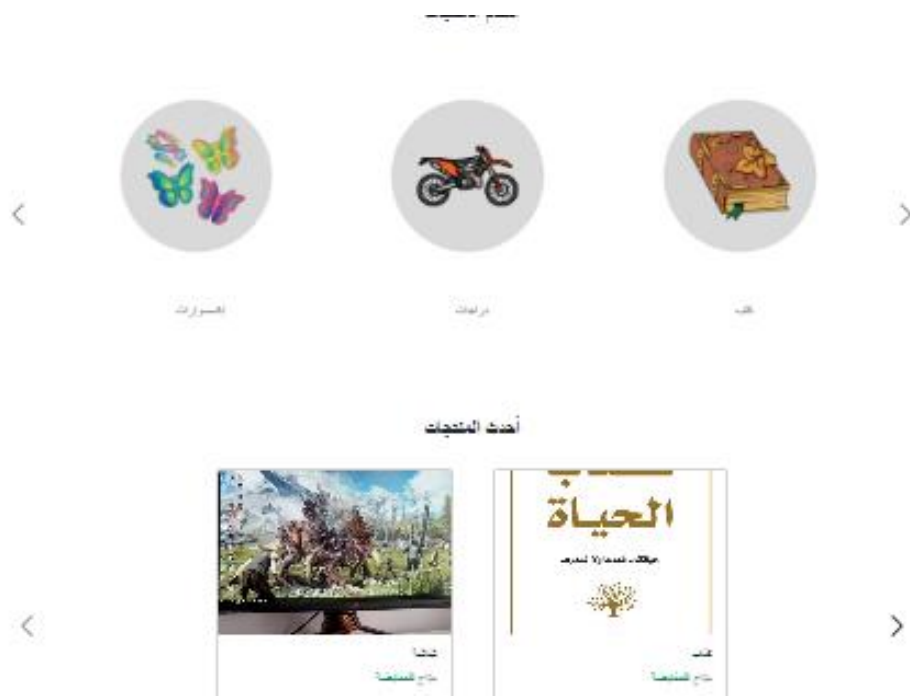


Figure 3.7.6 Home Page

Chapter 4: Implementation

4.1 Introduction

In this chapter, we will delve into the implementation phase of the system development process. Here, we will focus on the practical aspects of turning the design and requirements into a functional system. This chapter will cover various topics related to the implementation, including programming languages, frameworks, tools, and methodologies used during the development process.

4.2 Programming Languages and Technologies

During the implementation phase, we utilized the following programming languages and technologies:

- Node.js: We chose Node.js as our backend runtime environment due to its scalability and efficiency in handling concurrent requests. It allowed us to build a robust and performant backend for our system. Additionally, we designed our project to be compatible with relational databases and have already created an Oracle database.
- React.js: For the frontend development, we opted for React.js, a popular JavaScript library for building user interfaces. React.js provided us with a component-based approach, making it easier to develop and maintain the frontend of our system.
- Postman: We utilized Postman, an API development and testing tool, to test and validate our APIs during the implementation process. Postman allowed us to send requests, inspect responses, and ensure the proper functioning of our APIs.
- JSON: JSON (JavaScript Object Notation) was used as a data interchange format for exchanging data between the frontend and backend components of

our system. It provided a lightweight and easy-to-read format for data representation.

- Visual Studio Code: As our primary text editor, we relied on Visual Studio Code for writing and editing our code. Its extensive set of features, including syntax highlighting, debugging capabilities, and version control integration, greatly facilitated the development process.
- Bootstrap: To enhance the design and user interface of our system, we utilized Bootstrap, a popular CSS framework. Bootstrap provided pre-designed components and responsive layouts, allowing us to create visually appealing and user-friendly interfaces.

4.3 Implementation Approach

During the implementation phase of the project, we followed a systematic approach to bring the system design to life. Our implementation approach aimed to ensure the successful development of a functional and reliable system that meets the project requirements. Here is an overview of the implementation approach we adopted:

1. Technology Selection:

- Backend Runtime Environment: We selected Node.js as our backend runtime environment. Node.js is known for its scalability and efficiency in handling concurrent requests, making it a suitable choice for our system. Additionally, Node.js supports both relational and non-relational databases, allowing us to work with different database systems seamlessly.
- Frontend Framework: For the frontend development, we chose React.js, a popular JavaScript library for building user interfaces. React.js provided us with a component-based approach, making it easier to develop and maintain the frontend of our system.

- Database Hosting: We have made the decision to host our database on Mongo DB, a popular MySQL database known for its flexibility and scalability. However, we have also ensured that our system is compatible with relational databases, and as part of our preparations, we have already set up an Oracle database. To host the Oracle database, we have utilized Studio T3.

2. Project Setup:

- Environment Configuration: We set up the necessary development environment for both the backend and frontend. This included installing Node.js, React.js, MongoDB, and other relevant tools and dependencies.
- Folder Structure: We established a well-organized folder structure for the project, separating the backend and frontend components to ensure modularity and maintainability.
- Version Control: We utilized a version control system, such as Git, to track and manage the project's source code changes.

3. Backend Development:

- Architecture: We adopted a layered architecture for the backend, dividing it into different components such as controllers, models, and routes. This architecture promotes separation of concerns and facilitates code maintainability.
- Backend APIs: We designed and implemented Restful APIs using Node.js to enable communication between the frontend and backend components. These APIs defined the endpoints and data formats for various system functionalities.

- Database Integration: We have successfully integrated both MongoDB and Oracle databases into our system to ensure efficient storage and retrieval of data. MongoDB serves as our primary MySQL database, offering flexibility and scalability. We designed the database schema and implemented the necessary data access logic in the backend for seamless interaction with MongoDB. Additionally, we have made provisions for future developments by ensuring compatibility with relational databases, such as Oracle. We have set up an Oracle database using Studio T3, allowing us to leverage the benefits of both MySQL and relational database technologies. This integration enables us to accommodate potential expansion and cater to various data management requirements in our system.

4. Frontend Development:

- Component-Based Approach: With React.js, we developed the frontend using reusable and modular components. This approach facilitated code reusability, component composition, and efficient UI rendering.
- User Interface Design: We utilized Bootstrap, a popular CSS framework, to enhance the design and user experience of our system. Bootstrap provided pre-designed components and responsive layouts, ensuring consistency and ease of use.

5. Integration and Testing:

- Frontend-Backend Integration: We integrated the frontend and backend components, ensuring smooth communication through API calls. This integration involved testing the endpoints and handling data exchange between the two layers.

- Testing and Quality Assurance: We conducted various testing activities, including unit testing and integration testing, to verify the functionality and reliability of the system. We used tools like Postman to test and validate the APIs, ensuring their correctness and proper functioning.

6. Conclusion:

- At the end of the implementation phase, we successfully transformed the system design into a functional software solution. We achieved the desired integration between the frontend and backend, ensuring the smooth operation of the system.

4.4 Backend Development

During the backend development phase, we focused on implementing the server-side components of our system using Node.js. This section provides an overview of the architecture, folder structure, and key functionalities that were implemented.

Architecture

To ensure modularity and maintainability, we adopted a layered architecture for our backend. This architecture separates different concerns and promotes code organization. The key components of our backend architecture include:

1. **Controllers:** The controllers act as intermediaries between the frontend and the backend, handling user requests and managing the flow of data. They receive requests from the frontend, process them, and invoke the appropriate services or models to retrieve or manipulate data.
2. **Models:** The models represent the data structures and business logic of our system. They encapsulate the database operations, such as retrieving, creating, updating, and deleting data. We designed the models based on the requirements and data schema defined in the previous chapters.
3. **Routes:** The routes define the API endpoints and map them to the corresponding controller functions. They specify the URL paths, HTTP methods, and the controller methods to be invoked when a particular endpoint is requested.
4. **Middleware:** We implemented middleware functions to handle common tasks, such as authentication, request validation, and error handling. Middleware functions are executed before reaching the route handlers and allow us to add additional functionality to the request/response lifecycle.

Folder Structure

To maintain a clean and organized codebase, we structured our backend code into various folders. The folder structure ensures separation of concerns and facilitates code navigation. Our typical backend folder structure includes:

- `controllers/`: Contains the controller files that handle the business logic and interaction with the models.
- `models/`: Houses the model files responsible for database operations and data manipulation.
- `routes/`: Defines the route files that specify the API endpoints and map them to the appropriate controller functions.
- `middleware/`: Includes middleware functions used for request preprocessing and handling specific aspects of the system, such as authentication and error handling.
- `config/`: Contains configuration files, such as database configuration, environment variables, and other system settings.
- `utils/`: Includes utility functions and helper modules used across different components.

| Name | Date modified | Type | Size |
|-------------------|------------------|------------------|--------|
| .vscode | 14/05/2023 20:56 | File folder | |
| api | 14/05/2023 20:56 | File folder | |
| config | 14/05/2023 20:56 | File folder | |
| frontend | 14/05/2023 20:59 | File folder | |
| middlewares | 14/05/2023 20:56 | File folder | |
| routes | 14/05/2023 20:56 | File folder | |
| uploads | 14/05/2023 20:56 | File folder | |
| utils | 14/05/2023 20:56 | File folder | |
| .eslintrc.json | 08/04/2022 03:27 | JSON Source File | 1 KB |
| .gitignore | 20/04/2023 20:15 | Text Document | 1 KB |
| app.js | 11/05/2023 00:03 | JavaScript File | 1 KB |
| config.env | 20/04/2023 03:11 | ENV File | 1 KB |
| package.json | 30/04/2023 15:48 | JSON Source File | 1 KB |
| package-lock.json | 30/04/2023 15:48 | JSON Source File | 695 KB |
| Procfile.txt | 20/04/2023 02:40 | Text Document | 1 KB |
| server.js | 20/04/2023 03:21 | JavaScript File | 1 KB |

Figure 4.4.1 An Overview of the System's Directory Hierarchy

4.5 Frontend Development

During the frontend development phase, we focused on creating an intuitive and visually appealing user interface using React.js as our frontend framework. Leveraging the power of React.js allowed us to build a dynamic and interactive frontend that seamlessly interacts with the backend APIs.

some key aspects of our frontend development approach:

- **Component-Based Architecture:** We adopted a component-based architecture in React.js, breaking down the user interface into reusable and modular components. This approach facilitated code reusability, maintainability, and allowed for easier collaboration among team members.
- **Responsive Design:** With the increasing use of mobile devices, we prioritized responsive design to ensure that our system is accessible and provides an optimal user experience across different screen sizes and devices. We utilized CSS frameworks like Bootstrap to implement responsive layouts and pre-designed components.

```

5  .App-logo {
6    height: 40vmin;
7    pointer-events: none;
8  }
9
10 @media (prefers-reduced-motion: no-preference) {
11   .App-logo {
12     animation: App-logo-spin infinite 20s linear;
13   }
14 }
15

```

Figure 4.5.1 Implementing Responsive Design with Media Queries

- API Integration: Our frontend interacts with the backend APIs to fetch data and update the user interface dynamically. We used modern JavaScript techniques, such as asynchronous programming and promises, to handle API requests and responses effectively. Axios, a popular HTTP client, was employed to make API calls and handle data retrieval and submission.

```

9    "http://localhost:8000/api/users/getAllUser",
10   header
11  )
12
13   return response.data.data.users;
14 } catch (error) {
15   console.error("Error while fetching all users: ", error);
16   throw error;
17 }
18 };
19 export const getAllProduct = async () => {
20   try {
21     const response = await axios.get(
22       "http://localhost:8000/api/users/getAllProduct",
23       header
24     );
25
26     return response.data.data.products;
27   } catch (error) {
28     console.error("Error while fetching all products: ", error);
29     throw error;
30   }
31 };
32

```

Figure 4.5.2 API Requests for Data Retrieval from Backend

- **Form Validation:** To enhance user experience and data integrity, we incorporated form validation techniques on the frontend to validate user inputs, and provide meaningful error messages.
- **Cross-Browser Compatibility:** We ensured that our frontend is compatible with major web browsers, including Chrome, Firefox, Safari, and Edge. We conducted extensive testing and utilized tools like Browser Stack to ensure consistent behavior and appearance across different browsers.

Throughout the frontend development phase, we followed modern best practices, adhered to coding standards, and conducted rigorous testing to deliver a robust and user-friendly frontend for our system. We strived to create a seamless and intuitive user experience while maintaining scalability and performance.

4.6 Overview of the Chapter

In this chapter, we delved into the implementation phase of the system development process. We covered various aspects related to the practical implementation of the system, including programming languages, technologies, and methodologies used during the development process.

We started by discussing the programming languages and technologies we employed in our project. Node.js was chosen as the backend runtime environment due to its scalability and efficiency in handling concurrent requests. React.js was utilized for frontend development, providing us with a component-based approach for building user interfaces. We also mentioned the use of Postman for API testing, JSON as the data interchange format, Visual Studio Code as the text editor, and Bootstrap for enhancing the design.

Next, we explored the implementation approach we followed, which involved technology selection, project setup, backend development, frontend development, integration and testing. We highlighted the importance of selecting the right technologies, configuring the development environment, and setting up the project

structure. We also discussed the architecture, folder structure, and key functionalities implemented in the backend and frontend development phases.

Additionally, we emphasized the significance of integration and API development, including the identification of integration points, design of integration interfaces, implementation of integration logic, and error handling mechanisms. We explained the concept of RESTful APIs and provided an example of error handling in the integration process.

Furthermore, we covered the API specification and how it documents the available endpoints, request/response formats, authentication mechanisms, and error handling. We also discussed the significance of responsive design in ensuring the system's accessibility and optimal user experience across different devices.

Finally, we provided an overview of the chapter, summarizing the key points discussed in each section. We highlighted the importance of transforming the system design into a functional software solution and the successful integration between the frontend and backend components. The implementation phase laid the foundation for subsequent testing, deployment, and future developments.

Overall, this chapter focused on the practical implementation aspects of the system, providing insights into the programming languages, technologies, implementation approach, integration, API development, and frontend development.

Chapter 5: Testing and Validation

5.1 Introduction

In this chapter, we will explore the crucial phase of testing and validation in the system development process. Testing and validation are essential to ensure the quality, functionality, and reliability of the developed system. This chapter will cover various topics related to testing methodologies, test planning, test execution, and validation techniques.

5.2 Testing Methodologies

In the testing phase, we employed a combination of manual and automated testing methodologies to thoroughly assess the system's performance and functionality. We followed industry-standard testing methodologies, including:

1. **Unit Testing:** We conducted unit tests to verify the functionality of individual components, such as functions, methods, and modules. Unit testing helped us identify and fix bugs at an early stage, ensuring the reliability of our system.

- **Example for Unit Testing:**

For the unit testing phase, we focused on testing individual components of our system. As an example, we conducted unit tests for a specific module responsible for user authentication. We created test cases to verify the functionality of functions and methods within this module, such as user registration, login, and password reset. By simulating different scenarios and inputs, we were able to identify and fix any bugs or issues at an early stage, ensuring the reliability and correctness of the authentication module.

2. **Integration Testing:** Integration testing was performed to validate the interaction and compatibility of different system components. We tested the integration of frontend and backend components, APIs, and database connectivity to ensure smooth communication and data exchange.

- **Example for Integration Testing:**

During the integration testing phase, we aimed to ensure the seamless interaction and compatibility of different system components. As an example, we tested the integration between the frontend and backend components of our system. We simulated various user interactions on the frontend, such as submitting a form or making a request, and validated that the data was correctly transmitted to the backend via APIs. We also checked the database connectivity and verified that data was properly stored and retrieved. This integration testing helped us identify any issues or inconsistencies in the communication between the frontend and backend, ensuring smooth data exchange and system functionality.

3. System Testing: System testing focused on evaluating the overall behavior and functionality of the complete system. We tested various use cases and scenarios to ensure that the system met the defined requirements and provided the expected outcomes.

- **Example for System Testing:**

System testing was crucial to evaluate the overall behavior and functionality of our complete system. As an example, we created a comprehensive set of test cases that covered different use cases and scenarios. For instance, we tested the system's ability to handle concurrent user requests, ensuring that it maintained stability and provided the expected outcomes. We also validated the system against the defined requirements, verifying that all functionalities, such as user management, product listings, and transaction processing, were working correctly. System testing provided us with confidence in the system's functionality, reliability, and its ability to meet the user's needs

4. Performance Testing: Performance testing was conducted to assess the system's responsiveness, scalability, and resource utilization under different load conditions. We measured response times, analyzed system bottlenecks, and optimized performance to enhance user experience.

- **Example for Performance Testing:**

To assess the system's performance, we conducted performance testing under various load conditions. For example, we simulated a high number of concurrent user requests to measure the system's responsiveness and scalability. We monitored response times, CPU and memory utilization, and database query execution times. By analyzing the performance metrics, we were able to identify potential bottlenecks, optimize the system's resource usage, and enhance its overall performance. This performance testing ensured that our system could handle the expected user load and provide a satisfactory user experience.

5. User Acceptance Testing (UAT): UAT involved involving end-users to validate the system against their requirements and expectations. Feedback from end-users helped us identify any usability issues and make necessary improvements.

- **Example for User Acceptance Testing (UAT):**

User Acceptance Testing (UAT) involved involving end-users to validate the system against their requirements and expectations. As an example, we invited a group of representative users to test the system's functionality and provide feedback. We provided them with specific test scenarios and observed their interactions with the system. Their feedback helped us identify any usability issues, user interface glitches, or missing features that needed improvement. Incorporating user feedback through UAT ensured that the system met the users' expectations and provided a user-friendly experience.

The following tables provide examples of some of the tests we performed on the application side, as it is the part that interacts with the user:

| # | Process | Type | Input | Expected Output | Actual Output | Success/Failure |
|---|----------------------------------|------|---|---|---|-----------------|
| 1 | post_Signup /api/users/signup | POST | username: user_1 email: email_1@gmail.com password: Aa123456 phone: 0595303030 address: dir samet userType: Action | The user has been added successfully | status: "0" code: "0" message: "no error." data: username: user_1 email: email_1@gmail.com userType: normal | Success |
| 2 | post_Signup /api/users/signup | POST | username: user_1 email: email_1@gmail.com password: Aa1234 phone: 0595303030 address: dir samet userType:Action | There was an error when registering, the user was not added | status: "1" code: "3" message: "Email or password not correct." | Failure |
| 3 | post_Login /api/users/login | POST | username: user_1 password: Aa123456 | Logged in successfully | status: "0" code: "0" message: "no error." userId: "643d715b2ae393c8d85941be" username: user_1 userType: normal token:"..." | Success |
| 4 | post_Login /api/users/login | POST | username: user_1 password: Aa12345 | User does not exist or there is an error | status: "1" code: "6" message: "Incorrect password." | Failure |

5.2.1 Test: Add New User And Login

| # | Process | Type | Input | Expected Output | Actual Output | Success/Failure |
|---|---|------|--|---|---|-----------------|
| 1 | post_create_new /api/products/create_new | POST | username: user_1 productName: product_1 description: any description_1 transactionType: exchange image: file.jpg categoryId: "6456c463b27d96d06f232f04" | The product has been added successfully | status: "0" code: "0" message: "no error." data productName: product_1 description: any description_1 username: user_1 categoryId: "6456c463b27d96d06f232f04" transactionType: exchange image: ["http://127.0.0.1:8000/products/product-2e82c894-a3d1-4c23-b13f-80deca63cb25-1684781983410.jpeg"], date: 2023-05-22T18:59:43.462Z status: pending requests:[], productId: 646bbb9f6a087a80b359808a "__v": 0 | Success |

5.2.2 Test: Add New Product

| # | Process | Type | Input | Expected Output | Actual Output | Success/ Failure |
|---|--|------|-------|---|---|---------------------|
| 1 | get_fetch all product /api/products/fe tch_all_product | GET | — | Bring all the special products of the current session owner | status: "0" code: "0" message: "no error." data: products[productId: "64693d6fae849a8d2e64e92a" productName: product_1 description: any description_1 categoryId: "6469284c0f6816c7edf1844d", transactionType: exchange date: 2023-05-20T21:36:47.144Z status: active] | Success |

5.2.3 Test: Get All User Product

| # | Process | Type | Input | Expected Output | Actual Output | Success/Failure |
|---|---|------|---|---|---|-----------------|
| 1 | post_create request /api/requests/create_new | POST | productId: "6446538dfe2ea896ef9dc643" senderName: mohammad recieverName: ahmood offer:{ cash: Any text, productId: "6446538dfe2ea896ef9dc643" } communication:{ "cash": "0595142222" } | The exchange request has been sent successfully | status: "0" code: "0" message: "no error." data: request_generation_confirmation: true | Success |
| 2 | post_create request /api/requests/create_new | POST | productId: "645a5ae938b9b6af67cdde68" senderName: mohammad recieverName: ahmood offer:{ cash: Any text, productId: "645a5ae938b9b6af67cdde68" } communication:{ "cash": "0595142222" } | The request was not sent for exchange because the product was exchanged | status: "0" code: "0" message: "no error." data: request_generation_confirmation: false | Failure |

5.2.4 Test: Create New Request

| # | Process | Type | Input | Expected Output | Actual Output | Success/Failure |
|---|---|------|-------|-------------------------------------|---|-----------------|
| 1 | get_get all user (admin) /api/users/getAllUser | GET | – | Bring all users on the Qaied system | status: "0" code: "0" message: "no error." data: users:[{ userId: "643d715b2ae393c8d85941be" username: mohammad email: mohammad2021@gmail.com phone: 0599996655 address: 0052 }, { userId: "643da6fd41abba84fe39356a", username: mohammadhroubv6 email: mohammad22030@gmail.com phone: 972595303030 address: 3021 }] | Success |

5.2.5 Test: Get All Users

| # | Process | Type | Input | Expected Output | Actual Output | Success/Failure |
|---|---|------|---|-----------------------|--|-----------------|
| 1 | post_approve (admin) /api/products/approve | POST | action: approved productId: "60a12d7f39ae2c21408341d8" | Approval of a product | status: "0" code: "0" message: "no error." data: acknowledged: true modifiedCount: 1 upsertedId: null upsertedCount: 0 matchedCount: 1 | Success |

5.2.6 Test: Approve Product

5.3 Validation Techniques

In addition to testing, we employed validation techniques to ensure that the system met the intended purpose and satisfied the requirements of stakeholders. These techniques included:

1. Requirement Validation:

Objective:

The goal of requirement validation is to review and validate the system requirements to ensure they are complete, accurate, and aligned with the project's objectives.

process:

- Requirement review: Thoroughly examine the documented requirements, ensuring they are clear, feasible, and relevant to the project goals.
- Stakeholder feedback: Seek input from relevant stakeholders, including yourself, to validate the requirements and ensure they effectively address the business needs.

Example: During a requirement review, we carefully assessed the requirement stating "The system should allow users to register using their email and password." This validation confirms its alignment with the project's vision of enabling user registration through email authentication.

2. User Validation:

Objective:

User validation involves engaging end-users to gather their feedback and ensure the system meets their needs and expectations.

process:

- User acceptance testing (UAT): Involve a representative group of end-users to test the system in a controlled environment, following predefined scenarios, and evaluate its performance, usability, and functionality.
- User feedback sessions: Conduct structured feedback sessions with end-users to gather their insights, suggestions, and concerns regarding the system's usability and user experience.

Example: During UAT, selected end-users perform tasks within the system, such as exchanging products. Their feedback helps evaluate the ease of use, clarity of instructions, and overall satisfaction with the process.

3. Data Validation:

Objective:

Data validation ensures the accuracy, integrity, and consistency of data stored and processed by the system.

Process:

- Data reconciliation: Compare data in the system with trusted data sources or existing records to identify discrepancies or inconsistencies.
- Validate data formats and constraints: Verify that data inputs adhere to specified formats, meet required constraints, and follow predefined rules or business logic.

Example: Data validation includes verifying the uniqueness of usernames during the user registration process. When a new user attempts to register, the system checks if the chosen username is already taken by comparing it with existing usernames in the database. If the username is already in use, the system prompts the user to select a different username to maintain uniqueness and avoid conflicts.

5.4 Overview of the Chapter

Chapter 5 focused on the crucial phase of testing and validation in the system development process. We employed a combination of testing methodologies and validation techniques to ensure the quality, functionality, and reliability of our system. The chapter covered various topics related to testing, including testing methodologies, validation techniques, and examples of their application in our project.

We began by discussing the different testing methodologies we utilized, including unit testing, integration testing, system testing, performance testing, and user acceptance testing (UAT). Each methodology served a specific purpose and helped us assess different aspects of the system. We provided clear objectives, processes,

and examples for each testing methodology, demonstrating how we applied them in our project.

Next, we explored validation techniques that ensured the system met the intended purpose and satisfied stakeholders' requirements. We discussed requirement validation, user validation, and data validation techniques. For each technique, we outlined the objectives, processes, and provided examples of how we applied them in our project. These validation techniques helped us validate the system's requirements, gather user feedback, and ensure data accuracy and integrity.

Chapter 6: Future Plans

6.1 Introduction

In this chapter, we will explore the future plans for the system and outline our vision for its further development and enhancement. As technology and business needs evolve, it is crucial to have a roadmap for future improvements, scalability, and innovation. This chapter will cover our proposed future plans, including potential features, upgrades, and considerations for expanding the system's capabilities.

6.2 System Enhancements and Feature Expansion

To ensure the long-term success and relevance of the system, we have identified several areas for future enhancements and feature expansion. These enhancements aim to address user feedback, industry trends, and emerging technologies. Some of the key areas we plan to focus on include:

- **Enhanced User Experience:** We aim to further improve the user experience by refining the system's interface, streamlining workflows, and incorporating user feedback. This may involve implementing a more intuitive navigation system, optimizing loading times, and personalizing user interactions.
- **Advanced Search and Filtering:** To enhance product discovery and facilitate efficient searching, we plan to introduce advanced search and filtering capabilities. This may include options such as search by category, price range, location, and product attributes. These enhancements will enable users to find relevant products more quickly and easily.
- **Social Media Integration:** Recognizing the influence of social media in today's digital landscape, we plan to integrate social media functionalities into the system. This may involve allowing users to share products on their social media profiles, implementing social login options.

- **Mobile Application Development:** With the increasing popularity of mobile devices, we plan to develop a dedicated mobile application to provide a seamless and optimized user experience on smartphones and tablets. The mobile application will offer all the core functionalities of the web-based system, ensuring accessibility and convenience for users on-the-go.

6.3 Scalability and Performance Considerations

As the system gains traction and user base grows, scalability and performance become critical considerations. To ensure the system can handle increased demand and maintain optimal performance, we plan to implement the following strategies:

- **Server Load Balancing:** To distribute incoming requests evenly and prevent overload on a single server, we will introduce server load balancing techniques. This will help improve system availability, scalability, and responsiveness.
- **Caching Mechanisms:** Implementing caching mechanisms, such as content caching and database query caching, will help reduce database load and improve response times. This will enhance system performance, especially during peak usage periods.
- **Database Optimization:** As the system accumulates large amounts of data, we will continually optimize the database performance by fine-tuning queries, indexing data, and implementing database partitioning strategies. These optimizations will ensure efficient data retrieval and storage.

6.4 Collaboration and Feedback Channels

To ensure continuous improvement and address evolving user needs, we plan to establish collaboration and feedback channels with users, stakeholders, and the development team. This will involve:

- **User Feedback Portal:** Implementing a user feedback portal or system within the application, allowing users to provide suggestions, report issues, and share their experiences. This feedback will be carefully reviewed and considered in future updates and enhancements.
- **Stakeholder Meetings:** Regular meetings and communication with stakeholders, including us as the project owners, to discuss system performance, future requirements, and strategic alignment. These meetings will facilitate a shared understanding of goals and enable effective decision-making.
- **Agile Development Practices:** Embracing agile development practices, such as iterative development and continuous delivery, will allow us to respond quickly to changing requirements and incorporate user feedback into regular development cycles.

6.5 Overview of the Chapter

In this chapter, we have explored the future plans for the system, outlining our vision for its further development and enhancement. We recognize the importance of staying ahead of technological advancements and evolving business needs to ensure the long-term success and relevance of the system. The chapter focuses on system enhancements and feature expansion, scalability and performance considerations, collaboration and feedback channels, and the utilization of agile development practices.

To enhance the user experience, we plan to improve the system's interface, streamline workflows, and incorporate user feedback. Advanced search and filtering capabilities will be introduced to facilitate efficient product discovery. Social media

integration will leverage the influence of social platforms to enhance user engagement and expand the system's reach. Additionally, a dedicated mobile application will be developed to provide an optimized experience on smartphones and tablets, ensuring accessibility and convenience.

Scalability and performance considerations will be addressed through server load balancing techniques, caching mechanisms, and ongoing database optimization. These strategies aim to handle increased demand, improve response times, and optimize data storage and retrieval.

To ensure continuous improvement, collaboration, and user satisfaction, we will establish collaboration and feedback channels. A user feedback portal will be implemented to gather suggestions, report issues, and collect user experiences. Regular stakeholder meetings will facilitate communication and alignment with project owners, enabling effective decision-making. Embracing agile development practices will allow us to adapt quickly to changing requirements and incorporate user feedback into regular development cycles.

By implementing these future plans, we aim to ensure the system's long-term success, maintain user satisfaction, and adapt to emerging technologies and industry trends. These plans reflect our commitment to continuous improvement, innovation, and delivering a system that evolves with the changing needs of users and stakeholders.

References:

(1) MVC Architecture in 5 minutes: a tutorial for beginners. (n.d.). Educative. <https://www.educative.io/blog/mvc-tutorial>

(2) Pedriquez, D. (n.d.). What is a Context Diagram (and How Can You Create One)? - Venngage. Venngage. <https://venngage.com/blog/context-diagram/>

(3) What is Sequence Diagram? (n.d.). <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>

(4) Introduction | React Bootstrap. (n.d.). <https://react-bootstrap.github.io/docs/getting-started/introduction>

(5) Documentation | Node.js. (n.d.). Node.js. <https://nodejs.org/en/docs>

(6) UML Use Case Diagram Tutorial. (n.d.). Lucidchart. <https://www.lucidchart.com/pages/uml-use-case-diagram>

(7) Layered architecture | learning-notes. (n.d.). <https://learning-notes.mistemicheels.com/architecture-design/reference-architectures/layered-architecture/>

(8) UML Class Diagram Tutorial. (n.d.). Lucidchart . <https://www.lucidchart.com/pages/uml-class-diagram>

(9) Express Tutorial Part 4: Routes and controllers - Learn web development | MDN. (n.d.). https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

(10) Best Practice for Node.js Folder Structure. (n.d.). Habilelabs Private Limited. <https://habilelabs.io/blog/best-practice-for-node-js-folder-structure>

(11) Front-End Development: The Complete Guide. (n.d.). Cloudinary. <https://cloudinary.com/guides/front-end-development/front-end-development-the-complete-guide>

(12) What is Software Testing and How Does it Work? | IBM. (n.d.). <https://www.ibm.com/topics/software-testing>

(13) Harrison, P. (2020, December 3). Caching strategies to speed up your API - LogRocket Blog. LogRocket Blog. <https://blog.logrocket.com/caching-strategies-to-speed-up-your-api/>