**Project Report**

**Motivational Quotes- Android & iOS Application**

Submitted by

Hammad Ahmad

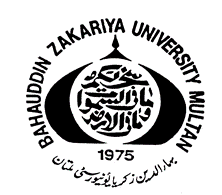
BSIT(M)-20-56

2020-2024

Supervised By

Dr. Maroof Pasha

[Document title]



## DEPARTMENT OF INFORMATION TECHNOLOGY

**BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN PAKISTAN**

**FINAL APPROVAL**

This is to certify that we have read this report **Motivational Quotes- Android & iOS Application** submitted by ***Hammad Ahmad BSIT(M)-20-56*** and it is our judgment that this report is of sufficient standard to warrant its acceptance by Bahauddin Zakariya University, Multan for the degree of BS (IT) / MIT (Master of Information Technology).

# ***Committee:***

**1. External Examiner \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2. Supervisor**

Dr. Maroof Pasha  
Head of Department

Department of Information Technology, **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Bahauddin Zakariya Multan

.

**3. Head of Department \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Dr. Maroof Pasha

Head of Department,

Department of Information Technology

Bahauddin Zakariya Multan

**DEDICATION**

***To my Loving Parents***

**ACKNOWLEDGMENT**

All the acclamation and appreciation are for Almighty ALLAH, the most beneficent and the most merciful. I am very grateful to ALLAH who made me able to complete the work presented here. It is due to His unending mercy that this work moved towards success.

With humble and profound sense of devotion, I express my most sincere thanks to **Dr. Maruf Pasha,** Head of Department of Information Technology, BZU Multan for his valuable guidance and support.

My sincere gratitude to project supervisor **Dr. Maruf Pasha,** Head of Department of Information Technology, Department of Information Technology, BZU Multan for having faith in me and thus allowing me to carry out a project on a technology completely new to me. He helped immensely by guiding me throughout the course of the project, inspiring me to take up new challenges along the road, and at the same time providing valuable suggestions & constructive criticisms.

I would also thank **Mr. Abbas Hassan,** Department of Information Technology, BZU Multan for giving me support on this project and giving me the confidence that I can do it.

**Hammad Ahmad**

**PROJECT BRIEF**

|  |  |
| --- | --- |
| PROJECT NAME | Motivational Quotes- Android & iOS Application |
| UNDERTAKEN BY | Hammad Ahmad |
| SUPERVISED BY | Dr. Maruf Pasha |
| STARTING DATE | April 01, 2024 |
| COMPLETION DATE | June 01, 2024 |
| COMPUTER USED | Core i5, 4.9 Ghz,8GB RAM, 1TB Hard disk+225 SSD |
| OPERATING SYSTEM | MS Windows 10 64-bit/Mac OS |
| SOURCE LANGUAGE(S) | Dart (Android) |
| DBMS USED | Firebase, Firestore |
| TOOLS/PACKAGES | Android Studio, VS code, MS Office, Xbox |

**ABSTRACT**

"Quotez" is a smartphone software that curates quotes from a variety of sources to provide users daily doses of inspiration and encouragement. People frequently find themselves in need of a positive boost in today's fast-paced environment in order to overcome obstacles and pursue their goals with newfound zeal. The purpose of "Quotez" is to meet this demand by providing users with motivational quotes that speak to them personally.

**Table of Contents**

[DEPARTMENT OF INFORMATION TECHNOLOGY 1](#_Toc168493456)

[***Committee:*** 2](#_Toc168493457)

[Motivational Quotes- Android & iOS Application 5](#_Toc168493458)

[**Chapter 01: INTRODUCTION** 12](#_Toc168493459)

[1.1 Introduction of Android operating system 13](#_Toc168493460)

[**1.1.1 Technology Features** 13](#_Toc168493461)

[**1.1.2. Android Versions till Date** 15](#_Toc168493462)

[1.2 Project introduction 15](#_Toc168493463)

[**1.2.1 Main Theme** 15](#_Toc168493464)

[**1.2.2 Scope of the Project** 15](#_Toc168493465)

[**1.2.3 Objectives of the Project** 16](#_Toc168493466)

[**1.2.4 Thesis Organization** 16](#_Toc168493467)

[**1.2.5 Summary** 17](#_Toc168493468)

[**Chapter 02: System Analysis** 18](#_Toc168493469)

[2.1 Feasibility Study 19](#_Toc168493470)

[**2.1.1 Technical Feasibility** 19](#_Toc168493471)

[**2.1.2 Economic Feasibility** 19](#_Toc168493472)

[**2.1.3 Operational Feasibility** 20](#_Toc168493473)

[**2.1.4 Legal Feasibility** 20](#_Toc168493474)

[2.2 Existing System: Data Gathering 20](#_Toc168493475)

[**2.2.1 Questionnaires** 21](#_Toc168493476)

[**2.2.2 Sampling & Observations** 21](#_Toc168493477)

[2.3 Existing System: Data Analysis 22](#_Toc168493478)

[**2.3.1 Data Flow Diagrams (DFDs)** 22](#_Toc168493479)

[**2.3.2 Requirements Engineering** 24](#_Toc168493480)

[**2.3.3 Deliverables** 25](#_Toc168493481)

[**Chapter 03: System Design** 27](#_Toc168493482)

[3.1 Introduction to System Design 28](#_Toc168493483)

[3.2 System Design using UML 28](#_Toc168493484)

[**3.2.1 Use Case Diagrams** 28](#_Toc168493485)

[**3.2.2 Sequence Diagrams** 28](#_Toc168493486)

[**3.2.3 Class Diagram** 30](#_Toc168493487)

[3.3 Database Design 30](#_Toc168493488)

[**3.3.1 Requirement Analysis** 32](#_Toc168493489)

[**3.3.2 Conceptual Database Design (Data Modeling)** 33](#_Toc168493490)

[**3.3.3 Logical Database Design** 33](#_Toc168493491)

[**3.3.4 Physical Database Design** 34](#_Toc168493492)

[3.4 Entity Relationship Diagrams (ERDs) 34](#_Toc168493493)

[3.5 Building a Relational Model from ERDs 35](#_Toc168493494)

[**Steps to Build a Relational Model:** 35](#_Toc168493495)

[**Relational Model for Quoteza:** 35](#_Toc168493496)

[3.6 Normalizing Relational Model to 3NF 36](#_Toc168493497)

[**Normalization Steps:** 36](#_Toc168493498)

[**Chapter 04: System Development** 37](#_Toc168493499)

[4.1 Introduction to System Development 38](#_Toc168493500)

[4.2 Tool/Language Selection 38](#_Toc168493501)

[**4.2.1 Programming Languages and Frameworks**: 38](#_Toc168493502)

[**4.2.2 Backend Services**: 38](#_Toc168493503)

[**4.2.3 APIs**: 38](#_Toc168493504)

[**4.2.4 Development Tools**: 38](#_Toc168493505)

[**4.2.5 Version Control**: 38](#_Toc168493506)

[4.3 Hardware for the System 39](#_Toc168493507)

[**4.3.1 Development Hardware**: 39](#_Toc168493508)

[**4.3.2 Server Hardware**: 39](#_Toc168493509)

[4.4 Software Development & Implementation 39](#_Toc168493510)

[**4.4.1 Development Methodology**: 39](#_Toc168493511)

[**4.4.2 Development Phases**: 39](#_Toc168493512)

[**4.4.3 Implementation Details**: 39](#_Toc168493513)

[**4.4.4 Client-Side Technology** 40](#_Toc168493514)

[**4.4.5 Server-Side Technology** 41](#_Toc168493515)

[4.5 Code/Algorithms of Important Modules 41](#_Toc168493516)

[**4.5.1 User Authentication Module** 41](#_Toc168493517)

[**4.5.2 Quote Management Module** 43](#_Toc168493518)

[**4.5.3 API Integration Module** 44](#_Toc168493519)

[**Chapter 05: Testing and Quality Assurance** 46](#_Toc168493520)

[5.1 Testing Process 47](#_Toc168493521)

[5.2 Test Case Design 47](#_Toc168493522)

[5.3 Test Data 48](#_Toc168493523)

[5.4 Black Box Testing (Behavioral Testing) 48](#_Toc168493524)

[5.5 White Box Testing (Structural Testing) 49](#_Toc168493525)

[**Chapter 06: User Guide** 51](#_Toc168493526)

[6.1 Input Forms 52](#_Toc168493527)

[**6.1.1 User Registration Form** 52](#_Toc168493528)

[**6.1.2 User Login Form** 54](#_Toc168493529)

[**6.1.3 Favorite Quote Form** 56](#_Toc168493530)

[**6.1.4 Profile Management** 58](#_Toc168493531)

[**6.1.5 Home Screen** 60](#_Toc168493532)

[**6.1.6 Subscription Screen** 62](#_Toc168493533)

[**6.1.7 Reminders Screens** 64](#_Toc168493534)

[**Chapter 07: Conclusion** 67](#_Toc168493535)

[Project Summary: Development of Quoteza App 68](#_Toc168493536)

[**Key Tasks and Activities** 68](#_Toc168493537)

**List of figures**

[Figure 1. 1 Android Operating System Architecture 13](#_Toc168246194)

[**Figure 3.3. 1 Steps For Database Design** 26](#_Toc168246204)

[**Figure3.4. 1 ER Diagram of Quoteza App** 29](#_Toc168246212)

[**Figure 2.3.1. 1 DFD Level Zero** 23](#_Toc168493538)

[**Figure 2.3.1. 2 DFD Level 1 diagram** 24](#_Toc168493539)

[**Figure 3.2.2. 1 Sequence Diagram of Quoteza** 29](#_Toc168493540)

[**Figure 3.2.3. 1 Class Diagram of Quoteza** 30](#_Toc168493547)

[**Figure 3.3. 1 Steps For Database Design** 32](#_Toc168493566)

[Figure 6.1.1. 1 User Registration Screen 53](#_Toc168493588)

[Figure 6.1.2. 1 User Login Form 55](#_Toc168493602)

[Figure 6.1.3. 1 Favorites Screen 57](#_Toc168493608)

[Figure 6.1.4. 1 Profile Management 59](#_Toc168493615)

[Figure 6.1.5. 1 Home Screen 61](#_Toc168493620)

[Figure 6.1.6. 1 Subscription Plan 63](#_Toc168493626)

[Figure 6.1.7. 1 Reminders Screen 65](#_Toc168493630)

**List of tables**

[Table 1. 1 Android Versions 13](#_Toc166694544)

# **Chapter 01: INTRODUCTION**

## Introduction of Android operating system

Being a mobile operating system, Android OS is a modified version of Linux, originally developed by a start-up, Android, Inc. As Google entered mobile market, it purchased Android and, in a bid, to encourage independent development works, it released the developer tools under the open-source Apache License. The permissive licensing allows the OS and related software to be modified and distributed by enthusiastic developers, network operators and device manufacturers.

### **1.1.1 Technology Features**

**Dalvik VM**

A modified version of JAVA programming language is used for app development with Dalvik VM used to run the apps on Android devices. Dalvik VM can be viewed as modified version of JVM constrained in terms of memory and processor speed and converts the java bytecode (in form of JVM compatible class files) to Dalvik compatible dex executables before installation.

**Application Interface & II/W Support**

Based on Direct Manipulation, the on screen objects have been programmed to respond to real world actions like swiping, touching etc. Boasting of a fast & responsive fluidic touch screen, the OS supports various dedicated hardware like proximity sensors, gyroscopes, magnetometer and accelerometer etc. The Home Screen is analogue to the Desktop in a Windows OS. Powered by Google Play Store, millions of apps can be readily downloaded and used. Apps are available in the apk format. Google provides the SDK free of cost and it supports a comprehensive set of developing tools which primarily includes an IDE (Android Studio), a debugger, and support for emulator and sample codes etc. It even supports C/C++ extensions or bytecodes through JNI and the support is available through Native Development Kit (NDK).

**Architecture**

Based on Linux kernel, most of the middleware, APIs & libraries are written in C. The hardware platform is generally of ARM architecture (hence parallel processing) with later support being available for x86 & MIPS also. All GNU libraries are not supported, hence restricting porting of Windows applications onto Android Device owners are not given ROOT access and hence they have access only to /data partition on flash storage and not to the /system which holds OS and boot files and other sensitive read-only partitions.

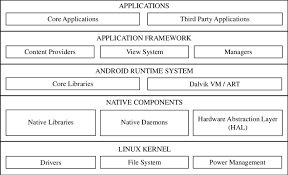


Figure 1. 1 Android Operating System Architecture

**Memory Management**

The OS supports multi-threading but depending on the instant memory availability, it can kill application to reduce overloading. The RAM management is such so that power consumption is at minimum. As far as third-party applications are considered, the SDK provides with ample library entities such as Services, Background Tasks & Foreground Tasks for working with application lifetime.

**Security & Privacy**

Though the OS is immune to normal user usage, but the security flaws can be exploited, as done by the open-source community, to get ROOT access (can be used for malicious purposes by crackers) and modify device capabilities. Except that, the device owners' applications are mostly run in an isolated area of OS called sandbox which restricts access to the system resources and hardware unless the user explicitly gives the access permissions during installation itself. Hence, the app gains access to /data partition through this method and the data partition only. The newest Android OS versions have enhanced security features such as malware scanners built into system to keep a tab on malicious software downloaded through Google Play or any other third- party application. Newer applications now rely on OAUTH 2.0 for secure access to internet.

**Network Connectivity**

The OS supports a full range of connectivity solutions ranging from Bluetooth to ZigBee (through accessory support) and from 2G to LTE support. It supports data packet transmissions through GPRS/EDGE support. Internet can also be accessed through Wi-Fi, WiMAX and shared among other devices through tethering (both over Wi-Fi & USB) support. PC communication is established through device management software using USB & Bluetooth. HTTP service is supported and through use of Google APIs, the phone is an effective GPS enabled device.

### **1.1.2. Android Versions till Date**

Android is the operating system that powers more than one billion smartphones and tablets. Since these devices make our lives so sweet, each Android version is named after a dessert. Whether it's getting directions or even slicing virtual fruit, each Android release makes something new possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Android | Code Name | API Level | Version |
| Android 1.0 | N/A | 1 | 1.0 |
| Android 1.1 | Petit Four | 2 | 1.1 |
| Android 1.5 | Cupcake | 3 | 1.5 |
| Android 1.6 | Donut | 4 | 1.6 |
| Android 2.0 | Eclair | 5 | 2.0 |
| Android 2.2 | Froyo | 8 | 2.2 |
| Android 2.3 | Gingerbread | 9 | 2.3 |
| Android 3.0 | Honeycomb | 11 | 3.0 |
| Android 4.0 | Ice Cream Sandwich | 14 | 4.0 |
| Android 4.1 | Jelly Bean | 16 | 4.1 |
| Android 4.4 | KitKat | 19 | 4.4 |
| Android 5.0 | Lollipop | 21 | 5.0 |
| Android 6.0 | Marshmallow | 23 | 6.0 |
| Android 7.0 | Nougat | 24 | 7.0 |
| Android 8.0 | Oreo | 26 | 8.0 |
| Android 9.0 | Pie | 28 | 9.0 |
| Android 10.0 | Android Q | 29 | 10.0 |

Table 1. 1 Android Versions

## 1.2 Project introduction

### **1.2.1 Main Theme**

The main theme of Quotez revolves around the fundamental human need for inspiration and motivation. In today's digitally-driven society, individuals often seek moments of reflection and encouragement amidst their busy lives. Quotez aims to cater to this need by providing a digital platform where users can easily access a curated collection of inspirational quotes.

The essence of Quotez lies in its ability to empower users with words that uplift, inspire, and resonate with their personal journeys. Whether seeking motivation for professional endeavors, personal growth, or simply a daily dose of positivity, Quotez endeavors to be the go-to destination for users looking to enrich their lives through meaningful quotes.

### **1.2.2 Scope of the Project**

The scope of the Quotez project encompasses the development of a comprehensive mobile application coupled with a robust backend infrastructure to support its functionality. The app will be designed to deliver a seamless user experience across various devices and platforms, ensuring accessibility for a diverse user base.

Key features of the Quotez app include:

* User Registration and Authentication: Users will have the ability to create accounts, log in securely, and manage their profiles within the app.
* Curated Database of Inspirational Quotes: The app will boast a vast repository of inspirational quotes sourced from renowned authors, thinkers, and leaders across different cultures and time periods.
* Search and Filter Functionalities: Users can easily explore the quote database using intuitive search and filter options based on keywords, authors, themes, and more.
* Personalized Collections: Quotez enables users to save their favorite quotes and organize them into personalized collections, facilitating easy access and management.
* Social Sharing Capabilities: Users can seamlessly share their favorite quotes with friends and followers on social media platforms, fostering a community of inspiration and connection.

### **1.2.3 Objectives of the Project**

The primary objectives of the Quotez project are outlined as follows:

1. User-Centric Design: Develop an intuitive and visually appealing user interface that prioritizes ease of navigation and engagement.

2. Comprehensive Quote Database: Curate a diverse collection of high-quality quotes spanning various themes, genres, and cultural backgrounds to cater to the diverse interests of users.

3. Enhanced User Experience: Implement robust search, filter, and personalization features to empower users with personalized content tailored to their preferences.

4. Social Integration: Integrate seamless social sharing functionalities to enable users to connect with others and spread inspiration across digital platforms.

5. Scalability and Reliability: Build a scalable backend infrastructure capable of accommodating future growth in user base and quote database size, while ensuring reliability and security.

By fulfilling these objectives, Quotez aims to establish itself as a premier destination for individuals seeking daily inspiration, motivation, and empowerment.

### **1.2.4 Thesis Organization**

The following thesis has been organized to give a clear view of what and how the app behaves. Chapter 1 gives a clear introduction to why android was chosen as the target platform. It also talks about why the apps were built and to what ends the apps are required plus the scope of the applications. Chapter 2 gives a literature view of the work that has already been done in the field of Quoteza. Chapter 3 provides an analysis of various use case scenarios of the project. It shows relevant use-case diagrams, sequence diagrams and class diagrams. Chapter 4 speaks in detail about Quoteza application. It gives a clear view of the database model used for the project through various tools such as ER-Diagrams, Data Flow Diagrams, Table Design etc. Chapter 4 goes into details of the technologies used, software used & backend configurations. It also gives view on how to implement and connect to those technologies from Android Application and then back to the device end. Chapter 5 shows screenshots of various screens of the application tested on a live tablet.

### **1.2.5 Summary**

This chapter dealt with questions like why the application was created and what does it stand for. Overview or general working principles have been provided. The problem statement for has been detailed and analyzed well. The nature of the project has been explained. An introduction into why android was selected as target OS has also been provided.

# **Chapter 02: System Analysis**

## 2.1 Feasibility Study

The feasibility study for the Quotez app involves an in-depth analysis of various factors to determine the practicality and potential success of the project. This includes technical, economic, market, operational, and legal feasibility. Each of these aspects is explored to ensure a comprehensive understanding of the feasibility of developing and launching the Quotez app.

### **2.1.1 Technical Feasibility**

Objective: To evaluate the technical requirements and challenges associated with the development of the Quotez app.

* **Technology Stack:** The Quotez app will be developed using Flutter, a powerful and flexible framework for cross-platform mobile app development. Flutter allows for a single codebase to be deployed on both iOS and Android platforms, significantly reducing development time and costs. The backend will be powered by Firebase, which offers a suite of tools and services such as real-time database, Firestore, authentication, cloud storage, and cloud functions. This combination ensures a robust and scalable architecture.
* **Development Tools:** The development will utilize modern tools and environments:

1. Flutter SDK: For developing the cross-platform app.
2. Firebase Console: For managing backend services.
3. IDEs: Such as Android Studio or Visual Studio Code.
4. Version Control: Using GitHub for collaborative development and version control.

* **Personal Expertise:**

I am familiar with Dart programming language (used in Flutter) and Firebase services is advantageous. Solo development allows you to leverage your expertise and focus on building the app according to your vision.

* **Integration Capabilities:**

Integration with Firebase services and third-party APIs can be efficiently managed by a single developer. Firebase provides easy-to-use SDKs and documentation, while third-party APIs often offer straightforward integration methods.

### **2.1.2 Economic Feasibility**

**Initial Development Costs:**

Being a solo developer, initial development costs primarily include personal expenses such as time investment and software subscriptions. Open-source tools and free-tier plans for services like Firebase help minimize upfront costs.

**Ongoing Maintenance Costs:**

Solo development reduces ongoing maintenance costs as there are no salaries to be paid. Cloud service expenses for Firebase may increase as user base and usage grow, but costs can be managed effectively.

**Revenue Projections:**

As a solo developer, revenue streams such as a freemium model, in-app advertisements, and premium features can generate income. Revenue projections need to be conservative and aligned with the app's development timeline and target audience.

### **2.1.3 Operational Feasibility**

**Development Plan:**

I am creating a realistic development plan with achievable milestones is essential. Breaking down tasks into manageable chunks and prioritizing features based on user feedback can help ensure operational feasibility.

**Resource Allocation:**

I need to manage my time and resources effectively. Setting realistic expectations and balancing development tasks with other commitments is crucial for successful project execution.

**Risk Management:**

Identifying potential risks such as technical challenges or scope creep and proactively addressing them can mitigate project risks. Flexibility and adaptability are key traits for me to navigate unforeseen challenges.

**Scalability:**

I should design the app with scalability in mind, allowing for future growth and expansion. Choosing scalable technologies like Flutter and Firebase and implementing efficient coding practices can facilitate scalability.

### **2.1.4 Legal Feasibility**

**Intellectual Property:**

Ensuring proper attribution of quotes and compliance with copyright laws is essential for legal feasibility. Solo developers can use public domain quotes or obtain necessary permissions for copyrighted content.

**Data Privacy and Security:**

Implementing robust data privacy and security measures to protect user data is paramount. Solo developers should adhere to data protection regulations and prioritize user privacy.

**Terms of Service and Privacy Policy:**

Creating clear and comprehensive terms of service and privacy policy documents that outline user rights and data usage policies is important for legal compliance. Solo developers can use online templates or seek legal advice to draft these documents.

## 2.2 Existing System: Data Gathering

Data gathering for the existing system involves collecting information about similar apps, user preferences, and market trends to understand the landscape in which Quoteza will operate. As a solo developer, conducting thorough research is crucial to inform decision-making and ensure the app meets user needs effectively.

### **2.2.1 Questionnaires**

**Design of Questionnaires:**

Create structured questionnaires tailored to gather insights into user preferences, experiences, and expectations regarding existing inspirational quote apps. Questions should cover aspects such as app usage frequency, favorite features, areas for improvement, and desired functionalities in a new app like Quoteza.

**Distribution:**

Utilize online platforms, social media, and relevant communities to distribute the questionnaires to a diverse sample of potential users. Engaging with target audiences through targeted messaging and incentives can encourage participation and gather valuable feedback.

**Questions Focus:**

Focus questions on understanding user behavior, motivations, and pain points when using existing quote apps. Inquire about specific features users find valuable, challenges they encounter, and their willingness to adopt new features or platforms.

**Analysis of Responses:**

Collect and analyze responses systematically to identify common themes, patterns, and trends. Pay attention to recurring feedback, user preferences, and areas of consensus or divergence among respondents. This analysis will guide decision-making during the development of Quoteza.

### **2.2.2 Sampling & Observations**

**Sampling Method:**

Utilize stratified sampling to ensure representation from different demographics, age groups, and user segments within the target audience. This approach provides a comprehensive understanding of user needs and preferences across diverse user groups.

**Observation Techniques:**

Conduct direct observations of users interacting with existing inspirational quote apps in real-world settings. Observe user behavior, navigation patterns, and interactions with app features to identify usability issues, pain points, and areas for improvement.

**Data Collection:**

Systematically record observations, noting user actions, reactions, and feedback. Capture insights into user experience, interface design, and feature usability to inform the development of Quoteza.

**Insights Gained:**

Summarize key insights gained from questionnaires and observations, highlighting common trends, user preferences, and areas for innovation. These insights will serve as a foundation for designing and developing Quoteza to meet user needs effectively.

By conducting comprehensive data gathering through questionnaires and observations, valuable insights into user preferences, behaviors, and market trends will be obtained. This data will inform the development of Quoteza, ensuring it addresses user needs and stands out in the competitive landscape of inspirational quote apps.

## 2.3 Existing System: Data Analysis

Data analysis plays a crucial role in understanding the existing landscape of inspirational quote apps and extracting valuable insights to inform the development of Quoteza. As a solo developer, thorough analysis of collected data will guide decision-making and ensure Quoteza meets user needs effectively.

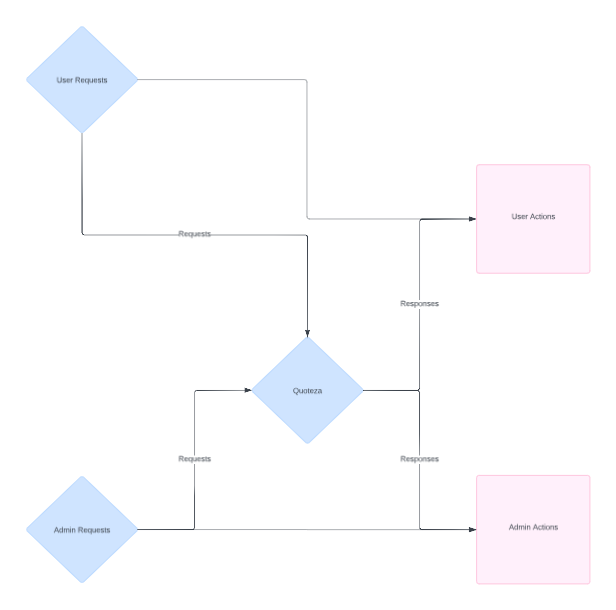
### **2.3.1 Data Flow Diagrams (DFDs)**

**Purpose of DFDs:**

Develop Data Flow Diagrams (DFDs) to visually represent the flow of data within existing inspirational quote apps. DFDs provide a structured overview of how data moves through the system, helping identify key processes, inputs, outputs, and interactions.

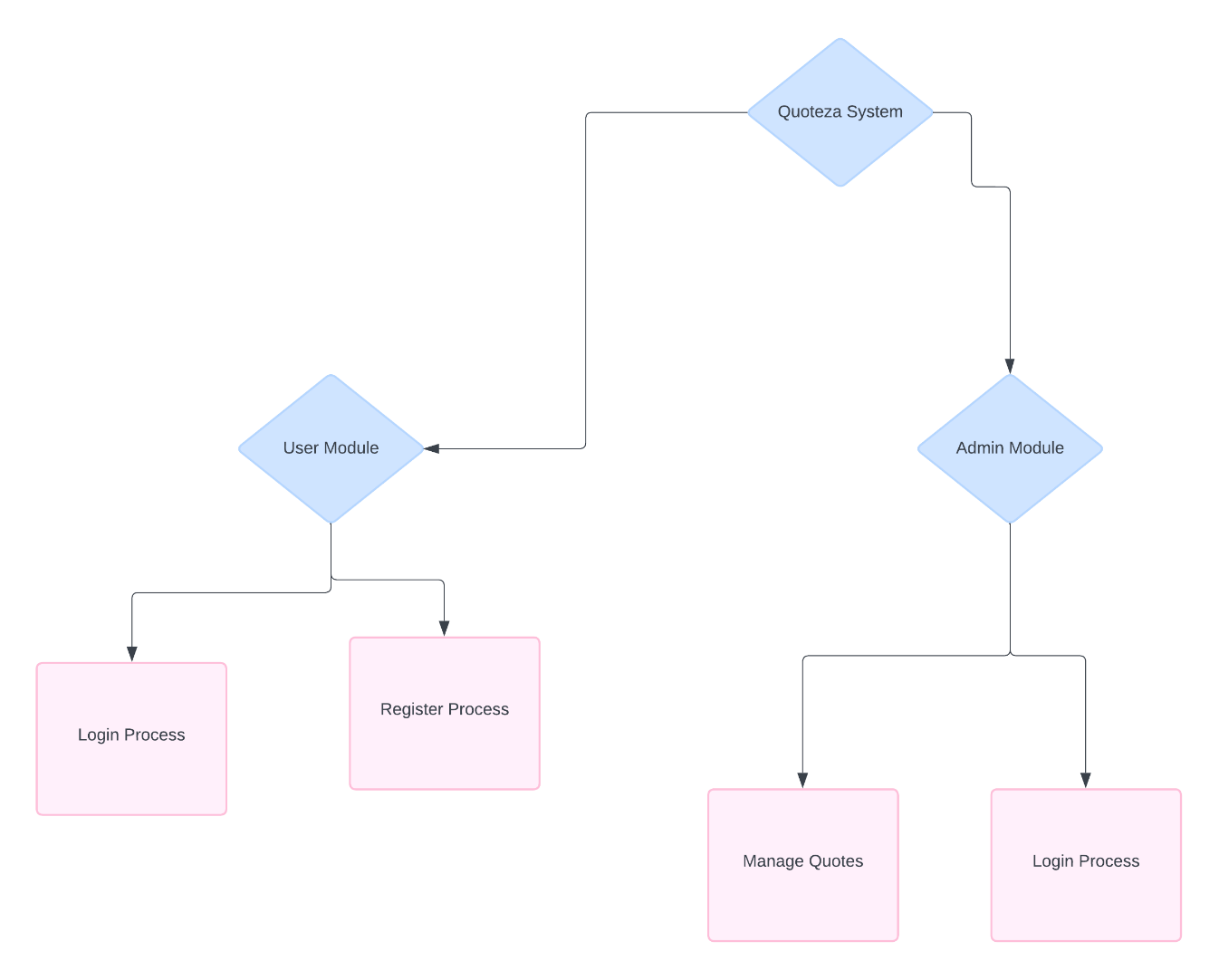
**DFD Levels:**

Level 0 (Context Diagram): Provide an overarching view of the system, illustrating high-level processes and interactions with external entities such as users and databases.



**Figure 2.3.1. 1 DFD Level Zero**

Level 1 (Decomposition Diagram): Break down the main processes identified in the context diagram into more detailed subprocesses, inputs, outputs, and data flows.



**Figure 2.3.1. 2 DFD Level 1 diagram**

**Interpretation:**

Analyze the generated DFDs to identify inefficiencies, bottlenecks, and areas where existing quote apps excel or fall short. Pay attention to the flow of data between different components of the system, identifying opportunities for optimization and enhancement in Quoteza.

### **2.3.2 Requirements Engineering**

**Requirement Elicitation:**

Gather functional and non-functional requirements based on insights from data gathering activities, including questionnaires and observations. Engage with potential users to understand their needs, preferences, and pain points when using existing quote apps.

**Requirement Analysis:**

Analyze collected requirements to ensure they are clear, complete, and feasible for implementation. Prioritize requirements based on user feedback, market trends, and project constraints, focusing on features that add the most value to Quoteza.

**Requirement Specification:**

Document requirements in a structured format, detailing functional specifications (e.g., user authentication, quote browsing, social sharing) and non-functional specifications (e.g., performance, security, usability). Clearly define the scope and objectives of Quoteza to guide development efforts.

**Requirement Validation:**

Validate requirements with stakeholders, including potential users and project sponsors, to ensure alignment with user expectations and project goals. Gather feedback and iterate on requirements as needed to refine and finalize the scope of Quoteza.

### **2.3.3 Deliverables**

**Functional Specifications:**

Document detailed specifications for Quoteza's functionality, outlining user stories, use cases, and system behavior under different scenarios. Specify features such as user registration, quote browsing, saving favorites, and social sharing with clear acceptance criteria.

**Non-Functional Specifications:**

Define requirements related to performance, security, usability, and scalability to ensure Quoteza meets user expectations and industry standards. Specify metrics and benchmarks for measuring and evaluating non-functional aspects of the app.

**System Architecture:**

Design a high-level system architecture for Quoteza, outlining the integration of Flutter for the frontend, Firebase for backend services, and third-party APIs for additional functionality. Define components, interfaces, and data flows to guide implementation.

**Prototypes and Wireframes:**

Develop prototypes and wireframes to visualize the user interface and user experience design of Quoteza. Iterate on design concepts based on user feedback and usability testing to create intuitive and engaging interfaces.

**Implementation Plan:**

Outline a comprehensive plan for implementing Quoteza, including development tasks, timelines, resource allocation, and milestones. Break down the development process into manageable increments, prioritizing features based on user needs and project objectives.

By conducting thorough data analysis and documenting requirements, specifications, and deliverables, Quoteza's development will be guided by a clear understanding of user needs, market trends, and system requirements. This structured approach ensures Quoteza meets user expectations and stands out in the competitive landscape of inspirational quote apps.

# **Chapter 03: System Design**

## 3.1 Introduction to System Design

The system design phase is a critical stage in the development of the Quoteza app, where the conceptual framework and architectural blueprint of the system are established. This section provides an overview of the system design process, outlining the key objectives, methodologies, and considerations involved in designing Quoteza.

## 3.2 System Design using UML

Unified Modeling Language (UML) is a standardized modeling language widely used in software engineering for visualizing, specifying, constructing, and documenting software systems. In the design of Quoteza, UML diagrams serve as powerful tools for modeling the system's structure and behavior. This section discusses the utilization of UML in system design, focusing on two essential diagram types: Use Case Diagrams and Sequence Diagrams.

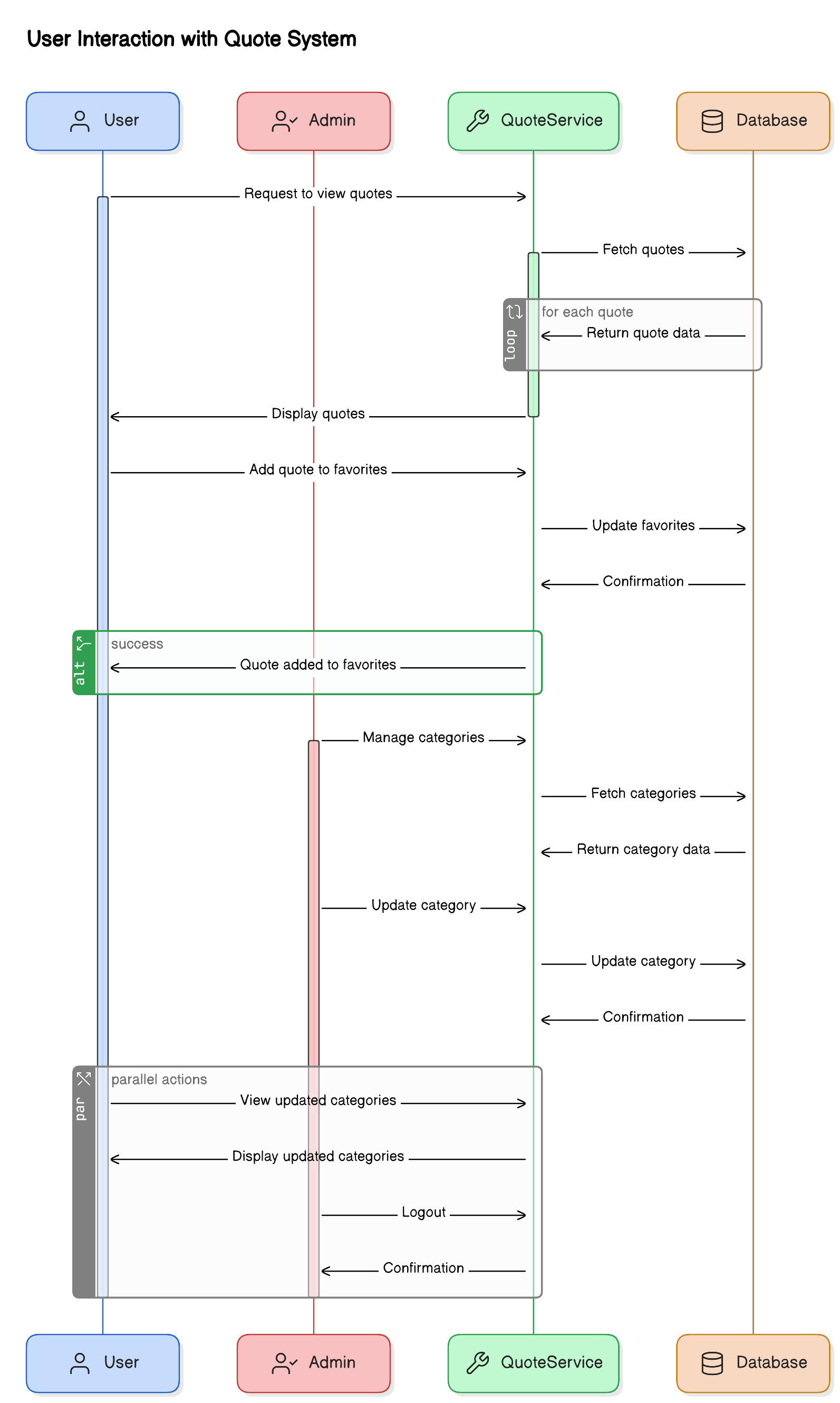
### **3.2.1 Use Case Diagrams**

Use Case Diagrams provide a graphical representation of the interactions between users (actors) and the system (Quoteza) to accomplish specific tasks or goals. These diagrams help identify and define the functional requirements of the system from a user's perspective. In the context of Quoteza, Use Case Diagrams depict various user roles, their interactions with the app, and the primary functionalities they perform. This enables a clear understanding of the system's behavioral aspects and aids in defining the scope and boundaries of Quoteza.

**Diagram**

### **3.2.2 Sequence Diagrams**

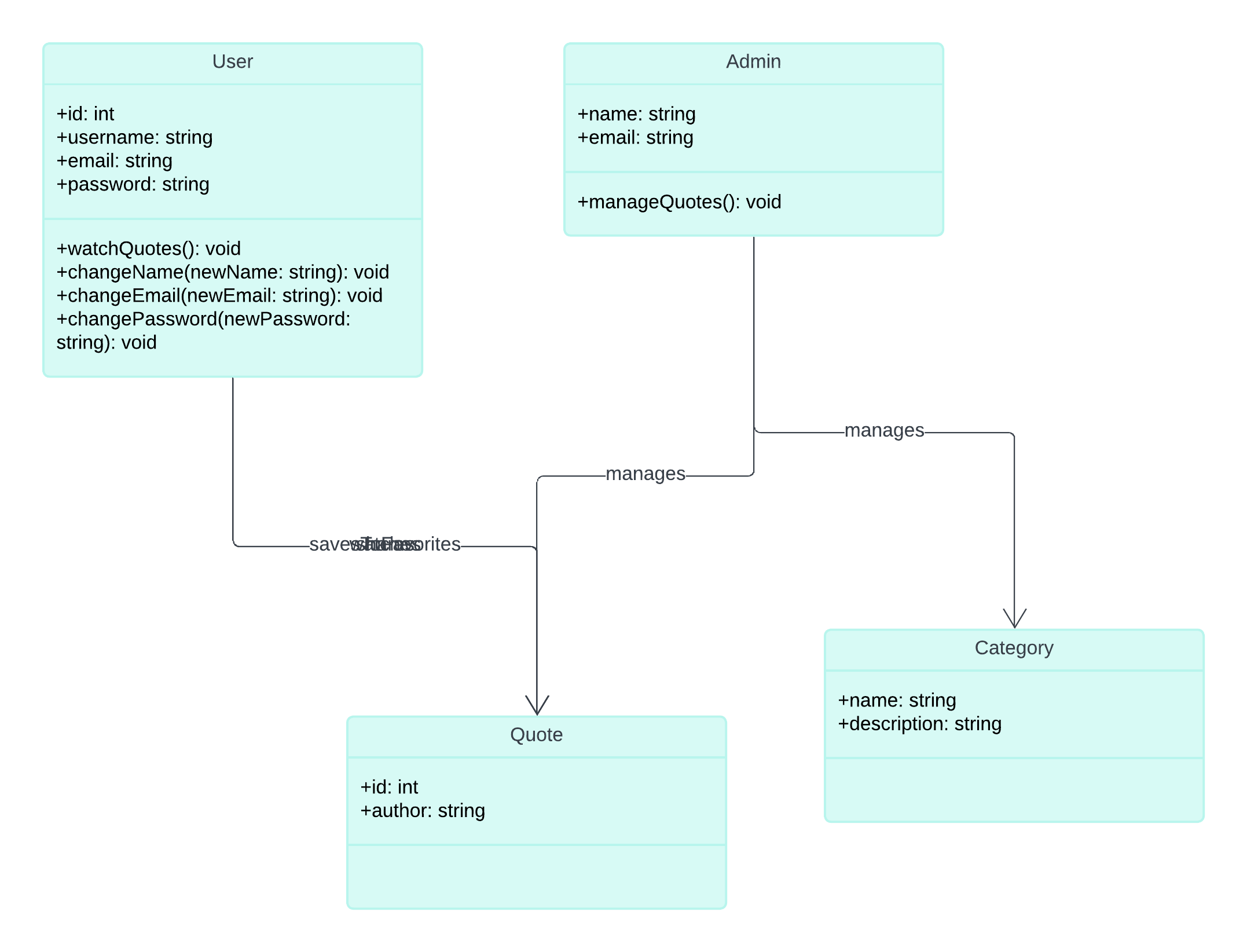
Sequence Diagrams illustrate the sequence of interactions between objects (actors, components) in a system over time. They depict the flow of messages exchanged between objects in response to specific actions or events. In the design of Quoteza, Sequence Diagrams capture the dynamic behavior of the system, showcasing how users interact with the app's components and how these interactions trigger system responses. By visualizing the sequence of events, dependencies, and interactions within Quoteza, Sequence Diagrams provide valuable insights into the system's functionality, communication patterns, and control flow.



**Figure 3.2.2. 1 Sequence Diagram of Quoteza**

### **3.2.3 Class Diagram**

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Following is the class diagram of this project:

s

**Figure 3.2.3. 1 Class Diagram of Quoteza**

## 3.3 Database Design

Database design is a crucial component of the system design phase, focusing on the structured organization of data to ensure efficient storage, retrieval, and management. For the Quoteza app, an effective database design is essential to handle quotes, user data, and app settings seamlessly. This section outlines the database design process, including the creation of Entity Relationship Diagrams (ERDs), building a relational model from ERDs, and normalizing the relational model to the third normal form (3NF).

**Advantages of Databases**

The most important advantages of databases are as follows:

* Reduced data redundancy
* Reduced updating errors and increased consistency
* Greater data integrity and independence from applications programs
* Improved data access to users through use of host and query languages
* Improved data security
* Reduced data entry, storage, and retrieval costs
* Facilitated development of new applications program

**Disadvantages of Databases**

* Database systems are complex, difficult, and time-consuming to design
* Substantial hardware and software start-up costs
* Damage to database affects virtually all applications programs
* Extensive conversion costs in moving form a file-based system to a database system
* Initial training required for all programmers and users

**Database Design**

Database design is one of the major phases in software development. Database acts as the backbone of any software. A well-designed database can play a vital role in the success of a system so this phase must be carried out with great responsibility.

**Qualities of Good Database Design**

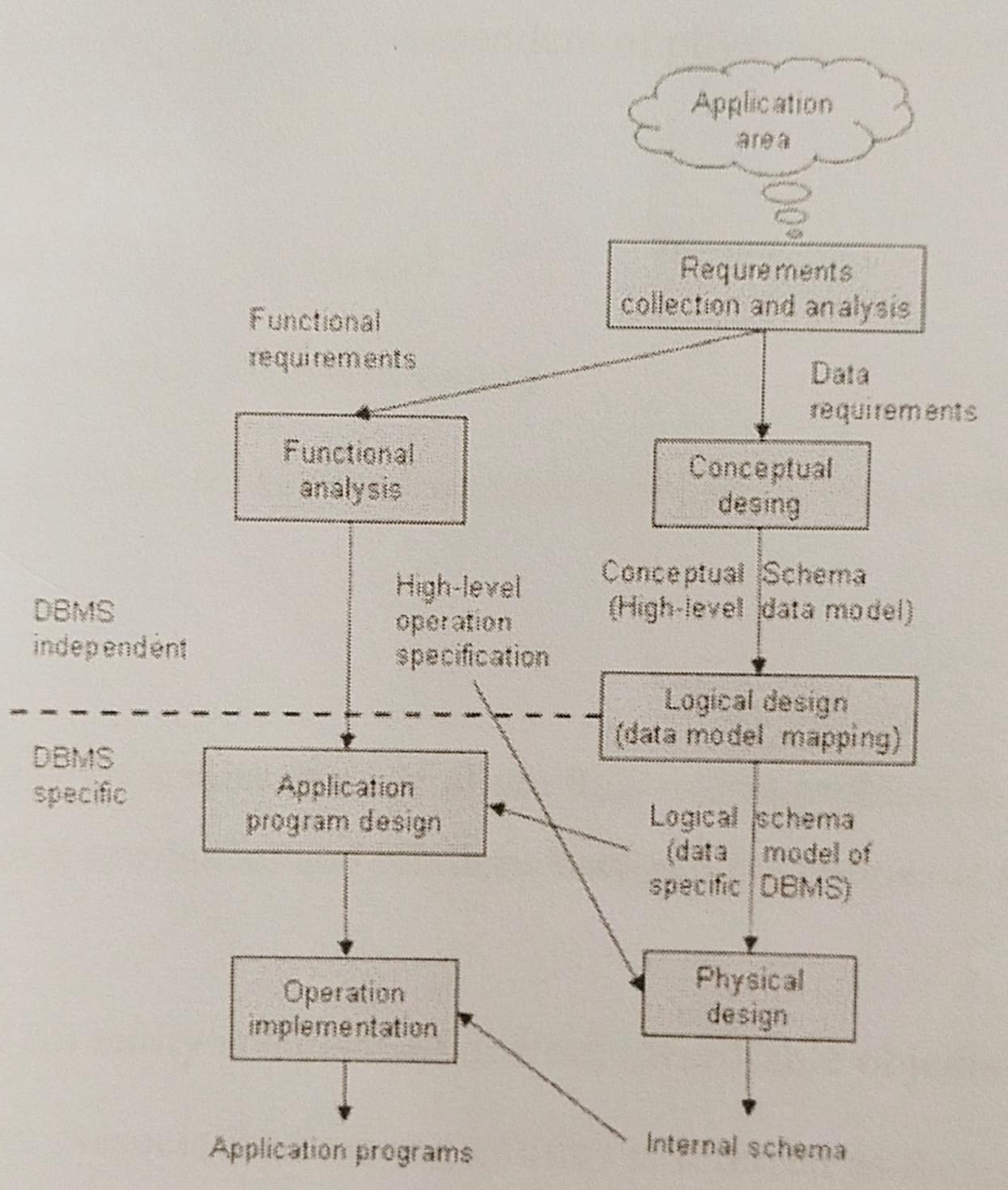
* . Reflects real-world structure of the problem.
* Can represent all expected data over time.
* Avoids redundancy and ensures Consistency.
* Provides efficient access to data.
* Supports the maintenance of data integrity over time.

**Steps in Database Design**

The process of database design is divided into different parts.

It consists of a series of steps which are as follows:

1. Requirement Analysis
2. Conceptual Database Design (ER-Diagram)
3. Logical Database Design (Tables, Normalization etc.)
4. Physical Database design (Table Indexing, Clustering etc.)



**Figure 3.3. 1 Steps For Database Design**

### **3.3.1 Requirement Analysis**

In this phase a detailed analysis of the requirement is done. The objective of this phase is to get a clear understanding of the requirements. Various information gathering methods can be used for this purpose. Some of them are

* Interview
* Analyzing documents
* Survey
* Site visit
* Joint Applications Design (JAD) and Joint Requirements Analysis(JRA)
* Prototyping

### **3.3.2 Conceptual Database Design (Data Modeling)**

The requirements analyzed in the previous step is converted into conceptual model. It is a process of constructing a data model for each view of the real-world problem which is independent of physical considerations.

This step involves:

* Constructing the ER Model
* Check the model for redundancy
* Validating the model against user transactions to ensure all the scenarios are supported.

**ER Modelling:**

Pictorial Representation of the Real-world problem in terms of entities (which have attributes) and relations between the entities is referred as ER diagram.

* **Entities**: An entity is a class of distinct identifiable objects or concepts
* **Relations**: Associations among entities is referred as Relations
* **Attributes**: Attributes are properties or characteristics of entities.

### **3.3.3 Logical Database Design**

Once the relationships and dependencies are identified the data can be arranged into logical structures and is mapped into database management system tables.

This step involves:

* Table generation from ER-Model.
* Normalization of tables.

The Cardinality of relationships among the entities can be considered while deriving the tables from ER Model into:

**One-to-one:**

Entities with "one-to-one" relationships should be merged into a single entity. Each remaining entity is modeled by a table with a primary key and attributes, some of which may be foreign keys.

**One-to-many:**

"One-to-many" relationships are modeled by a foreign key attribute in the table. This foreign key would refer to another table that would contain the "many" sides of the relation.

**Many-to-many:**

"Many-to-many" relationships among two entities are modeled by a third table that has foreign keys that refer to the entities. Attribute of entities corresponds to fields in relational tables.

**Normalization of Tables:**

Normalization is a process of eliminating redundancy and other anomalies in the system. Redundancy means that the same data is saved more than once in a database. Update anomaly is a consequence of redundancy. If a piece of data is saved in more than one place, the same data must be updated in more than one place. Normalization is a technique by which one can modify the relation schema to reduce the redundancy. Each normalization phase adds more relations (tables) into the database. In most cases in the enterprise world, normalization up to Third Normal form would suffice.

### **3.3.4 Physical Database Design**

It deals with the physical implementation of the database in a database management system. It includes the specification of data elements, data types, indexing etc. All this information is stored in the data dictionary.

## 3.4 Entity Relationship Diagrams (ERDs)

**Purpose of ERDs:**

Entity Relationship Diagrams (ERDs) are used to visually represent the data entities, their attributes, and the relationships between them. ERDs serve as a blueprint for designing the database schema by providing a clear and structured view of the data model.

**Components of ERDs:**

**Entities**: Represent objects or concepts within the system, such as Users, Quotes, and Categories.

**Attributes**: Define the properties or characteristics of entities, such as userName, email, quoteText, and categoryName.

**Relationships**: Illustrate the connections between entities, such as the relationship between Users and Quotes (a user can save multiple quotes) and Quotes and Categories (a quote belongs to a category).

**Creating ERDs for Quoteza:**

Identify the key entities involved in the app, such as User, Quote, Category, and Favorite.

Define the attributes for each entity.

Establish relationships between entities to reflect how they interact within the app.

ERD for Quoteza:

**Entities**:

User: userID (Primary Key), userName, email, password

Quote: quoteID (Primary Key), quoteText, author, categoryID (Foreign Key)

Category: categoryID (Primary Key), categoryName

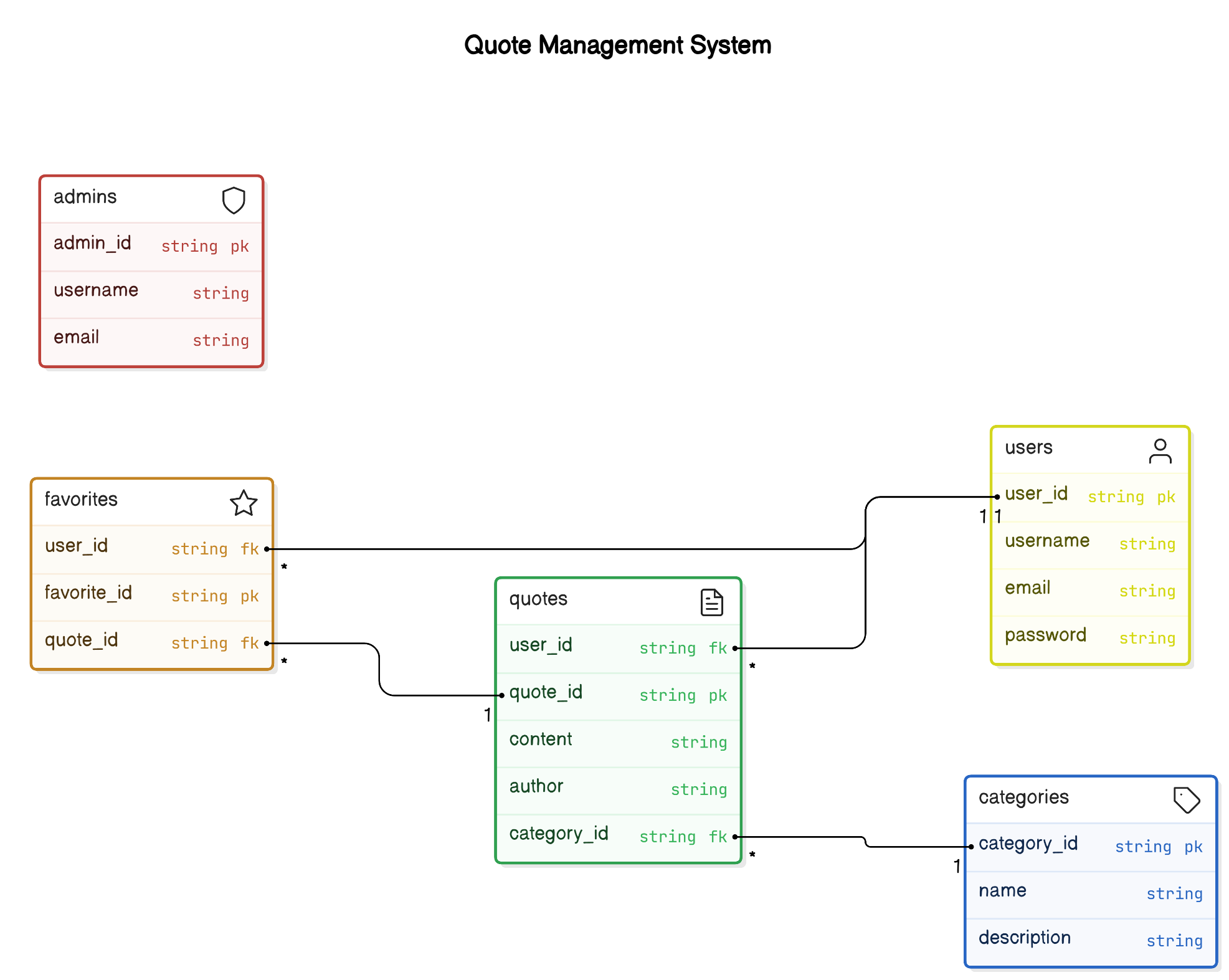
Favorite: favoriteID (Primary Key), userID (Foreign Key), quoteID (Foreign Key)

**Relationships**:

User to Favorite: One-to-Many (one user can have multiple favorites)

Quote to Favorite: One-to-Many (one quote can be favorited by multiple users)

Category to Quote: One-to-Many (one category can contain multiple quotes)



**Figure3.4. 1 ER Diagram of Quoteza App**

## 3.5 Building a Relational Model from ERDs

Transforming ERDs to Relational Models:

Once the ERD is created, the next step is to translate it into a relational model. This involves defining tables, columns, and relationships based on the ERD.

### **Steps to Build a Relational Model:**

1. Create Tables: Each entity in the ERD becomes a table in the relational model.
2. Define Columns: Attributes of each entity become columns in the corresponding table.
3. Set Primary Keys: Identify primary keys for each table to uniquely identify records.
4. Establish Foreign Keys: Use foreign keys to represent relationships between tables.

### **Relational Model for Quoteza:**

**User Table:**

Columns: userID (Primary Key), userName, email, password

**Quote Table:**

Columns: quoteID (Primary Key), quoteText, author, categoryID (Foreign Key)

**Category Table:**

Columns: categoryID (Primary Key), categoryName

**Favorite Table:**

Columns: favoriteID (Primary Key), userID (Foreign Key), quoteID (Foreign Key)

## 3.6 Normalizing Relational Model to 3NF

**Purpose of Normalization:**

Normalization is the process of organizing data to minimize redundancy and improve data integrity. Normalizing the relational model to the third normal form (3NF) ensures that the database is free of update anomalies and maintains data consistency.

### **Normalization Steps:**

**First Normal Form (1NF):** Ensure that all columns contain atomic (indivisible) values and each column contains values of a single type.

**Second Normal Form (2NF):** Ensure that the table is in 1NF and all non-key attributes are fully functionally dependent on the primary key.

**Third Normal Form (3NF):** Ensure that the table is in 2NF and all attributes are only dependent on the primary key, removing any transitive dependencies.

Normalization for Quoteza:

* User Table: Already in 3NF (all attributes depend only on userID)
* Quote Table: Ensure quoteText and author depend only on quoteID
* Category Table: Already in 3NF (categoryName depends only on categoryID)
* Favorite Table: Ensure that userID and quoteID depend only on favoriteID

# **Chapter 04: System Development**

## 4.1 Introduction to System Development

System development is the phase where the theoretical design is transformed into a functional and operational system. This phase involves coding, testing, integrating, and deploying the system. For Quoteza, system development will leverage modern development practices and tools to ensure a robust and user-friendly application. The development process aims to meet all specified requirements and deliver a high-quality product that enhances the user experience of interacting with inspirational quotes.

## 4.2 Tool/Language Selection

### **4.2.1 Programming Languages and Frameworks**:

* **Flutter**: Flutter is chosen as the primary framework for developing Quoteza due to its capability to create natively compiled applications for mobile, web, and desktop from a single codebase. Flutter uses Dart, a modern language with a strong focus on performance and productivity.
* **Dart**: Dart is the programming language used by Flutter. It is optimized for building mobile, desktop, server, and web applications.

### **4.2.2 Backend Services**:

* **Firebase**: Firebase provides a comprehensive suite of backend services, including authentication, real-time database, Firestore, cloud storage, and analytics. Its seamless integration with Flutter makes it an ideal choice for Quoteza.

### **4.2.3 APIs**:

* Third-party APIs will be used to fetch quotes and authors. APIs such as QuoteGarden, TheySaidSo, or custom APIs can provide a rich database of quotes.

### **4.2.4 Development Tools**:

* **Android Studio**: A powerful integrated development environment (IDE) that supports Flutter and Dart, providing features like code completion, debugging, and a robust set of tools for building and testing apps.
* **Visual Studio Code**: An alternative lightweight IDE that supports Flutter development through extensions and plugins.

### **4.2.5 Version Control**:

* **Git**: Git will be used for version control, enabling collaborative development, version tracking, and efficient management of code changes.
* **GitHub**: A platform for hosting the Git repositories, facilitating collaboration, issue tracking, and code review.

## 4.3 Hardware for the System

### **4.3.1 Development Hardware**:

* **Development Machine**: A modern laptop or desktop with 8GB of RAM, a multi-core processor, and a solid-state drive (SSD) for efficient development. Examples include a MacBook Pro, Dell XPS, or any equivalent system.
* **Mobile Devices**: Access to various mobile devices (both Android and iOS) for testing and debugging is crucial to ensure the app performs well across different platforms and screen sizes.

### **4.3.2 Server Hardware**:

* **Cloud Services**: Since Firebase is a cloud-based backend service, no physical server hardware is required. Firebase's cloud infrastructure will handle database management, storage, and server-side logic, ensuring scalability and reliability.

## 4.4 Software Development & Implementation

### **4.4.1 Development Methodology**:

* **Agile Development**: An agile methodology will be adopted, emphasizing iterative development, continuous feedback, and flexibility. This approach allows for incremental improvements and rapid response to changing requirements.

### **4.4.2 Development Phases**:

1. **Planning and Requirement Analysis**: Review the requirements gathered during the system analysis phase and prioritize features based on user needs and project goals.
2. **Design**: Create detailed designs for the user interface, system architecture, and database schema. Develop prototypes and wireframes to visualize the user experience.
3. **Implementation**: Begin coding based on the designs and specifications. This phase involves developing the frontend using Flutter and integrating Firebase for backend services.
4. **Testing**: Conduct rigorous testing to identify and fix bugs, ensure functionality, and verify that the app meets all requirements. Types of testing include unit testing, integration testing, system testing, and user acceptance testing.
5. **Deployment**: Prepare the app for deployment to app stores (Google Play Store and Apple App Store). This involves packaging the app, complying with store guidelines, and setting up Firebase services.
6. **Maintenance**: After deployment, ongoing maintenance is essential to address user feedback, fix bugs, and update the app with new features or improvements.

### **4.4.3 Implementation Details**:

* **Frontend Development**: Develop the user interface with Flutter, focusing on creating a responsive and visually appealing design. Implement core features such as quote browsing, user authentication, and social sharing.
* **Backend Integration**: Use Firebase for real-time database management, user authentication, and cloud storage. Implement necessary cloud functions and database rules to ensure data security and integrity.
* **API Integration**: Integrate third-party APIs to fetch and display quotes dynamically. Handle API responses and ensure the data is presented correctly to users.

### **4.4.4 Client-Side Technology**

The client side of Quoteza involves technologies that create the user interface and handle interactions on the user’s device. Although Flutter is the primary framework for this project, we will also provide an overview of other relevant client-side technologies such as HTML, PHP, and DHTML.

#### **4.4.1.1 HTML, PHP, and DHTML**

**HTML (Hypertext Markup Language)**:

* **Purpose**: HTML is the standard language for creating and structuring web pages. It defines the content and structure of web pages using elements and tags.
* **Usage in Quoteza**: While HTML is primarily used for web applications, it provides the foundation for web views in mobile apps. If Quoteza includes any web-based content or web views, HTML will be utilized to structure that content.

**PHP (Hypertext Preprocessor)**:

* **Purpose**: PHP is a server-side scripting language designed for web development. It is embedded in HTML to manage dynamic content, databases, session tracking, and even build entire e-commerce sites.
* **Usage in Quoteza**: Although Quoteza is built using Flutter and Firebase, PHP can be used if any web-based backend components are required, such as custom APIs or additional server-side functionality that isn’t handled by Firebase.

**DHTML (Dynamic HTML)**:

* **Purpose**: DHTML is an umbrella term for a collection of technologies used together to create interactive and animated web sites by using a combination of static markup languages (like HTML), a client-side scripting language (like JavaScript), the Document Object Model (DOM), and CSS.
* **Usage in Quoteza**: DHTML is primarily relevant for web applications. If Quoteza includes web views that require dynamic, interactive content, DHTML can be utilized to enhance user interaction.

**Primary Technology - Flutter**:

* **Purpose**: Flutter is a UI toolkit that enables the development of natively compiled applications for mobile, web, and desktop from a single codebase.
* **Usage in Quoteza**: Flutter is the main technology for developing the client side of Quoteza, providing a consistent and responsive user experience across different platforms.

### **4.4.5 Server-Side Technology**

The server side involves technologies and frameworks that handle data processing, storage, and business logic. For Quoteza, we utilize Firebase as the primary backend service, which is an open-source technology.

#### **4.4.5.1 Open-Source Technology**

**Firebase**:

* **Purpose**: Firebase provides a suite of cloud-based services, including a real-time database, Firestore, authentication, cloud storage, and serverless functions. It offers real-time data synchronization, offline capabilities, and robust security features.
* **Usage in Quoteza**: Firebase handles the backend functionalities of Quoteza, such as user authentication, real-time database management, cloud storage, and server-side logic through Firebase Cloud Functions.

**Additional Open-Source Technologies**:

* **Node.js**: Used for server-side scripting and running scripts server-side to produce dynamic web page content before the page is sent to the user’s web browser.
* **Express.js**: A web application framework for Node.js, designed for building web applications and APIs.

## 4.5 Code/Algorithms of Important Modules

This section provides an overview of the code and algorithms implemented in critical modules of Quoteza. We focus on user authentication, quote management, and API integration.

### **4.5.1 User Authentication Module**

**Objective**: To authenticate users and manage user sessions securely.

#### **Algorithm**:

1. **User Registration**:
   * Input: User’s email and password.
   * Process:
     1. User enters email and password.
     2. Validate the input data.
     3. Use Firebase Authentication to create a new user with email and password.
     4. Store additional user details in Firestore.
   * Output: User account created and user logged in.

// User Registration Code

Future<void> registerUser (String email, String password) async {

try {

UserCredential userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(

email: email,

password: password,

);

// Store additional user info in Firestore

await FirebaseFirestore.instance.collection('users').doc(userCredential.user.uid).set({

'email': email,

'createdAt': Timestamp.now(),

});

} catch (e) {

print("Error: $e");

}

}

#### User Login:

* Input: User’s email and password.
* Process:
  1. User enters email and password.
  2. Validate the input data.
  3. Use Firebase Authentication to sign in the user.
  4. Retrieve user details from Firestore.
* Output: User logged in and session initialized.

// User Login Code

Future<void> loginUser(String email, String password) async {

try {

UserCredential userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(

email: email,

password: password,

);

// Retrieve user info from Firestore

DocumentSnapshot userDoc = await FirebaseFirestore.instance.collection('users').doc(userCredential.user.uid).get();

print("User Info: ${userDoc.data()}");

} catch (e) {

print("Error: $e");

}

}

### **4.5.2 Quote Management Module**

**Objective**: To fetch, display, and manage quotes within the app.

#### **Algorithm**:

1. **Fetching Quotes**:
   * Input: None (initially) or user query.
   * Process:
     1. Make an API call to fetch quotes from a third-party service.
     2. Parse the JSON response to extract quote details.
     3. Store quotes in Firestore for offline access.
   * Output: List of quotes displayed to the user.

// Fetching Quotes Code

Future<void> fetchQuotes() async {

try {

final response = await http.get(Uri.parse('https://api.example.com/quotes'));

if (response.statusCode == 200) {

List<dynamic> quotes = jsonDecode(response.body);

// Store quotes in Firestore

quotes.forEach((quote) {

FirebaseFirestore.instance.collection('quotes').add(quote);

});

} else {

print("Failed to fetch quotes");

}

} catch (e) {

print("Error: $e");

}

}

### **4.5.3 API Integration Module**

**Objective**: To integrate third-party APIs for dynamic data fetching.

**Algorithm**:

1. **API Integration**:
   * Input: API endpoint.
   * Process:
     1. Send an HTTP GET request to the API endpoint.
     2. Handle the response and parse JSON data.
     3. Update the app's UI with the fetched data.
   * Output: Dynamic content from the API displayed in the app.

// API Integration Code

Future<void> fetchFromApi(String apiUrl) async {

try {

final response = await http.get(Uri.parse(apiUrl));

if (response.statusCode == 200) {

var data = jsonDecode(response.body);

// Update UI with fetched data

// Example: List of quotes or categories

} else {

print("Failed to fetch data");

}

} catch (e) {

print("Error: $e");

}

}

By detailing the client-side and server-side technologies along with the essential code and algorithms, this section provides a comprehensive view of the development and implementation process for Quoteza. This structured approach ensures a robust, scalable, and efficient application that meets user needs and project goals.

# **Chapter 05: Testing and Quality Assurance**

## 5.1 Testing Process

The testing process ensures that the Quoteza app meets all specified requirements and functions correctly. Testing identifies defects and verifies that the app works as intended. The process includes multiple types of testing, such as unit testing, integration testing, system testing, and user acceptance testing (UAT).

**Testing Phases**:

1. **Unit Testing**: Testing individual components or functions of the app to ensure they work correctly.
2. **Integration Testing**: Testing the interaction between different components to ensure they work together as expected.
3. **System Testing**: Testing the complete system to verify that it meets all requirements.
4. **User Acceptance Testing (UAT)**: Testing the app with potential users to gather feedback and ensure it meets their needs.

## 5.2 Test Case Design

Test case design involves creating detailed test cases that specify the inputs, execution conditions, and expected results for each test. These test cases help systematically verify the functionality and performance of the app.

**Components of a Test Case**:

* **Test Case ID**: A unique identifier for the test case.
* **Test Description**: A brief description of what the test case verifies.
* **Preconditions**: Any conditions that must be met before executing the test.
* **Test Steps**: Detailed steps to execute the test.
* **Expected Result**: The expected outcome of the test.
* **Actual Result**: The actual outcome observed after executing the test.
* **Status**: Pass or fail based on the comparison between expected and actual results.

**Example Test Case**:

* **Test Case ID**: TC001
* **Test Description**: Verify user registration with valid inputs.
* **Preconditions**: The app is installed and the registration screen is accessible.
* **Test Steps**:
  1. Open the app and navigate to the registration screen.
  2. Enter a valid email and password.
  3. Click the "Register" button.
* **Expected Result**: The user is successfully registered and redirected to the main screen.
* **Actual Result**: (To be filled after test execution)
* **Status**: (Pass/Fail)

## 5.3 Test Data

Test data is the set of inputs used to execute the test cases. It is essential to use a variety of test data to thoroughly test the app’s functionality and handle edge cases.

**Types of Test Data**:

* **Valid Data**: Data that meets all input criteria and should produce expected results.
* **Invalid Data**: Data that does not meet input criteria and should trigger error messages or validations.
* **Boundary Data**: Data at the edges of input ranges to test boundary conditions.
* **Edge Cases**: Unusual but possible inputs that test the app’s robustness.

**Example Test Data**:

* **Valid Data**:
  + Email: user@example.com
  + Password: StrongPass123!
* **Invalid Data**:
  + Email: user@.com
  + Password: pass
* **Boundary Data**:
  + Email: user@domain.com (minimal length)
  + Password: StrongPass12345678901234567890 (maximal length)
* **Edge Cases**:
  + Email: user@domain.c (invalid domain)
  + Password: empty string

## 5.4 Black Box Testing (Behavioral Testing)

Black box testing, also known as behavioral testing, involves testing the app’s functionality without knowing its internal code structure. The focus is on input-output verification to ensure the app behaves as expected.

**Types of Black Box Testing**:

* **Functional Testing**: Verifying that each function of the app works according to requirements.
* **Non-Functional Testing**: Testing non-functional aspects such as performance, usability, and security.

**Example Black Box Test**:

* **Test Case**: Verify the "Forgot Password" functionality.
* **Test Steps**:
  1. Open the app and navigate to the login screen.
  2. Click the "Forgot Password" link.
  3. Enter a registered email address and submit.
* **Expected Result**: The app sends a password reset email to the registered email address.

## 5.5 White Box Testing (Structural Testing)

White box testing, also known as structural testing, involves testing the internal code structure and logic of the app. This type of testing ensures that the code is functioning as intended and covers all possible execution paths.

**Types of White Box Testing**:

* **Unit Testing**: Testing individual functions or methods in the code.
* **Integration Testing**: Testing the interaction between integrated components.
* **Code Coverage**: Ensuring that all parts of the code are executed during testing.

**Example White Box Test**:

* **Test Case**: Verify the function that validates email format.
* **Test Steps**:
  1. Call the email validation function with various inputs (valid and invalid emails).
  2. Check the function's output.
* **Expected Result**: The function correctly identifies valid and invalid email formats.

// Example Dart code for email validation function

bool validateEmail(String email) {

String pattern = r'^[^@]+@[^@]+\.[^@]+';

RegExp regex = RegExp(pattern);

return regex.hasMatch(email);

}

// Unit Test for validateEmail function

void testValidateEmail() {

assert(validateEmail('user@example.com') == true);

assert(validateEmail('user@.com') == false);

assert(validateEmail('user@domain.c') == false);

assert(validateEmail('') == false);

}

These structured testing processes and methodologies, we can ensure that Quoteza is reliable, functional, and user-friendly. Comprehensive testing helps identify and fix issues early, leading to a higher quality application.

# **Chapter 06: User Guide**

## 6.1 Input Forms

Input forms are crucial components of the Quoteza app, allowing users to interact with the application by providing necessary information. Here, we provide a detailed overview of the key input forms, including screenshots and descriptions.

### **6.1.1 User Registration Form**

**Description**: The user registration form allows new users to create an account on Quoteza. The form collects essential information such as the user's email and password.

**Fields**:

* **Email**: The user's email address. This field is required and must be in a valid email format.
* **Password**: The user's password. This field is required and must meet the app's password policy (e.g., minimum 8 characters, including letters and numbers).

**Screenshot**:

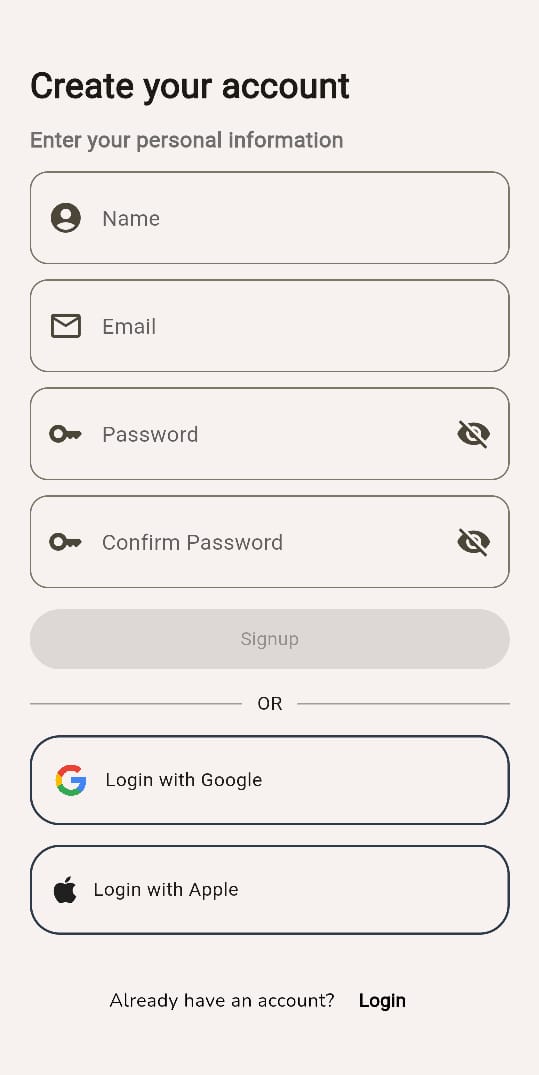


Figure 6.1.1. 1 User Registration Screen

**Instructions**:

1. Enter a valid email address in the "Email" field.
2. Enter a strong password in the "Password" field.
3. Click the "Register" button to create your account.

### **6.1.2 User Login Form**

**Description**: The user login form allows existing users to access their Quoteza account by providing their email and password.

**Fields**:

* **Email**: The user's email address used during registration.
* **Password**: The user's password.

**Screenshot**:

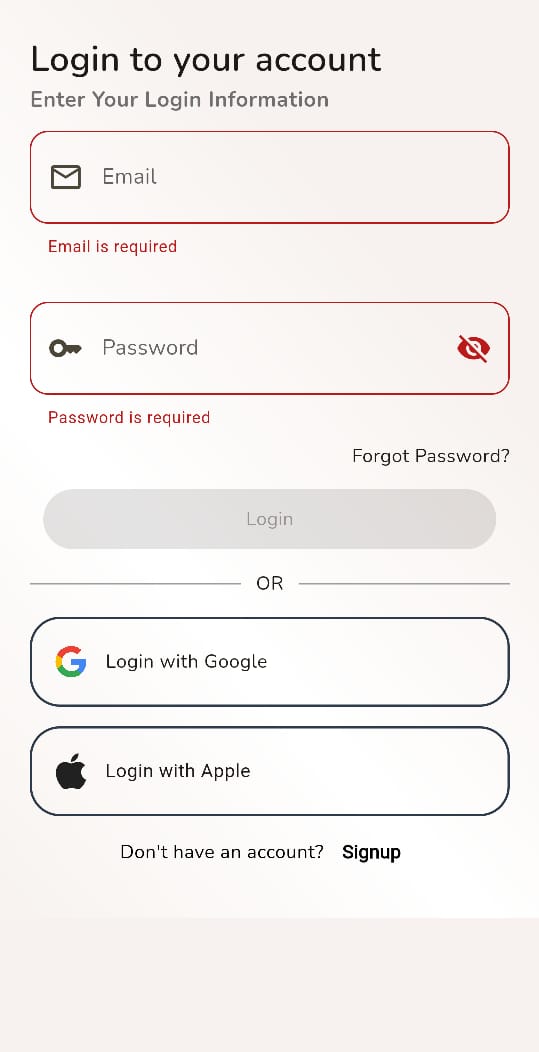


Figure 6.1.2. 1 User Login Form

**Instructions**:

1. Enter your registered email address in the "Email" field.
2. Enter your password in the "Password" field.
3. Click the "Login" button to access your account.

### **6.1.3 Favorite Quote Form**

**Description**: The favorite quote form allows users to save their favorite quotes to their personal list for easy access later.

**Fields**:

* **Quote ID**: The identifier of the quote to be saved as a favorite (hidden field).
* **User ID**: The identifier of the user saving the quote (hidden field).

**Screenshot**:

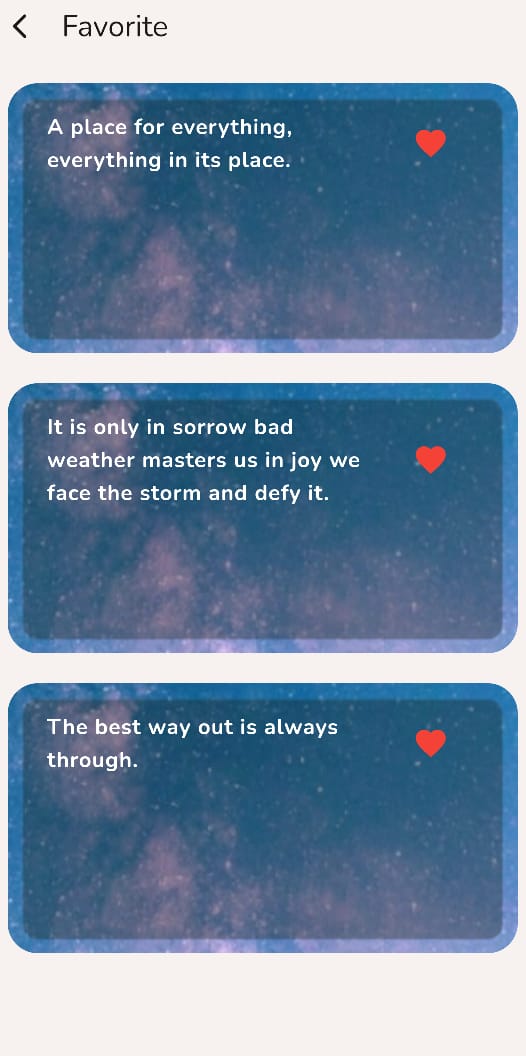


Figure 6.1.3. 1 Favorites Screen

**Instructions**:

1. Navigate to a quote you wish to save.
2. Click the "Add to Favorites" button to save the quote.

### **6.1.4 Profile Management**

Profile management in Quoteza allows users to view and edit their personal information, manage their saved quotes, and configure their preferences.

**Features**:

* **View Profile**: Users can view their personal details such as name, email, and profile picture.
* **Edit Profile**: Users can update their personal information and change their profile picture.
* **Manage Favorites**: Users can view and manage their list of favorite quotes.
* **Settings**: Users can configure app settings, such as notification preferences and account security.

**Screenshot**:

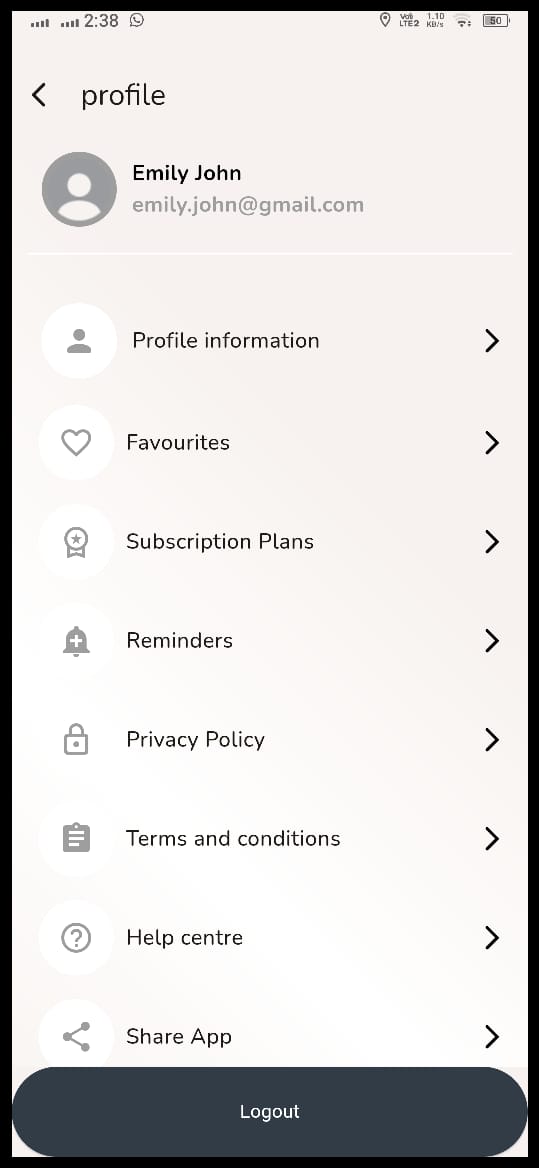


Figure 6.1.4. 1 Profile Management

**Instructions**:

1. **Accessing Profile**:
   * Tap on the profile icon located in the bottom navigation bar.
2. **Viewing Profile Details**:
   * The profile screen displays the user's name, email, and profile picture.
3. **Editing Profile**:
   * Tap on the "Edit Profile" button.
   * Update the necessary fields and tap "Save" to apply changes.
4. **Managing Favorites**:
   * Tap on the "Favorites" tab to view the list of saved quotes.
   * Swipe left on a quote to delete it from the favorites list.
5. **Settings**:
   * Tap on the "Settings" icon to configure notification preferences and security settings.

### **6.1.5 Home Screen**

The home screen of Quoteza is the main dashboard where users can view quotes, search for new quotes, and access various features of the app.

**Features**:

* **Daily Quote**: Displays a new quote every day.
* **Navigation**: Quick access to other features like profile, favorites, and settings.

**Screenshot**:

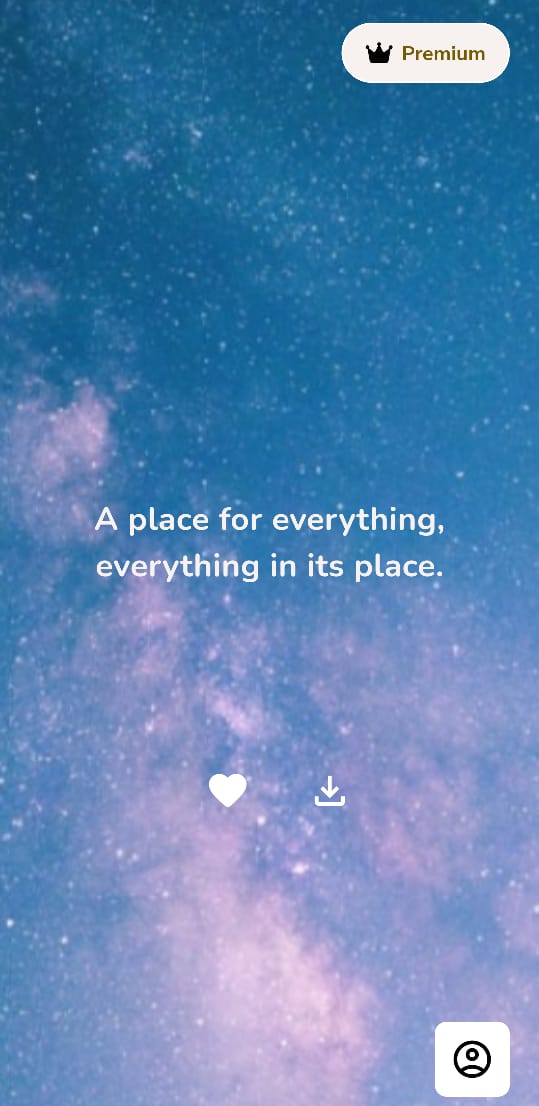


Figure 6.1.5. 1 Home Screen

**Instructions**:

1. **Viewing Daily Quote**:
   * The daily quote is prominently displayed at the top of the home screen.
2. **Navigating**:
   * Use the bottom navigation bar to quickly switch between home, profile, favorites, and settings.

### **6.1.6 Subscription Screen**

The subscription screen allows users to upgrade to a premium version of Quoteza for additional features and benefits.

**Features**:

* **Subscription Plans**: Different plans with varying features and prices.
* **Benefits Overview**: Detailed list of benefits for subscribing to the premium version.
* **Payment Options**: Secure payment options for subscribing.

**Screenshot**:

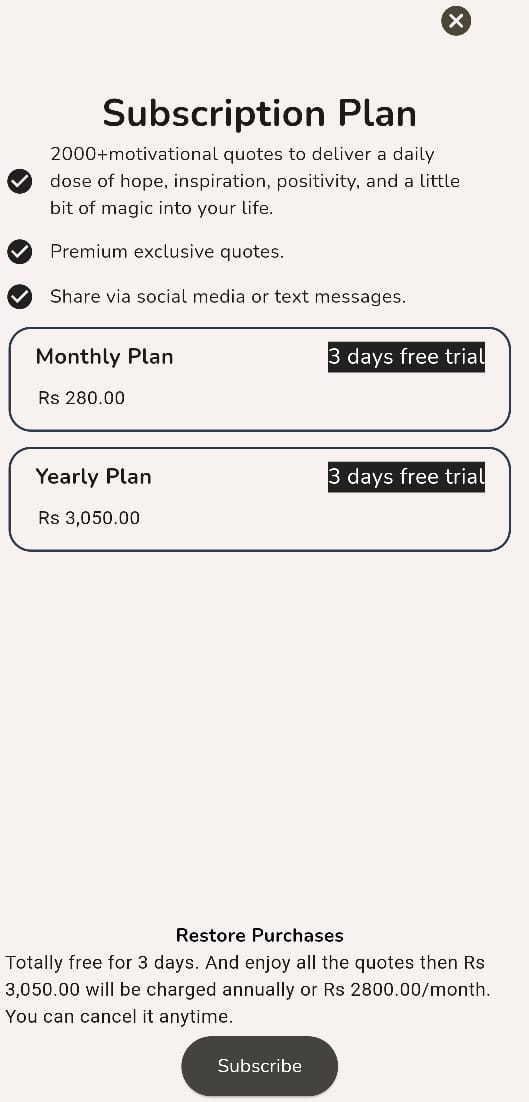


Figure 6.1.6. 1 Subscription Plan

**Instructions**:

1. **Accessing the Subscription Screen**:
   * Tap on the "Subscribe" button located in the settings or on the home screen.
2. **Viewing Subscription Plans**:
   * The screen displays available subscription plans.
   * Tap on each plan to view its details and benefits.
3. **Choosing a Plan**:
   * Select the desired subscription plan by tapping on the "Subscribe" button next to it.
4. **Making a Payment**:
   * Enter payment details in the secure payment form.
   * Confirm the payment to complete the subscription process.

### **6.1.7 Reminders Screens**

The reminders feature in Quoteza allows users to set reminders for their favorite quotes or motivational messages.

**Features**:

* **Set Reminders**: Schedule reminders for specific times and dates.
* **Manage Reminders**: View, edit, or delete existing reminders.
* **Notification Settings**: Configure how and when reminders are delivered.

**Screenshot**:

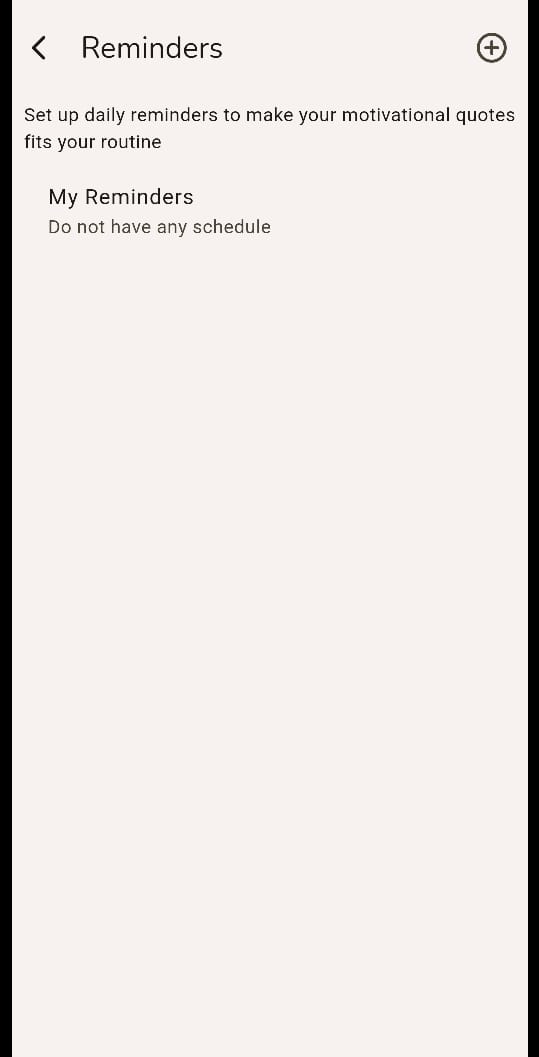


Figure 6.1.7. 1 Reminders Screen

**Instructions**:

1. **Accessing the Reminders Screen**:
   * Tap on the "Reminders" icon in the settings menu or on the profile screen.
2. **Setting a New Reminder**:
   * Tap on the "Add Reminder" button.
   * Select the quote you want to set a reminder for.
   * Choose the date and time for the reminder and save.
3. **Managing Existing Reminders**:
   * The reminders screen lists all active reminders.
   * Tap on a reminder to edit its details or delete it.
4. **Configuring Notification Settings**:
   * Tap on the "Notification Settings" option.
   * Choose how you want to be notified (e.g., push notification, email).
   * Set your preferred notification times and save the settings.

# **Chapter 07: Conclusion**

## Project Summary: Development of Quoteza App

**Project Title**: Quoteza - A Mobile Application for Inspirational Quotes

**Project Overview**: The Quoteza app is a comprehensive mobile application designed to provide users with daily inspirational quotes. Developed using Flutter and Firebase, the app offers features such as user registration, quote search, favorites management, reminders, and premium subscription options. The primary goal of this project was to create a user-friendly platform that motivates and inspires users through curated quotes from various authors and categories.

### **Key Tasks and Activities**

#### **1. Project Introduction and Feasibility Study**:

* **Main Theme**: The main theme of Quoteza is to deliver daily inspirational quotes to users, helping them stay motivated and positive.
* **Scope of the Project**: The scope included the development of a mobile application with core features like user registration, quote search, favorites, reminders, and subscription management.
* **Objectives**: Objectives included creating an intuitive user interface, integrating robust backend support with Firebase, and providing premium features through a subscription model.

#### **2. System Analysis**:

* **Feasibility Study**: Conducted a feasibility study to assess the technical, economic, and operational viability of using Flutter and Firebase. The study confirmed that these technologies were suitable for developing a scalable, cross-platform mobile app with real-time database capabilities.
* **Data Gathering and Analysis**: Collected data using questionnaires and observations to understand user needs and preferences. This data was analyzed to define requirements and create data flow diagrams (DFDs) for visualizing the app’s data processes.

#### **3. System Design**:

* **UML Design**: Utilized UML diagrams to plan the app’s structure and interactions. Created use case diagrams to outline user interactions and sequence diagrams to detail the flow of operations.
* **Database Design**: Developed Entity Relationship Diagrams (ERDs) to model the database structure. Transformed ERDs into a relational model and normalized it to the third normal form (3NF) for efficiency.

#### **4. System Development**:

* **Tool/Language Selection**: Chose Flutter for the frontend development due to its cross-platform capabilities and Firebase for backend services due to its real-time database and authentication features.
* **Hardware and Software Implementation**: Implemented the app on standard mobile devices with the necessary software tools for development.
* **Client-Side Technology**: Utilized Dart programming language within Flutter for building the user interface.
* **Server-Side Technology**: Leveraged Firebase for server-side operations, including database management and user authentication.

#### **5. Testing**:

* **Testing Process**: Conducted a comprehensive testing process to ensure the app’s functionality, usability, and performance.
* **Test Case Design**: Created detailed test cases covering all features and functionalities.
* **Black Box and White Box Testing**: Performed black box testing to validate the app’s behavior against user requirements and white box testing to ensure the internal code structure was sound.

#### **6. User Guide**:

* **Input Forms**: Detailed documentation on how users can interact with the app through various input forms, including registration, login, and quote search.
* **Reports**: Provided samples of key reports such as user activity, favorite quotes, and system usage, to help users understand their engagement with the app.
* **Profile Management**: Instructions on how users can manage their profile, including viewing and editing personal information, managing subscriptions, and configuring settings.
* **Home, Subscription, and Reminders Screens**: Guidance on navigating the home screen, managing subscriptions, and setting up reminders for favorite quotes.

**Conclusion**: The Quoteza app was successfully developed and tested, meeting the outlined objectives and providing a robust platform for delivering daily inspiration to users. The project’s comprehensive approach, from feasibility study to user guide documentation, ensures a well-rounded and user-friendly application. Future enhancements could include additional personalization features and expanded quote databases to further enrich user experience.