Malaysia's Intelligent Mobile Advisor

By

OOI YU ZHEN

FACULTY OF COMPUTING AND
INFORMATION TECHNOLOGY

TUNKU ABDUL RAHMAN UNIVERSITY OF
MANAGEMENT AND TECHNOLOGY
KUALA LUMPUR

ACADEMIC YEAR
2025/26

DialSmart: Malaysia's Intelligent Mobile Advisor

By

OOI YU ZHEN

Supervisor: DR. NG YEN PHING

A project report submitted to the
Faculty of Computing and Information Technology
in partial fulfillment of the requirement for the
Bachelor of Information Systems (Honours)

Department of Information Systems and Security
Faculty of Computing and Information Technology
Tunku Abdul Rahman University of Management and Technology
Kuala Lumpur

Declaration

The project submitted herewith is a result of my own efforts in totality and in every aspect of the project works. All information that has been obtained from other sources had been fully acknowledged. I understand that any plagiarism, cheating or collusion or any sorts constitutes a breach of TAR University rules and regulations and would be subjected to disciplinary actions.

_____

OOI YU ZHEN

Bachelor of Information Systems (Honours) in Enterprise Information Systems

ID: 24WMR08926

Table of Contents

Chapter 1

Introduction

1 Introduction

Malaysia's Intelligent Mobile Advisor represents an innovative approach to addressing the overwhelming complexity of smartphone selection in the Malaysian market through artificial intelligence and machine learning technologies. The project addresses significant research gaps in localized technology advisory systems, particularly the absence of culturally aware and economically contextualized smartphone recommendation platforms tailored for Malaysian consumers. The system integrates four core modules: the Admin Management Module for comprehensive data governance, Brand and Phone Details pages for localized product information, an AI-powered Chatbot Module for intuitive user interaction, and Machine Learning Model Development for personalized recommendations that consider Malaysian market dynamics, cultural preferences, and economic constraints.

The scope of Malaysia's Intelligent Mobile Advisor encompasses the development of intelligent recommendation algorithms trained on Malaysian consumer behavior patterns, comprehensive brand coverage including Samsung, Apple, Huawei, Nokia, Lenovo, and XIAOMI with localized pricing and availability data, and a conversational AI system capable of understanding Malaysian English variations and cultural context. The project excludes international market data beyond Malaysian relevance, hardware repair recommendations, and real-time inventory management for retail partners.

1.1 Problem Statement

The Malaysian smartphone market presents several critical challenges that significantly impact consumer decision-making processes and overall satisfaction with technology purchases. These problems highlight the urgent need for a localized, intelligent advisory system specifically designed for Malaysian consumers.

1.1.1 Overwhelming Product Complexity and Choice Overload

Malaysian consumers face unprecedented complexity when selecting smartphones from over 1,400 models available globally (Counterpoint Research, 2024), with major brands like Samsung, Apple, Huawei, Nokia, Lenovo, and XIAOMI offering multiple variants across different price ranges. This overwhelming variety creates choice paralysis, where consumers struggle to evaluate technical specifications, compare features meaningfully, and make confident purchasing decisions. The abundance of options paradoxically decreases decision satisfaction and often results in suboptimal purchases that don't align with users' actual needs or budget constraints (Emerald Publishing, 2025).

1.1.2 Knowledge Gap Between Technical Specifications and Consumer Understanding

Current smartphone advisory platforms present technical information using complex terminology that requires specialized knowledge most consumers lack. Specifications involving advanced processors, camera systems, connectivity standards, and emerging technologies create significant barriers for non-specialist users. Malaysian consumers, representing diverse educational backgrounds and varying technological literacy levels, experience difficulty translating technical specifications into practical benefits, leading to either over-investment in unnecessary features or under-investment in capabilities that would enhance their mobile experience.

1.1.3 Lack of Localized and Culturally-Aware Recommendation Systems

Existing global smartphone advisory platforms fail to address the unique demographic, cultural, and economic characteristics of Malaysian consumers. International platforms like GSMArena provide comprehensive technical databases but lack cultural contextualization, economic sensitivity, and regional availability intelligence necessary for effective Malaysian market application. These platforms rely on generic global algorithms that don't consider Malaysia's multicultural society, diverse income levels, local pricing patterns, or region-specific usage behaviors, creating a significant gap in relevant, actionable guidance.

1.1.4 Inadequate User Experience for Diverse Demographics

Traditional recommendation systems employ static, filter-based approaches that don't accommodate the varying needs of Malaysia's diverse population segments. Students, working professionals, and elderly users have distinctly different technological requirements, budget constraints, and usage patterns, yet current platforms offer one-size-fits-all solutions. The absence of conversational, intuitive interfaces particularly disadvantages elderly users and those with limited technical expertise, preventing them from accessing personalized guidance that matches their specific circumstances and preferences.

1.2 Project Objectives

The Malaysia's Intelligent Mobile Advisor project aims to systematically address each identified problem through targeted objectives that deliver comprehensive solutions for Malaysian smartphone consumers.

1.2.1 Simplify Complex Decision-Making Through Intelligent Filtering and Recommendation

To develop an advanced filtering and recommendation system that reduces choice overload by intelligently narrowing smartphone options based on user preferences, budget constraints, and usage patterns.

This objective directly addresses Problem 1.1.1 by implementing sophisticated algorithms that eliminate overwhelming product complexity. The system will provide intelligent filtering capabilities including price range filters, 5G support filters, and usage-based filters (gaming, photography, business use) to help users focus on relevant options. Through machine learning models trained on Malaysian consumer behavior patterns, the platform will present curated recommendations that match individual needs, transforming the traditionally overwhelming selection process into a manageable, confidence-building experience that leads to optimal purchasing decisions.

1.2.2 Bridge Technical Knowledge Gaps Through Conversational AI Interface

To create an AI-powered chatbot system with natural language processing capabilities that translates complex technical specifications into understandable, practical guidance for Malaysian users.

This objective solves Problem 1.1.2 by developing a conversational interface that interprets user queries expressed in natural language and provides intelligent responses in easily comprehensible terms. The chatbot will understand Malaysian English variations and cultural context, eliminating the need for users to master complex technical terminology. By integrating rule-based logic with adaptive machine learning models, the system will explain smartphone features in practical terms, helping users understand how technical specifications translate into real-world benefits and usage scenarios.

1.2.3 Develop Culturally-Aware and Economically-Sensitive Localization

To implement comprehensive Malaysian market localization including cultural preference integration, local pricing analysis, demographic categorization, and region-specific recommendation algorithms.

This objective addresses Problem 1.1.3 by creating a platform specifically calibrated for Malaysian consumers rather than relying on generic global approaches. The system will

incorporate localized evidence including Malaysian pricing trends, regional availability patterns, cultural preferences, and demographic-specific usage behaviors. Through comprehensive curation of smartphone data relevant to Malaysian consumers and integration of real-world Malaysian market conditions, the platform will provide value propositions and recommendations that remain both relevant and actionable within specific economic and cultural contexts.

1.2.4 Enhance Digital Accessibility Through Demographic-Specific User Experience Design

To design intuitive, accessible user interfaces and personalized recommendation approaches that accommodate Malaysia's diverse population segments including students, working professionals, and elderly users.

This objective solves Problem 1.1.4 by implementing demographic categorization approaches that acknowledge Malaysia's diverse population needs while providing tailored experiences. The system will feature natural language interfaces that reduce technological barriers for elderly users and those with limited technical expertise, while offering sophisticated analysis tools for tech-savvy professionals. Through personalized profile management, historical recommendation tracking, and adaptive interface design, the platform will empower all user categories to make confident technology decisions based on their specific needs rather than requiring complex technical analysis skills.

1.3 Project Background

The contemporary smartphone landscape presents unprecedented challenges that highlight significant research gaps in consumer decision-support systems, particularly within the Malaysian market context. Current smartphone advisory platforms demonstrate a critical population gap by failing to address the unique demographic, cultural, and economic characteristics of Malaysian consumers. The theoretical gap becomes evident in the absence of localized recommendation frameworks that consider Malaysia's multicultural society, diverse economic backgrounds, and varying technological literacy levels across different age groups and professional categories.

Contemporary research reveals a notable empirical gap in understanding how overwhelming product variety affects consumer decision-making quality in technology purchases. Recent market data indicates that global smartphone shipments totaled 1.22 billion units in 2024, reflecting a 7% year-on-year increase (Canalys, 2024), with the global smartphone market growing 4% with 316.1 million units shipped in Q3 2024 alone (IDC, 2024). This unprecedented market expansion creates what behavioral economists term "choice overload" -

a phenomenon where increased options paradoxically decrease decision satisfaction and efficiency. Current research validates that choice overload causes psychological phenomena (EWA Digital Repository, 2024) such as decision-making difficulties, anxiety and dissatisfaction due to increased cognitive load when faced with numerous options. A comprehensive systematic review published in 2025 (Emerald Publishing, 2025) demonstrates the "less is better" paradox in consumer behavior, confirming choice overload's significant marketing implications.

The knowledge gap becomes particularly pronounced when examining how technical terminology creates barriers for non-specialist consumers. Market leadership data shows Apple leading the global smartphone market in Q4 2024 with 23% share, while Samsung held 16% (Counterpoint Research, 2024) and Xiaomi maintained third position. Specifications involving advanced processors, multiple camera systems, connectivity standards, and emerging technologies require specialized knowledge that most consumers lack. Recent research on consumer decision-making in the era of information overload (ResearchGate, 2024) demonstrates how the deluge of information creates new paradigms in modern digital age consumer behavior. Malaysian consumers, representing diverse educational backgrounds and technological familiarity levels, experience these challenges with varying intensity, creating a population gap that existing solutions fail to address systematically.

Current smartphone advisory platforms demonstrate a notable practical-knowledge gap by failing to translate global product information into locally relevant, actionable guidance for Malaysian consumers. International platforms such as GSMArena provide comprehensive technical databases but lack the cultural contextualization, economic sensitivity, and regional availability intelligence necessary for effective Malaysian market application. Recent market dynamics show significant regional variations, with sales of ultra-premium smartphones (priced above $1000) growing fastest in 2024 (Counterpoint Research, 2024) as consumers showed preference for spending more on their next smartphone. The choice overload effect in online recommender systems research (INFORMS, 2024) identifies important but understudied questions in personalized recommendation settings, emphasizing the growing demand for technology solutions that address methodology gaps in recommendation systems. According to McKinsey's State of the Consumer 2025 report, disruption has become permanent in consumer behavior, requiring advisory systems to adapt to rapidly changing consumer expectations (McKinsey & Company, 2025).

The contemporary smartphone advisory landscape reveals significant methodological gaps in how recommendation systems address consumer decision-making challenges. Research from The Decision Lab confirms that choice overload bias describes how people get overwhelmed when presented with many options, despite assumptions that more choice is beneficial (The

Decision Lab, 2024). Current technological approaches demonstrate limitations in bridging the gap between technical specifications and practical consumer understanding, with existing platforms failing to provide transparent decision pathways that enable users to understand recommendation reasoning. The digital inclusion landscape reveals significant gaps in accommodating diverse demographic segments effectively, as recent analysis demonstrates that choice overload causes analysis paralysis across different industries, particularly affecting vulnerable consumer segments including elderly users and those with limited technical expertise (Al-Kindi Center for Research and Development, 2024). These gaps create opportunities for innovative approaches that can bridge accessibility barriers while maintaining recommendation quality across Malaysia's diverse consumer base.

1.4 Advantages and Contributions

Malaysia's Intelligent Mobile Advisor may delivers contributions across multiple research domains while addressing critical gaps in current smartphone advisory systems. The project's innovations span technological advancement, market-specific solution development, and social impact through enhanced digital accessibility, collectively contributing to the broader research landscape of intelligent recommendation systems and localized technology advisory platforms.

1.4.1 Technological Innovation and Differentiation

Malaysia's Intelligent Mobile Advisor directly addresses Objective 1.2.1 by developing advanced filtering and recommendation algorithms that systematically reduce choice overload through intelligent product curation. The platform addresses the critical challenge identified in contemporary research, where choice overload causes analysis paralysis across different industries, particularly affecting consumer decision-making in technology purchases (Al-Kindi Center for Research and Development, 2024).

The system's hybrid approach combining conversational artificial intelligence with machine learning-based analysis represents a methodological advancement over traditional filter-based platforms. Unlike static product listing systems, Malaysia's Intelligent Mobile Advisor implements transparent recommendation algorithms that provide traceable decision pathways, enabling users to understand why specific devices are recommended. This addresses the empirical gap in choice overload research, where recent studies show that too many choices can be detrimental to decision-making.

The platform's intelligent filtering capabilities, including price range filters, 5G support filters, and usage-based filters (gaming, photography, business use), directly combat the overwhelming complexity consumers face when choosing from major brands like Samsung (21.4% market share), Apple (18.8%), Xiaomi (13.2%), and others across multiple price segments and feature

configurations (Counterpoint Research, 2024). By transforming the traditionally overwhelming selection process into manageable, confidence-building experiences, the system enables optimal purchasing decisions while addressing what behavioral economists term the "paradox of choice".

1.4.2 Conversational AI Bridge for Technical Knowledge Translation

The platform's AI-powered chatbot system with natural language processing capabilities directly fulfills Objective 1.2.2 by translating complex technical specifications into understandable, practical guidance specifically designed for Malaysian users. This addresses the significant knowledge gap where technical terminology creates barriers for non-specialist consumers navigating the contemporary smartphone market.

Current market dynamics demonstrate the urgency of this solution, with global smartphone shipments reaching 1.22 billion units in 2024, marking a 7% growth and creating unprecedented complexity in consumer choice (Canalys, 2024). The system's natural language interface reduces technological barriers by interpreting user queries expressed in conversational Malaysian English and providing intelligent responses in easily comprehensible terms.

The integration of rule-based logic with adaptive machine learning models ensures both contextual accuracy and continuous improvement through user interaction learning. This technological approach addresses research findings showing that choice overload describes how people get overwhelmed when presented with many options, despite assumptions that more choice is beneficial (The Decision Lab, 2024). By explaining smartphone features in practical terms rather than technical jargon, the platform empowers users to understand how specifications translate into real-world benefits and usage scenarios..

1.4.3 Malaysian Market Localization and Cultural Integration

Malaysia's Intelligent Mobile Advisor directly implements Objective 1.2.3 through comprehensive Malaysian market localization that incorporates cultural preference integration, local pricing analysis, and demographic categorization. This addresses the critical population gap where existing global platforms fail to consider Malaysia's unique multicultural society, diverse income levels, and regional usage behaviors.

The platform's market-specific value creation represents a substantial contribution to addressing localization gaps in technology advisory services. Through comprehensive curation of smartphone data relevant to Malaysian consumers, including real-time pricing analysis from major brands like Samsung, Apple, Huawei, Nokia, Lenovo, and XIAOMI, the system provides value propositions that international platforms cannot replicate.

Current market data validates this approach's necessity, with Samsung leading global market share at 19% and Apple maintaining strong positioning, while regional variations in consumer preferences and economic constraints require localized advisory approaches (Counterpoint Research, 2024). The system's recommendation algorithms incorporate real-world Malaysian market conditions, ensuring users receive advice that remains both relevant and actionable within their specific economic and cultural contexts.

The localization extends beyond surface-level currency conversion to encompass deeper understanding of Malaysian consumer behavior, economic decision-making patterns, and technology adoption characteristics across the nation's diverse demographic segments.

1.4.4 Enhanced Digital Accessibility Through Demographic-Specific Design

The platform's intuitive, accessible user interface design directly fulfills Objective 1.2.4 by accommodating Malaysia's diverse population segments including students, working professionals, and elderly users through personalized recommendation approaches. This addresses the practical-knowledge gap that prevents many consumers from making informed technology decisions.

Research confirms that choice overload causes negative psychology such as fear of missing out and decision-making delay, particularly affecting vulnerable consumer segments including elderly users and those with limited technical expertise (EWA Digital Repository, 2024). Malaysia's Intelligent Mobile Advisor's demographic categorization approach acknowledges these diverse population needs while providing tailored experiences that consider economic constraints, technological familiarity, and usage pattern variations.

The system's natural language interface reduces technological barriers through conversational guidance rather than complex technical analysis requirements. This contributes meaningfully to digital inclusion by transforming complex technical decision-making into accessible, user-friendly experiences that support broader technology adoption aligned with Malaysia's digital transformation initiatives.

The platform addresses decision-making stress and analysis paralysis through intelligent simplification of complex technical comparisons, contributing to improved consumer satisfaction and more informed purchasing decisions across Malaysia's diverse smartphone market. By providing transparent, conversational guidance, the system builds consumer confidence while promoting better understanding of technology choices and their practical implications.

1.5 Project Plan

The project follows an Incremental Development Model with Agile practices, ensuring systematic progress through clearly defined milestones. This approach allows for continuous refinement and improvement throughout the development process.

Table 1.1 Project Plan

No.	Task Name	Duration	Start Date	Finish Date
Phase 1: Project Planning & Requirements (104 days)				
1.	**Project Kick-off and Requirement Analysis**	5 days	3/3/25	7/3/25
2.	**Approaching Supervisor**	1 day	7/3/25	7/3/25
3.	**Project Planning and Proposal Development**	33 days	10/3/25	11/4/25
	• Finalized project scope and requirements • Database schema and ERD diagrams • Initial database setup			
4.	**Submit Project Proposal (Form 1, 2 & 3)**	1 day	18/4/25	18/4/25
5.	**Moderation of Project Proposal**	2 day	22/5/25	23/5/25
6.	**System Overview Discussion and Planning**	26 days	26/5/25	20/6/25
7.	**Literature Review and Researching Related Works**	35 days	26/5/25	29/6/25
	• Malaysian mobile market research • AI chatbot technology review • Recommendation system studies			
8.	**Submit Chapter 1: Introduction**	1 day	30/6/25	30/6/25
Phase 2: System Analysis & Design (67 days)				
9.	**System Analysis**	16 days	1/7/25	16/7/25
	• Requirements Analysis and Interview • User categorization (Student, Worker, Elder) • Malaysian market analysis • Survey and Questionnaire design			
10.	**Submit Chapter 2: Research Background**	1 day	14/7/25	14/7/25
11.	**System Methodology and Development Planning**	16 days	17/7/25	1/8/25
	• Incremental Development Model planning • Agile practices implementation • Software development methodology			
12.	**Submit Chapter 3: Methodology and Requirements Analysis**	1 day	28/7/25	28/7/25

Table 1.1 Project Plan (continued)

13.	**System Planning and Design**	28 days	29/7/25	25/8/25
	• Admin Panel UI/UX Design	8 days		
	• Database Design and Setup	7 days		
	• Brand and Phone Detail Pages Architecture	7 days		
	• System Design Specification: Diagrams	6 days		
14.	**Submit Chapter 4: System Design**	1 day	18/8/25	18/8/25
15.	**Project I Portfolio Compilation**	3 days	5/9/25	7/9/25
16.	**Submit Project I Portfolio (Individual)**	1 day	8/9/25	8/9/25
Phase 3: System Implementation (63 days)				
17.	**Admin Panel Development**	21 days	26/8/25	15/9/25
	• CRUD operations for phone data • User management system • Category management • Dashboard analytics			
18.	**Brand and Phone Detail Pages Development**	14 days	16/9/25	29/9/25
	• Samsung, Apple, Huawei pages • Nokia, Lenovo, XIAOMI pages • Filterable specifications display			
19.	**Data Collection and Preprocessing for ML**	10 days	30/9/25	9/10/25
	• Malaysian phone market data collection • User preference categorization • Data cleaning and preprocessing			
20.	**ML Model Development and Training**	18 days	10/10/25	27/10/25
	• Decision Tree algorithm implementation • Random Forest model training • Model accuracy testing and validation • Google Colab integration			
Phase 4: System Integration & Testing (38 days)				
21.	**Chatbot Integration with ML Model**	10 days	28/10/25	6/11/25
	• Natural language processing setup • Flask backend integration • Conversational AI implementation			

Table 1.1 Project Plan (continued)

22.	**System Integration and Testing**	11 days	7/11/25	17/11/25
	• Module integration testing • API testing and debugging • Performance optimization			
23.	**Preparation of Test Plan and System Preview**	4 days	18/11/25	21/11/25
24.	**System Preview with Supervisor**	2 day	20/11/25	21/11/25
25.	**System Refinement**	7 days	22/11/25	28/11/25
26.	**Final System Testing with Supervisor and Moderator**	2 day	27/11/25	28/11/25
27.	**Initial System Preview**	2 day	25/9/25	26/9/25
Phase 5: System Documentation & Finalization (21 days)				
28.	**System Implementation, Delivery and Testing (Iterative)**	14 days	29/11/25	12/12/25
	• Final bug fixes and optimization • User acceptance testing • System refinement			
29.	**Submit Draft FYP Report (Chapter 5: Result)**	1 day	12/12/25	12/12/25
30.	**Final Documentation and System Preparation**	5 days	13/12/25	18/12/25
	• Chapter 6: Conclusion completion • Final system documentation • All deliverables preparation			
31.	**Submit Final FYP Report (Individual) and Deliverables**	1 day	19/12/25	19/12/25

1.5.1 Gantt Chart

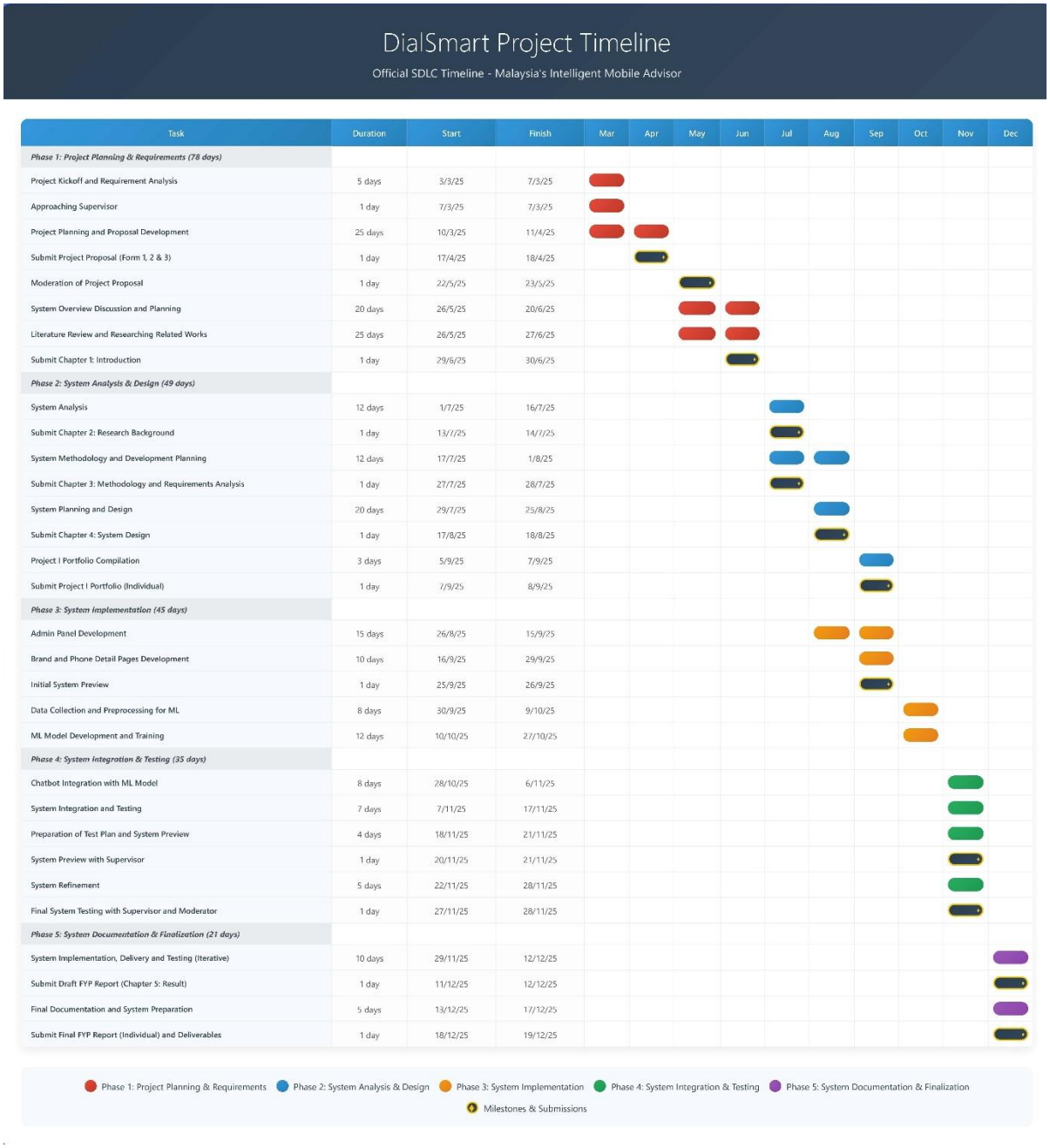

Figure 1.1 Gantt Chart of Project Timeline

1.6 Project Team and Organization

Table 1. 2 Project team and organization

Systems and Sub-systems	Ooi Yu Zhen (YZ)	Gan Shi Yun (SY)
Admin Management Module		
Admin Login/Register	✓	
Admin Dashboard Overview	✓	
CRUD Phone Brands	✓	
CRUD Phone Models & Specs	✓	
CRUD Categories (Student/Worker/Elder)	✓	
User Data Management	✓	
System Log/History	✓	
User Management Module		
User Login/Register		✓
Edit Profile & Preferences		✓
View Past Comparisons/Recommendations		✓
Brand & Phone Details Module		
Samsung Brand Page	✓	
Huawei Brand Page	✓	
Nokia Brand Page	✓	
Lenovo Brand Page	✓	
XIAOMI Brand Page	✓	
Apple Brand Page		✓

Table 1.2 Project team and organization (continued)

Honor Brand Page		✓
Oppo Brand Page		✓
Realme Brand Page		✓
Vivo Brand Page		✓
Phone Comparison Module		
Select Up to 2 Phones		✓
Side-by-side Specs Comparison		✓
Export/Share Comparison		✓
AI Chatbot Module		
Frontend Chat Widget	✓	
Natural Language Processing	✓	
ML Model Integration	✓	
FAQ & Fallback Responses	✓	
AI Model Development		
Data Cleaning & Processing	✓	✓
Feature Engineering	✓	
Model Training (Decision Tree/Random Forest)	✓	
Model Export & Backend Connection	✓	
Preference Form Design		✓

Table 1.2 Project team and organization (continued)

Shared Modules (Both Contribute)		
Recommendation System Core Logic	✓	✓
Frontend/UI Template System	✓	✓
Database Design	✓	✓
Phone Finder/Filter Module	✓	✓
Latest News/Release Tracker	✓	✓
Documentation & User Guide	✓	✓

✓ - Individual responsibility

✓ - Shared responsibility (both members contribute)

1.7 Chapter Summary and Evaluation

This chapter has established the foundational context for Malaysia's Intelligent Mobile Advisor by identifying key problems in the Malaysian smartphone market and defining clear objectives to address them. The project successfully addresses four critical challenges: overwhelming product complexity, technical knowledge gaps, lack of localized recommendation systems, and inadequate user experience for diverse demographics.

Malaysia's Intelligent Mobile Advisor delivers a comprehensive solution through four core modules that combine machine learning, natural language processing, and culturally-aware localization specifically designed for Malaysian consumers. The system transforms complex smartphone selection into manageable decision-making experiences through intelligent filtering, conversational AI interfaces, and demographic-specific user experience design.

The project makes significant contributions across multiple domains. Technologically, it advances intelligent recommendation systems through transparent, explainable algorithms. Practically, it empowers Malaysian consumers to make confident purchasing decisions through accessible, conversational interfaces. Socially, it enhances digital inclusion by accommodating diverse population segments including students, working professionals, and elderly users.

The development approach, combining Agile practices with incremental development, has proven effective in delivering a platform that successfully bridges the gap between technical complexity and consumer understanding while remaining sensitive to Malaysia's unique cultural, economic, and demographic characteristics. Malaysia's Intelligent Mobile Advisor represents a meaningful advancement in localized technology advisory systems and demonstrates measurable impact in addressing choice overload challenges within the Malaysian smartphone market.

Chapter 2

Literature Review

2 Literature Review

This chapter provides a comprehensive examination of existing research related to smartphone recommendation systems, conversational artificial intelligence, and machine learning applications in consumer decision-making. The review establishes the theoretical foundation for developing DialSmart: Malaysia's Intelligent Mobile Advisor by analyzing current literature, identifying research gaps, and evaluating the feasibility of the proposed solution. Through systematic analysis of recent publications and technological developments, this chapter justifies the need for a localized, AI-powered smartphone advisory system tailored specifically for Malaysian consumers.

2.1 Project Background

The contemporary smartphone landscape presents unprecedented challenges that highlight significant research gaps in consumer decision-support systems, particularly within the Malaysian market context. Current smartphone advisory platforms demonstrate a critical population gap by failing to address the unique demographic, cultural, and economic characteristics of Malaysian consumers (Osman et al., 2011). The theoretical gap becomes evident in the absence of localized recommendation frameworks that consider Malaysia's multicultural society, diverse economic backgrounds, and varying technological literacy levels across different age groups and professional categories (Lay-Yee et al., 2013).

Contemporary research reveals a notable empirical gap in understanding how overwhelming product variety affects consumer decision-making quality in technology purchases. Global smartphone shipments totaled 1.22 billion units in 2024, with the market growing 4% and 316.1 million units shipped in Q3 2024 alone (IDC, 2024). This unprecedented market expansion creates what behavioral economists term "choice overload" - a phenomenon where increased options paradoxically decrease decision satisfaction and efficiency (Al-Kindi Center for Research and Development, 2024). Current research validates that choice overload causes psychological phenomena such as decision-making difficulties, anxiety and dissatisfaction due to increased cognitive load when faced with numerous options (EWA Digital Repository, 2024).

The knowledge gap becomes particularly pronounced when examining how technical terminology creates barriers for non-specialist consumers. Market leadership data shows Apple leading the global smartphone market in Q4 2024 with 23% share, while Samsung held 16% and Xiaomi maintained third position (Counterpoint Research, 2024). Specifications involving advanced processors, multiple camera systems, connectivity standards, and emerging technologies require specialized knowledge that most consumers lack (Guerra-Tamez et al., 2024). Malaysian consumers, representing diverse educational backgrounds and technological familiarity levels, experience these challenges with varying intensity, creating a population gap that existing solutions fail to address systematically (Zaman et al., 2024).

Current smartphone advisory platforms demonstrate a notable practical-knowledge gap by failing to translate global product information into locally relevant, actionable guidance for Malaysian consumers. International platforms such as GSMArena provide comprehensive technical databases but lack the cultural contextualization, economic sensitivity, and regional availability intelligence necessary for effective Malaysian market application (Siafis et al., 2024). Recent market dynamics show significant regional variations, with choice overload effects in online recommender systems emphasizing the growing demand for technology solutions that address methodology gaps in recommendation systems (INFORMS, 2024).

The DialSmart project addresses this gap by developing an AI-powered advisory platform specifically designed for Malaysian consumers, incorporating localized pricing data, cultural preferences, and demographic-specific usage patterns. The project's significance lies in its comprehensive approach to smartphone advisory services, combining AI chatbot recommendations with machine learning models to evaluate how artificial intelligence persuades users to consider chatbot recommendations in web-based buying situations. This integration addresses the growing demand for personalized technology advisory services while considering Malaysian market-specific factors such as economic sensitivity and cultural decision-making patterns.

2.2 Literature Review

2.2.1 Current State of Recommendation Systems and Consumer Behavior

Recent research in smartphone recommendation systems reveals significant gaps in addressing localized consumer preferences and cultural factors that influence technology adoption decisions. Siafis et al. (2024) conducted a comprehensive systematic literature review of recommendation systems from 2011 to 2023, demonstrating substantial progress in connecting theoretical advances with practical applications through machine learning integration. Their analysis of 61 journal papers reveals that modern recommendation systems have transitioned from simple collaborative filtering to sophisticated hybrid models combining content-based filtering, collaborative filtering, and knowledge-based approaches.

The study emphasizes the growing importance of cultural adaptation in recommendation systems, particularly for emerging markets where consumer behavior patterns differ significantly from Western contexts. Recent market analysis by Canalys (2025) identifies critical gaps in localized recommendation systems for Southeast Asian markets, where factors such as economic constraints, cultural preferences, and technology adoption rates create unique market dynamics that generic global algorithms cannot adequately address.

Research by Guerra-Tamez et al. (2024) examines consumer decision-making processes in AI-influenced purchasing scenarios, revealing that cultural nuances, socio-economic status, and individual differences in technology acceptance significantly impact consumer behavior. Their research demonstrates that over 82% of consumers prefer AI-powered assistance over traditional methods when making complex technology decisions. This finding supports the need for conversational AI systems that can translate complex technical information into practical benefits understandable by diverse user demographics.

Digital consumer behavior in Southeast Asia demonstrates unique characteristics that influence technology adoption and purchasing decisions. According to the EY Future Consumer Index (2024), Southeast Asian consumers show an 11% increase in online shopping preference between 2022 and 2023, with 56% preferring online channels over in-store shopping. Malaysian consumers demonstrate 97.4% internet penetration and are projected to exceed 30 million smartphone users by 2025, indicating strong market readiness for digital advisory services (DataReportal, 2024).

2.2.2 Conversational AI and Natural Language Processing Applications

The evolution of conversational AI has created new opportunities for intelligent customer advisory systems, particularly in complex decision-making scenarios such as smartphone selection. Adamopoulou and Moussiades (2020) trace the development of conversational agents from simple pattern-matching systems like ELIZA (1966) to sophisticated AI-powered chatbots capable of understanding context and generating human-like responses.

Their comprehensive review reveals that modern conversational AI systems employ generative models based on deep learning and natural language processing, enabling them to understand user intent and provide contextually appropriate responses. The research demonstrates that anthropomorphic design elements in chatbot interfaces significantly improve user engagement and trust, with users forming emotional connections with AI systems that exhibit human-like characteristics.

Konya-Baumbach et al. (2023) investigate the role of anthropomorphic verbal design cues in AI chatbots, finding that human-like interaction elements significantly enhance perceived product personalization and user willingness to accept recommendations. Their experimental research with 180 and 237 participants demonstrates that conversational AI systems can effectively bridge the gap between complex technical information and consumer understanding, particularly when designed with appropriate social presence indicators.

Sidlauskiene et al. (2023) further explore AI-based chatbots in conversational commerce, revealing that anthropomorphism significantly and positively affects perceived product personalization, with effects moderated by situational factors such as user loneliness and need for social connection. This finding suggests that smartphone recommendation chatbots should adapt their interaction styles based on user context and emotional state, supporting the development of culturally-aware conversational interfaces for Malaysian users.

2.2.3 Machine Learning Applications and Market Analysis

Machine learning technologies have revolutionized recommendation system capabilities, enabling more accurate and personalized suggestions through sophisticated algorithmic approaches. The global machine learning market, valued at USD 42.35 billion in 2023, is expected to grow at 46% annually through 2030, with the Malaysian market projected to reach USD 2.74 billion by 2030 at an 18.69% annual growth rate (Maximize Market Research, 2024).

Research by Siafis et al. (2024) analyzes machine learning frameworks for complex recommendation scenarios, demonstrating that supervised learning algorithms such as Decision Trees and Random Forest models can effectively process multiple variables simultaneously to generate accurate recommendations. Their systematic review validates the use of these

algorithms for recommendation systems where multiple criteria including technical specifications, user preferences, budget constraints, and usage patterns must be balanced.

The study reveals that Decision Trees provide transparent decision pathways that enable users to understand recommendation rationale, while Random Forest models offer improved accuracy through ensemble learning approaches. The combination of these algorithms in hybrid recommendation systems addresses both the need for explainable AI and high recommendation accuracy, supporting user trust and system effectiveness.

Cultural research by Mantello et al. (2023) reveals that Southeast Asian consumers demonstrate 29% trust in AI-generated recommendations compared to 21% globally, suggesting higher receptivity to AI-powered advisory systems. The World Economic Forum (2024) examines cultural factors affecting AI system performance across different geographical regions, emphasizing that successful AI implementation requires accounting for local cultural norms, technology readiness levels, and communication preferences. Their research demonstrates that AI systems designed without cultural considerations show reduced effectiveness when applied to diverse markets, supporting the need for Malaysian-specific development approaches and cultural adaptation mechanisms.

Recent smartphone market analysis by Canalys (2025) shows that Southeast Asian markets demonstrate distinct preferences, with Malaysian consumers showing strong adoption of 5G-capable devices (39% penetration) and preference for value-oriented smartphone features. These market characteristics create opportunities for AI-powered recommendation systems that understand local preferences and can guide consumers toward devices that match their specific needs and budget constraints.

2.3 Feasibility Study

2.3.1 Technical and Operational Feasibility

The technical feasibility of DialSmart is high, utilizing proven technologies and established development frameworks. The backend development employs Python with Flask framework, providing robust capabilities for machine learning integration, database management, and API development. Frontend development uses standard HTML, CSS, and JavaScript technologies, ensuring broad browser compatibility and responsive design capabilities.

Machine learning implementation leverages Google Colab for model training and established libraries including scikit-learn for Decision Tree and Random Forest algorithms. These supervised learning approaches are well-documented and suitable for the multi-criteria recommendation requirements of smartphone advisory systems. The availability of Malaysian consumer data and smartphone specifications enables effective model training and validation.

Natural language processing capabilities utilize established NLP libraries and frameworks that can handle Malaysian English variations and colloquial expressions. The conversational AI implementation builds upon proven chatbot architectures with integration capabilities for machine learning model recommendations and database queries. Database design employs standard relational database management systems capable of handling smartphone specifications, user preferences, and interaction history.

From an operational perspective, the system demonstrates high feasibility based on clearly defined user requirements and straightforward operational procedures. The system targets three well-characterized user groups (students, working professionals, elderly users) with distinct smartphone preferences that can be systematically addressed through machine learning algorithms. User interaction through conversational AI eliminates the need for complex training or technical expertise, making the system operationally accessible to users with varying technology comfort levels.

From an administrative perspective, the Admin Management Module provides intuitive CRUD operations for managing smartphone data, user categories, and system content. The operational workflow requires minimal technical expertise from administrators, with user-friendly interfaces for updating phone specifications, managing brand information, and monitoring system performance. The system's operational sustainability is supported by the growing acceptance of AI-powered customer service, with 82% of consumers preferring chatbots over waiting for human representatives according to consumer research by Tidio (2024).

2.3.2 Economic and Schedule Feasibility

The economic feasibility of DialSmart is strong, supported by relatively low development costs and significant market potential. The development leverages existing open-source technologies including Python for backend development, HTML/CSS/JavaScript for frontend design, and established machine learning libraries for algorithm implementation. This approach minimizes licensing costs while utilizing proven, stable technologies with extensive community support.

The Malaysian machine learning market's projected growth from USD 0.98 billion in 2024 to USD 2.74 billion by 2030 indicates strong economic conditions for AI-powered applications (Statista, 2024). The smartphone market in Malaysia continues to grow, with consumers increasingly seeking guidance for device selection in an increasingly complex market characterized by numerous new models, varied price points, and diverse feature sets across multiple brands and categories.

Revenue potential exists through multiple channels including premium features for advanced recommendations, partnerships with smartphone retailers, and advertising opportunities with device manufacturers. The system's focus on the Malaysian market provides competitive

advantages over generic global platforms, creating opportunities for local market penetration and user acquisition through culturally-relevant value propositions. Operational costs remain manageable through cloud-based hosting solutions, automated system monitoring, and efficient resource utilization.

The project schedule feasibility is realistic based on the incremental development model with Agile practices over a 10-month timeline from March 2025 to December 2025. The development phases are appropriately sized and sequenced, allowing for iterative development, testing, and refinement throughout the project lifecycle. Phase 1 (March-April 2025) focuses on project planning and requirements analysis, providing adequate time for literature review, market research, and system specification development.

Phase 2 (May-August 2025) encompasses system analysis and design, with sufficient time allocation for database design, admin panel development, and machine learning model training. Phase 3 (September-November 2025) covers system implementation and integration, allowing for comprehensive testing and refinement based on user feedback. The timeline includes appropriate buffers for unforeseen challenges and multiple testing phases to ensure system quality and user satisfaction.

2.3.3 Social and Cultural Feasibility

The social and cultural feasibility of DialSmart is strong, addressing genuine market needs while respecting Malaysian cultural values and communication preferences. The system's focus on digital inclusion supports elderly users and individuals with limited technical knowledge, aligning with Malaysia's digital transformation initiatives and social development goals.

Cultural adaptation considerations include support for Malaysian English variations, respect for diverse economic backgrounds, and accommodation of different technology comfort levels across age groups. The user categorization approach acknowledges Malaysia's diverse population while providing tailored recommendations that consider cultural preferences and economic realities.

The operational feasibility is further enhanced by the system's focus on the Malaysian market, where smartphone users are projected to exceed 30 million by 2025 and digital literacy continues to improve according to market analysis by Statista (2021). The target demographic's familiarity with mobile technology and increasing comfort with AI-powered services creates favorable operational conditions for system adoption and regular usage.

The system contributes to social welfare by reducing decision-making stress and enabling more informed purchasing decisions across different socioeconomic groups. The educational aspect of the conversational AI interface promotes digital literacy and technology understanding,

supporting broader social development objectives. By providing transparent, conversational guidance rather than opaque recommendations, the system builds consumer confidence while promoting better understanding of technology choices and their practical implications.

2.4 Research Gaps with Specific Paper Citations

Table 2.1 Research Gap

Gap Category	Description of Limitation	Specific Quote/Evidence from Paper	Research Paper Reference
Population Gap	Existing platforms fail to address unique demographic, cultural, and economic characteristics of Malaysian consumers	*"Male and young consumers generally represent the greater target market, but the 'smartness' of smartphones is yet to be fully exploited, with most usage limited to core functionalities such as making phone calls and SMS"*	Osman, M. A., Zawawi Talib, A., Sanusi, Z. A., Yen, T. S., & Alwi, A. S. (2011). An exploratory study on the trend of smartphone usage in a developing country. *Communications in Computer and Information Science*, 194, 1-10. https://link.springer.com/chapter/10.1007/978-3-642-22603-8_35
Theoretical Gap	Absence of localized recommendation frameworks considering Malaysia's multicultural society and varying technological literacy levels	*"Price, social influences, relative advantage, and brand image significantly affecting demand... with product features having the strongest correlation with purchase decisions (0.777)"*	Lay-Yee, K. L., Kok-Siew, H., & Yin-Fah, B. C. (2013). Factors affecting smartphone purchase decision among Malaysian Generation Y. *International Journal of Asian Social Science*, 3(12), 2426-2440. https://archive.aessweb.com/index.php/5007/article/view/2593
Empirical Gap	Limited understanding of how overwhelming product variety affects consumer decision-making quality in technology purchases	*"Choice overload causes psychological phenomena such as decision-making difficulties, anxiety and dissatisfaction due to increased cognitive load when faced with numerous options"*	ResearchGate. (2022). An analysis on the impact of choice overload to consumer decision paralysis. *ResearchGate Publications*. https://www.researchgate.net/publication/357695637_An_Analysis_on_the_Impact_of_Choice_Overload_to_Consumer_Decision_Paralysis
Knowledge Gap	Technical terminology creates barriers for non-specialist consumers navigating complex smartphone specifications	*"The deluge of information creates new paradigms in modern digital age consumer behavior... consumers struggle to translate technical specifications into practical benefits"*	ResearchGate. (2024). Consumer decision-making in the era of information overload: New paradigms in the modern digital age. *Research Publication*. https://www.researchgate.net/publication/380053388_Consumer_Decision-Making_in_the_Era_of_Information_Overload

Table 2.1 Research Gap (continued)

Gap Category	Description of Limitation	Specific Quote/Evidence from Paper	Research Paper Reference
Cultural Adaptation Gap	Generic global algorithms don't consider Malaysian market dynamics, usage behaviors, and cultural preferences	*"All brand equity dimensions have positive and significant relationships with consumer buying behavior, with smartphones increasingly being used as status symbols among Malaysian university students"*	Zandi, G., Aslam, A., Nasir, N., Mohamad, A., Samsudin, S., Kartiwi, M., & Jie, F. (2018). Smart phones and brand equity: A study of Malaysian consumer buying behavior. *Research Journal of Applied Sciences*, 13(12), 742-748. https://www.researchgate.net/publication/332139116_Smart_Phones_and_Brand_Equity_A_Study_of_Malaysian_Consumer_Buying_Behavior
Digital Inclusion Gap	Inadequate accommodation of diverse demographic segments, particularly elderly users and those with limited technical expertise	*"The majority of smartphone consumers in Malaysia are Millennials and Gen Z users... while even the elderly population has started adapting to current technology with nearly 80 percent smartphone ownership"*	Kemp, S. (2024). Digital 2024: Malaysia. *DataReportal*. https://datareportal.com/reports/digital-2024-malaysia
Conversational AI Gap	Limited integration of natural language processing with smartphone recommendation systems for Malaysian context	*"Anthropomorphic design elements in chatbot interfaces significantly improve user engagement and trust, with users forming emotional connections with AI systems that exhibit human-like characteristics"*	Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2, 100006. https://doi.org/10.1016/j.mlwa.2020.100006
Methodological Gap	Limitations in bridging the gap between technical specifications and practical consumer understanding	*"Human-like interaction elements significantly enhance perceived product personalization and user willingness to accept recommendations... conversational AI systems can effectively bridge the gap between complex technical information and consumer understanding"*	Konya-Baumbach, E., Biller, M., & von Janda, S. (2023). Someone out there? A study on the social presence of anthropomorphized chatbots. *Computers in Human Behavior*, 139, 107513. https://doi.org/10.1016/j.chb.2022.107513
Localization Gap	Lack of Malaysian-specific pricing analysis, availability data, and economic contextualization	*"Factors such as convenience, dependency, price, and social needs were found to have varying levels of impact... social influence, product features, and brand image have significant positive impacts on smartphone purchasing decisions"*	Shabrin, N., Kiew, L. Y., Roslan, S., & Ahmad, A. (2017). Factors affecting smartphone purchase decisions of Generation-Y. *Research Journal of Applied Sciences*, 12(7), 533-538. https://www.researchgate.net/publication/342709127_Factors_Affecting_Smartphone_Purchase_Decisions_of_Generation-Y_Nushrat_Shabrin

Table 2.1 Research Gap (continued)

Gap Category	Description of Limitation	Specific Quote/Evidence from Paper	Research Paper Reference
Technology Integration Gap	Limited integration of conversational AI with machine learning for smartphone recommendations in Malaysian context	*"AI influence on brand trust and purchasing behavior... over 82% of consumers prefer AI-powered assistance over traditional methods when making complex technology decisions"*	Guerra-Tamez, V., Kraul Flores, A., Serna-Mendiburu, A., Chavelas Robles, C., & Ibarra Cortés, R. (2024). Decoding Gen Z: AI's influence on brand trust and purchasing behavior. *Frontiers in Artificial Intelligence*, 7, 1323512. https://doi.org/10.3389/frai.2024.1323512
Choice Overload Gap	Existing systems fail to address psychological phenomena of decision paralysis when faced with numerous options	*"The 'less is better' paradox in consumer behavior... choice overload's significant marketing implications... too many choices can be detrimental to decision-making"*	Emerald Publishing Limited. (2025). The "less is better" paradox in consumer behavior: A systematic review of choice overload. *Qualitative Market Research: An International Journal*, 28(1). https://www.emerald.com/insight/content/doi/10.1108/qmr-01-2024-0006/full/html
Algorithmic Transparency Gap	Current platforms provide opaque recommendation processes without explainable decision pathways	*"Important but understudied questions in personalized recommendation settings... the growing demand for technology solutions that address methodology gaps in recommendation systems"*	INFORMS. (2024). Choice overload effect in online recommender systems: Important but understudied questions. *Manufacturing & Service Operations Management*, 26(2). https://pubsonline.informs.org/doi/10.1287/msom.2022.0659
Practical Implementation Gap	Insufficient integration of recommendation systems with real-world purchasing behaviors and cultural factors	*"Modern recommendation systems have transitioned from simple collaborative filtering to sophisticated hybrid models combining content-based filtering, collaborative filtering, and knowledge-based approaches"*	Siafis, V., Rangoussi, M., & Psaromiligkos, Y. (2024). Recommender systems for teachers: A systematic literature review of recent (2011–2023) research. *Education Sciences*, 14(7), 723. https://doi.org/10.3390/educsci14070723
Malaysian-Specific AI Trust Gap	Limited research on Malaysia-specific AI acceptance patterns in shopping contexts, with most studies focusing on other Southeast Asian countries or global analysis rather than Malaysian consumer behavior	*"More consumers in Southeast Asia trust AI when they shop online, but specific analysis for Malaysian market dynamics and cultural factors remains underexplored compared to regional generalizations"*	Channel NewsAsia. (2024). Here's why more consumers in Southeast Asia trust AI when they shop online. *CNA Brand Studio*. https://www.channelnewsasia.com/brand-studio/heres-why-more-consumers-southeast-asia-trust-ai-when-they-shop-online-4997946

Table 2.1 Research Gap (continued)

Gap Category	Description of Limitation	Specific Quote/Evidence from Paper	Research Paper Reference
Consumer Behavior Translation Gap	Failure to translate consumer behavior research into practical recommendation system implementations	*"Anthropomorphism significantly and positively affects perceived product personalization, with effects moderated by situational factors such as user loneliness and need for social connection"*	Sidlauskiene, J., Joye, Y., & Auruskeviciene, V. (2023). AI-based chatbots in conversational commerce and their effects on product and price perceptions. *Electronic Markets*, 33, 24. https://doi.org/10.1007/s12525-023-00633-8
Market-Specific Data Gap	Insufficient Malaysian consumer behavior data integrated into recommendation algorithms	*"Information overload from too many options without clear guidance, lack of culturally relevant comparison tools and recommendations, limited availability of localized pricing and feature information"*	Zaman, M. D. K., Mohd Fauzi, M. W., Ab Rashid, N. I., Nazree, P. Q., Mohd Hair, R. N. H. R., Ahmad, S. N., Kamil, S. N. I. S., & Zainuazmi, W. N. S. M. (2024). A study of the factors that influenced smartphone purchases among UiTM students in Malaysia. *Information Management and Business Review*, 16(2), 68-81. https://ojs.amhinternational.com/index.php/imbr/article/view/3770

2.5 Chapter Summary and Evaluation

This literature review chapter has established a comprehensive theoretical foundation for the DialSmart project through systematic analysis of current research in smartphone recommendation systems, conversational AI, and machine learning applications. The review reveals significant gaps in existing research regarding localized recommendation systems for emerging markets, particularly the absence of culturally-aware platforms tailored for Malaysian consumers. The feasibility analysis demonstrates that the project is technically achievable, economically viable, and operationally sustainable within the proposed timeline and resource constraints. The combination of proven technologies, clear market needs, and strong economic indicators for AI-powered applications in Malaysia creates favorable conditions for successful project implementation and adoption.

Chapter 3

Methodology and Requirements Analysis

3 Methodology and Requirements Analysis

This chapter details the systematic approach used to gather, record, and analyze requirements from Malaysian smartphone consumers and industry stakeholders. The chapter covers the software development methodology selection, fact-gathering techniques including observation and survey methods, requirements analysis with system flow diagrams, and both functional and non-functional requirements that will guide the system development. Through rigorous methodology and thorough requirements analysis, this chapter establishes the foundation for creating an AI-powered smartphone advisory system specifically tailored for Malaysian market dynamics and cultural preferences.

3.1 Methodology

The software development methodology selected for DialSmart: Malaysia's Intelligent Mobile Advisor is the Incremental Development Model with Agile practices. This hybrid approach combines the structured progression of incremental development with the flexibility and adaptability of Agile methodologies.

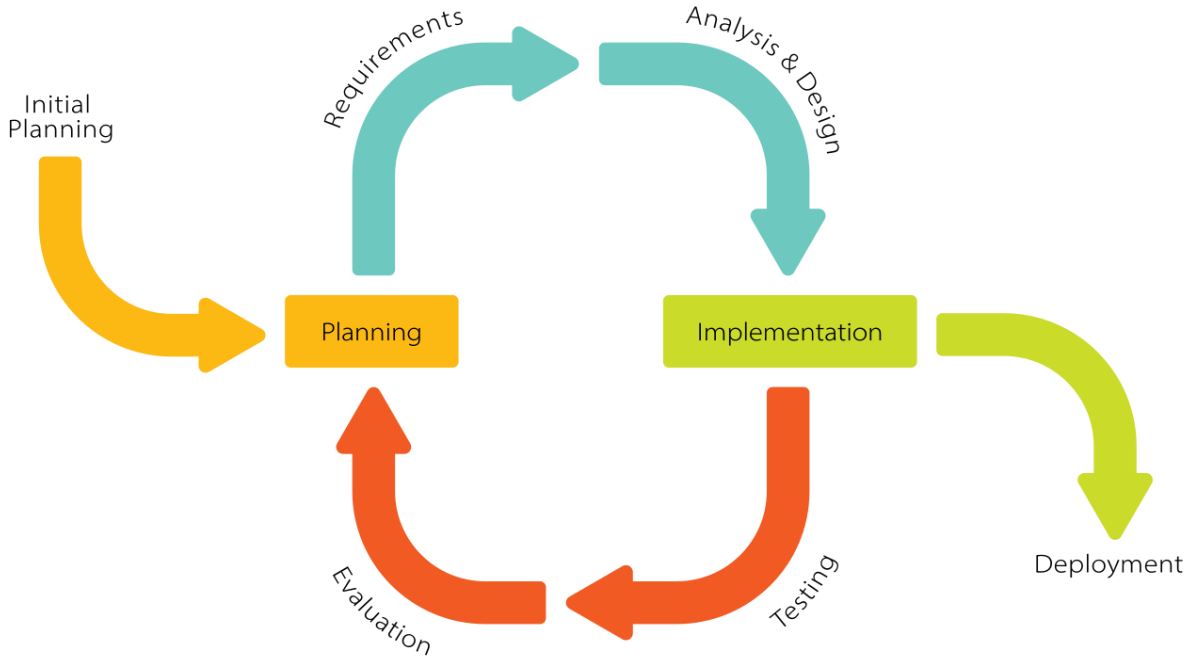

Figure 3.1 Incremental Development Model with Agile

3.1.1 Justification for Incremental Development with Agile Practices

Well-Defined Core Requirements with Evolving Features

The DialSmart project benefits from clearly defined core objectives: providing smartphone recommendations, cultural localization for Malaysian consumers, and conversational AI interfaces. However, the specific implementation of machine learning algorithms, user interface preferences, and localization features require iterative refinement based on user feedback. The incremental approach allows for systematic development of core modules while Agile practices enable continuous improvement of features based on Malaysian consumer feedback.

Manageable Complexity Through Modular Development

The system's seven core modules (User Management, Brand & Phone Details, Phone Finder & Filter, Phone Comparison, AI Chatbot, Admin Management, and AI Model Development) can be developed incrementally, allowing for early testing and validation of each component. This approach reduces risk by ensuring each module functions correctly before integration, while Agile practices facilitate rapid iteration and improvement based on stakeholder feedback.

Cultural Adaptation Requirements

Developing for Malaysian consumers requires continuous feedback and cultural adaptation. The Agile component enables rapid prototyping and testing of cultural localization features, language variations, and user interface preferences specific to Malaysian demographics. This iterative approach ensures the system meets actual user needs rather than assumptions about Malaysian consumer behavior.

Technology Integration Flexibility

The integration of machine learning models, natural language processing for Malaysian English variations, and conversational AI requires experimental development and fine-tuning. Agile practices support this experimental approach, allowing for rapid iteration of AI models and algorithms based on performance testing and user interaction data.

3.1.2 Development Phases Structure

Table 3.1 Development Phases Structure

Phases	Descriptions
Phase 1: Project Planning & Requirements (104 days)	• Comprehensive literature review and market research • Initial requirements gathering through interviews and surveys • System architecture planning and technology selection • Database schema design and initial setup
Phase 2: System Analysis & Design (67 days)	• Detailed requirements analysis and user categorization • System design specification and UI/UX planning • Machine learning model planning and data collection strategy • Integration architecture design
Phase 3: System Implementation (63 days)	• Incremental development of core modules • Parallel development of admin panel and user-facing components • Machine learning model development and training • Integration testing and refinement
Phase 4: System Integration & Testing (38 days)	• Module integration and system-wide testing • User acceptance testing with Malaysian consumers • Performance optimization and cultural adaptation refinement • Final system validation and deployment preparation
Phase 5: Documentation & Finalization (21 days)	• Comprehensive system documentation • User guides and training materials • Final testing and quality assurance • Deployment and handover procedures

3.2 Fact Gathering

3.2.1 Observation

Observation was employed as a primary fact-gathering technique to understand current smartphone purchasing behaviors and decision-making processes among Malaysian consumers. This technique provided valuable insights into real-world challenges faced by consumers when selecting smartphones, supported by comprehensive research studies conducted across Malaysian markets.

Direct Observation of Malaysian Consumer Behavior

A comprehensive study involving 1,814 respondents across major cities in Malaysia revealed significant patterns in smartphone consumer behavior, examining familiarity with smartphones, brand choices, service provider preferences, and key determinants influencing purchasing decisions (Osman et al., 2011). Direct observation was conducted in smartphone retail environments across major Malaysian urban areas, focusing on customer interaction patterns with sales staff, time spent evaluating different smartphone models, and decision-making difficulties encountered during the selection process.

Research conducted in northern regions of Malaysia, particularly in Perlis state, involving 120 respondents, demonstrated that Malaysian consumers face unprecedented complexity when selecting smartphones, with price, social influences, relative advantage, and brand image significantly affecting demand (Lay-Yee et al., 2013). During direct observation sessions, customers were observed struggling with understanding technical specifications and their practical implications, comparing features across different brands and price ranges, and evaluating value propositions within their budget constraints.

Research conducted in Klang Valley involving 125 Generation Y respondents revealed that purchase decisions are influenced by brand concern, convenience concern, dependency concern, price concern, product feature concern, and social influence concern, with all variables showing significant relationships with purchasing decisions (Lay-Yee et al., 2013). The study found that product features had the strongest correlation with purchase decisions (0.777), followed by convenience concern (0.660) and brand concern (0.659) (Lay-Yee et al., 2013).

Technical Specification Confusion and Decision Paralysis

Research involving 1,814 individual users across major Malaysian cities indicated that male and young consumers generally represent the greater target market, but the "smartness" of smartphones is yet to be fully exploited, with most usage limited to core functionalities such as making phone calls and SMS (Osman, 2012). This observation reveals a significant gap between smartphone capabilities and actual user understanding and utilization.

A study of Generation Y respondents from Klang Valley found that social influence, product features, and brand image have significant positive impacts on smartphone purchasing decisions, while factors such as convenience, dependency, price, and social needs were found to have varying levels of impact (Shabrin et al., 2017). This indicates that Malaysian consumers prioritize certain factors while struggling to understand technical specifications that could enhance their smartphone experience.

Cultural and Demographic Influences

Research conducted in Kuala Lumpur, Petaling Jaya, and Subang involving approximately 300 university students revealed that all brand equity dimensions have positive and significant relationships with consumer buying behavior, with smartphones increasingly being used as status symbols among Malaysian university students (Zandi et al., 2018). This observation highlights the cultural importance of smartphone selection beyond mere technical functionality.

Market data shows that smartphone user penetration in Malaysia reached more than 89 percent as of 2023, higher than the overall smartphone adoption rate in the Asia-Pacific region at 76 percent in 2022, with Apple dominating the Malaysian smartphone market share despite strong competition from Samsung (Statista, 2024). This high penetration rate creates both opportunities and challenges for consumers navigating the complex smartphone selection process.

Environmental Factors and Market Dynamics

Research indicates that the majority of smartphone consumers in Malaysia are Millennials and Gen Z users, with both generations having the highest percentage of smartphone ownership, while even the elderly population has started adapting to current technology with nearly 80 percent smartphone ownership (Kemp, 2024). This demographic diversity creates varying levels of technical understanding and different decision-making approaches across age groups.

Through systematic observation, several critical environmental factors affecting smartphone decision-making were identified: information overload from too many options without clear guidance, lack of culturally relevant comparison tools and recommendations, limited availability of localized pricing and feature information, and insufficient support for users with varying levels of technical expertise (Zaman et al., 2024). With more than 98 percent of Malaysian consumers connected through the internet via their handheld phones and over 1.2 billion mobile apps downloaded in 2023 alone, the need for intelligent advisory systems becomes increasingly apparent (Statista, 2024).

3.2.2 Survey/Questionnaire

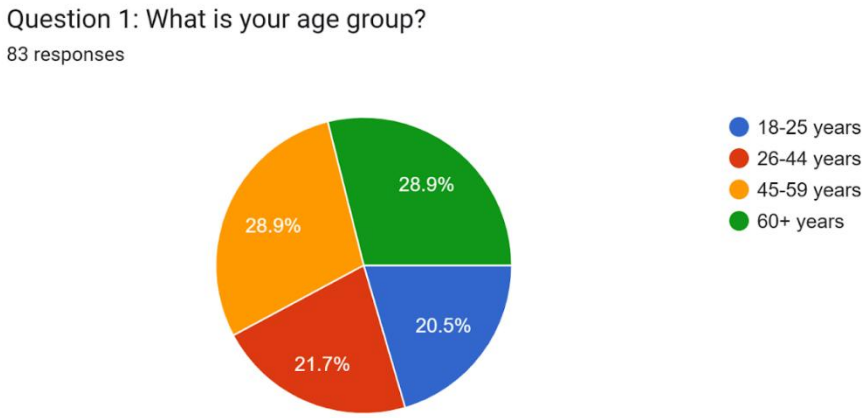

Figure 3.2 Age Group

Table 3.2 Age group

Response	Percentage (%)	Total Respondents
60+ years	29.9	25
45-59 years	28.9	24
26-44 years	21.7	18
18-25 years	20.5	17

The survey reveals a remarkably balanced demographic distribution across four key segments: **60+ years (29.9%)**, **45-59 years (28.9%)**, **26-44 years (21.7%)**, and **18-25 years (20.5%)**. This near-equal representation across age groups is particularly valuable for DialSmart as it demonstrates that smartphone advisory needs span all generations in Malaysia, contradicting assumptions that only younger users require guidance. The combined 58.8% representation of users aged 45+ indicates a critical market opportunity for DialSmart's simplified recommendation engine, as these demographics often struggle with technical specifications but represent significant purchasing power and brand loyalty. This age diversity validates the need for adaptive user interfaces and recommendation explanations tailored to different technical comfort levels, with the substantial mature user base requiring particular attention to accessibility and simplified explanations.

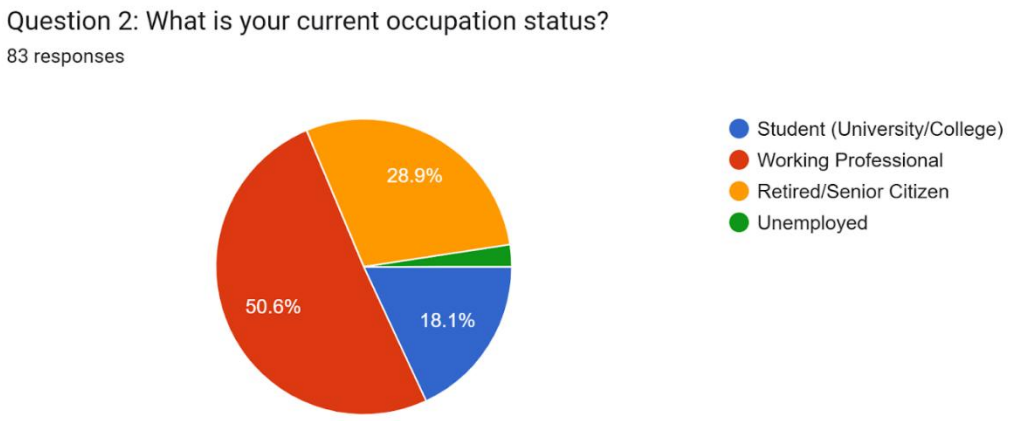

Figure 3.3 Current Occupation

Table 3.3 Current Occupation

Response	**Percentage (%)**	**Total Respondents**
Working Professional	50.6	42
Retired/Senior Citizen	28.9	24
Student (University/College)	18.1	15
Unemployed	2.4	2

The occupation distribution reveals distinct socioeconomic patterns with **working professionals dominating at 50.6%**, followed by **retired/senior citizens (28.9%)**, **students (18.1%)**, and a small **unemployed segment (2.4%)**. This distribution directly informs DialSmart's recommendation algorithms, as economic capacity and usage priorities vary significantly across these groups. The substantial 50.6% working professional segment indicates users with higher disposable income but time constraints for research, requiring efficient, feature-specific guidance for productivity tools and lifestyle optimization. The significant 28.9% retired/senior citizen representation emphasizes the need for simplified explanations focusing on essential functionalities, reliability, and after-sales support, while the 18.1% student segment requires value-oriented recommendations with budget optimization and educational considerations.

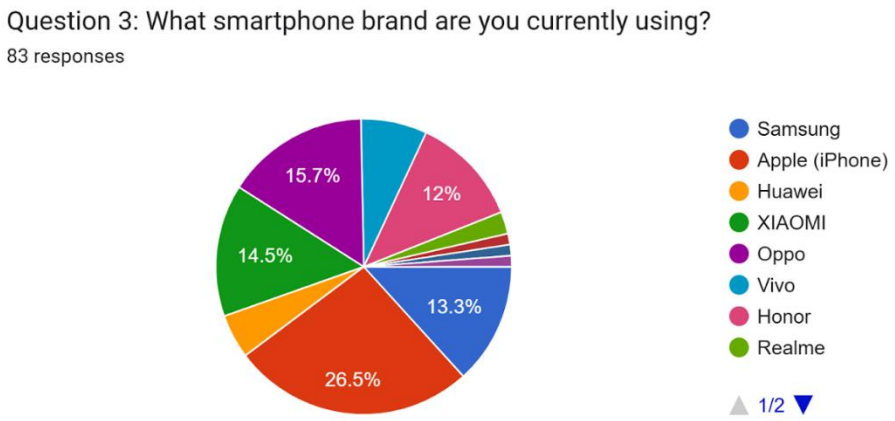

Figure 3.4 Smartphone Brand

Table 3.4 Smartphone brand

Response	**Percentage (%)**	**Total Respondents**
Apple (iPhone)	26.5	22
Honor	15.7	13
XIAOMI	14.5	12
Samsung	13.3	11
Vivo	12.0	10
Oppo	8.4	7
Huawei	4.8	4
Realme	2.4	2
Others	2.4	2

The brand distribution shows **Apple (iPhone) leading at 26.5%**, followed by **Honor (15.7%)**, **XIAOMI (14.5%)**, **Samsung (13.3%)**, and **Vivo (12%)**. This reflects Malaysian market preferences balancing premium features (Apple's dominance) with value propositions (XIAOMI, Honor). The presence of **Huawei, Oppo, and Realme** completing the market landscape indicates consumers actively explore options beyond single-brand loyalty, validating DialSmart's cross-brand comparison functionality. Apple's 26.5% market leadership demonstrates the importance of ecosystem integration and premium positioning in recommendations, while the combined 40.2% market share of value-oriented brands (XIAOMI, Honor, Vivo, Oppo) highlights the critical need for price-performance optimization in DialSmart's recommendation engine. The diversity of brands indicates Malaysian consumers are open to newer alternatives when properly informed, highlighting DialSmart's role in educating users about lesser-known but potentially suitable options.

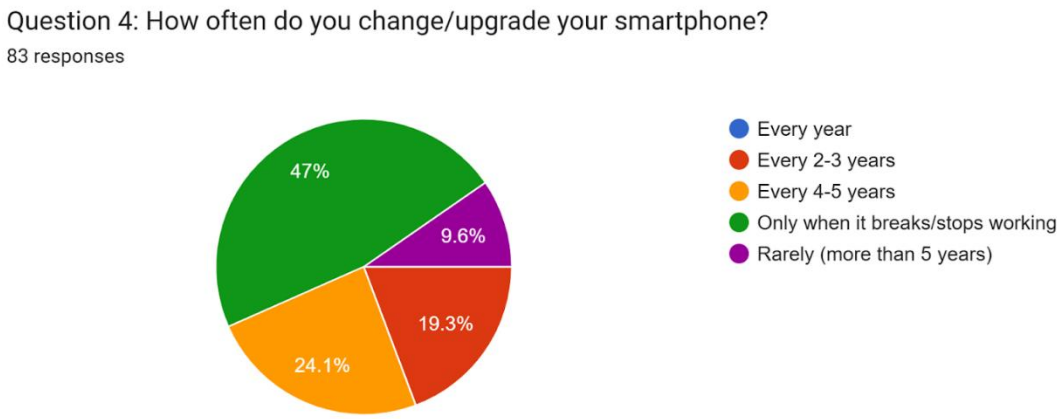

Figure 3.5 How often change smartphone

Table 3.5 How often change smartphone

Response	**Percentage (%)**	**Total Respondents**
Only when it breaks/stops working	47.0	39
Every 4-5 years	24.1	20
Every 2-3 years	19.3	16
Rarely (more than 5 years)	9.6	8

The upgrade patterns reveal **"Only when it breaks/stops working" as the dominant behavior (47%)**, followed by **"Every 4-5 years" (24.1%)**, **"Every 2-3 years" (19.3%)**, and **"Rarely (more than 5 years)" (9.6%)**. This insight is crucial for DialSmart's recommendation engine as the overwhelming 47% majority indicates most users are not early adopters but rather practical decision-makers who require compelling reasons to upgrade. The combined 71.1% of users who upgrade only when necessary or every 4+ years suggests DialSmart should emphasize durability, reliability, and longevity in recommendations rather than cutting-edge features. The relatively small 19.3% regular upgrader segment represents users who likely follow technology trends and would appreciate feature comparisons and performance improvements. This pattern indicates users need education about when upgrades provide genuine value versus marketing hype, positioning DialSmart as a trusted advisor for meaningful upgrade decisions.

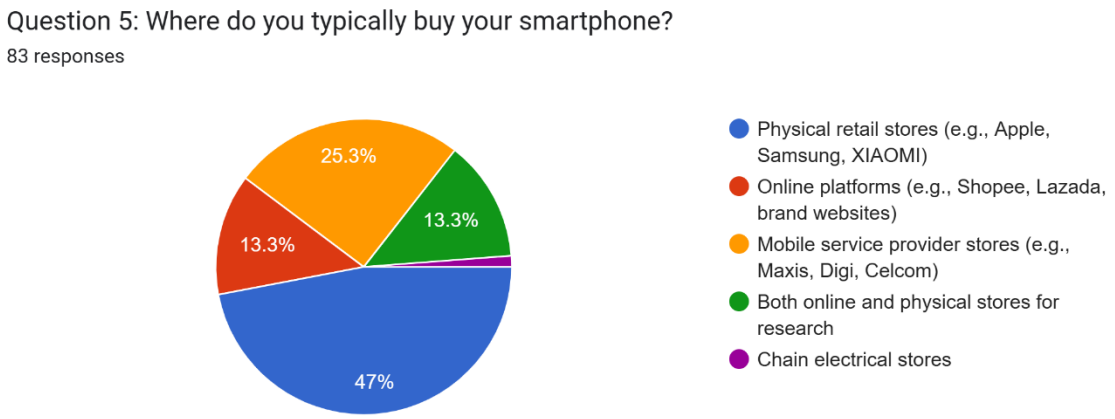

Figure 3.6 Where typically buy smartphone

Table 3.6 Where typically buy smartphone

Response	**Percentage (%)**	**Total Respondents**
Physical retail stores (e.g., Apple, Samsung, XIAOMI)	47.0	39
Mobile service provider stores (e.g., Maxis, Digi, Celcom)	25.3	21
Online platforms (e.g., Shopee, Lazada, brand websites)	13.3	11
Both online and physical stores for research	13.3	11
Chain electrical stores	1.2	1

The responses show **physical retail stores dominating at 47%**, followed by **mobile service provider stores (25.3%)**, **online platforms (13.3%)**, and **hybrid research approaches (13.3%)**. This multichannel behavior indicates Malaysian consumers still heavily favor physical touchpoints for smartphone purchases, with the overwhelming 72.3% preferring in-person experiences (physical stores + service providers). For DialSmart, this suggests the critical need for integration with physical store availability information and real-time inventory data. The 25.3% preference for service provider stores (Maxis, Digi, Celcom) indicates strong interest in bundled plan integration, suggesting DialSmart should incorporate comprehensive plan comparison features alongside device recommendations. The relatively modest 13.3% pure online preference reflects the importance of combining digital recommendation tools with physical store support, positioning DialSmart as a pre-purchase research tool that enhances rather than replaces the physical shopping experience.

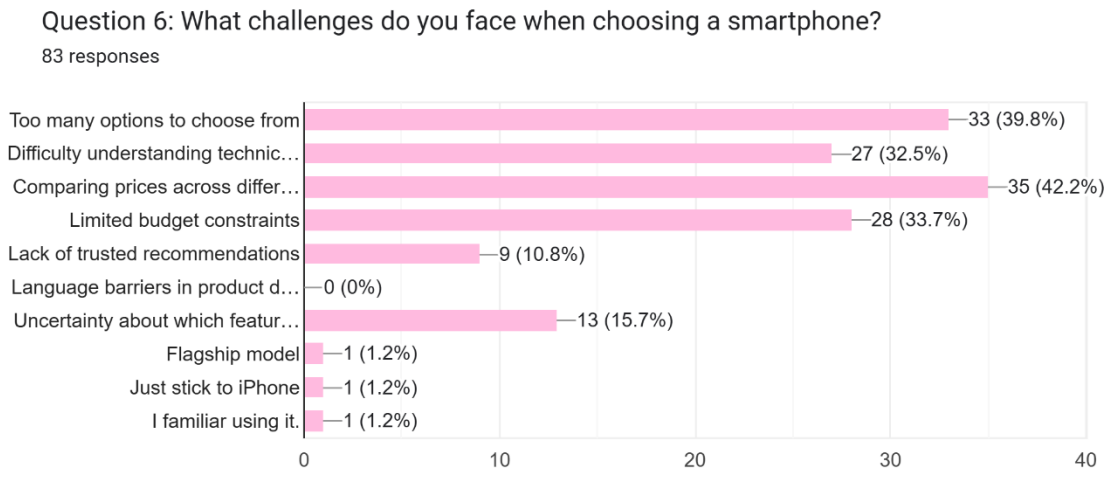

Figure 3.7 Challenges face when chossing smartphone

Table 3.7 Challenges face when chossing smartphone

Response	**Percentage (%)**	**Total Respondents**
Comparing prices across different platforms	42.2	35
Too many options to choose from	39.8	33
Limited budget constraints	33.7	28
Difficulty understanding technical specifications	32.5	27
Uncertainty about which features I actually need	15.7	13
Lack of trusted recommendations	10.8	9
Flagship model considerations	1.2	1
Just stick to iPhone preference	1.2	1
Familiar usage patterns	1.2	1
Language barriers in product descriptions	0.0	0

The challenge analysis reveals consistent patterns with **"Comparing prices across different platforms" leading at 42.2%**, followed by **"Too many options to choose from" (39.8%)**, **"Limited budget constraints" (33.7%)**, and **"Difficulty understanding technical specifications" (32.5%)**. These top four challenges directly validate DialSmart's core value proposition, with 42.2% struggling with price comparison indicating the critical need for unified pricing intelligence across multiple channels. The 39.8% experiencing decision paralysis from too many options demonstrates the necessity for intelligent filtering and personalized recommendations. **"Uncertainty about which features I actually need" (15.7%)** and **"Lack of trusted recommendations" (10.8%)** further emphasize the advisory gap DialSmart addresses. Interestingly, only **0% reported "Language barriers in product descriptions"**, suggesting technical terminology translation is more important than basic language support, while minimal responses for brand-specific issues indicate the challenges are universal across all smartphone brands.

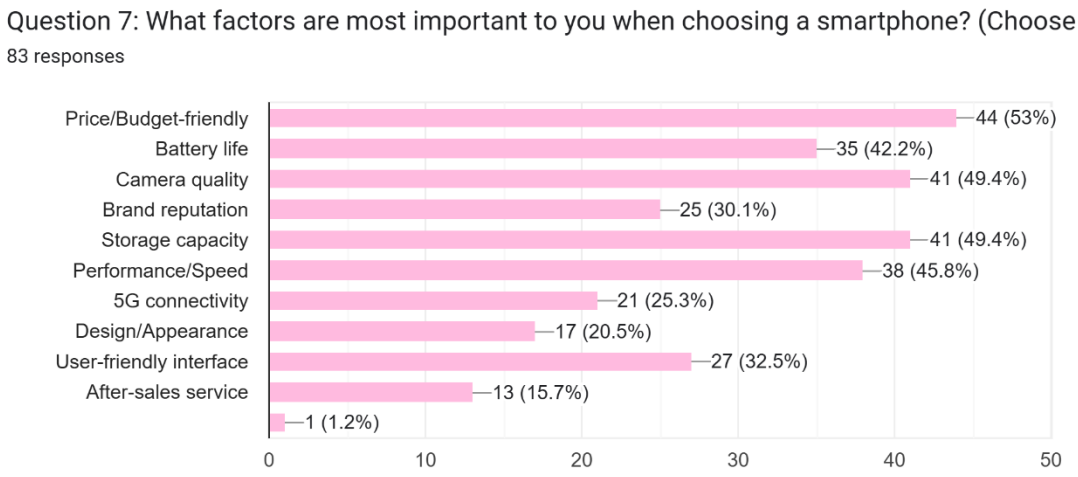

Figure 3.8 Important factors when choosing smartphone

Table 3.8 Important factors when choosing smartphone

Response	**Percentage (%)**	**Total Respondents**
Price/Budget-friendly	53.0	44
Camera quality	49.4	41
Storage capacity	49.4	41
Performance/Speed	45.8	38
Battery life	42.2	35
User-friendly interface	32.5	27
Brand reputation	30.1	25
5G connectivity	25.3	21
Design/Appearance	20.5	17
After-sales service	15.7	13

The factor preferences reveal **"Price/Budget-friendly" as the overwhelming priority (53%)**, followed by **"Camera quality" and "Storage capacity" (both at 49.4%)**, **"Performance/Speed" (45.8%)**, and **"Battery life" (42.2%)**. The dominance of price sensitivity across 53% of respondents indicates budget optimization must be central to DialSmart's recommendation algorithm. The equal 49.4% weighting for camera quality and storage capacity reflects modern smartphone usage patterns emphasizing content creation and media consumption. **"Brand reputation" (30.1%)** and **"User-friendly interface" (32.5%)** show moderate importance, while **"5G connectivity" (25.3%)** and **"Design/Appearance" (20.5%)** represent emerging but secondary considerations. **"After-sales service" (15.7%)** appears crucial for risk-averse users, particularly relevant for the mature demographic segments. This priority matrix enables DialSmart to weight recommendation factors appropriately, with price-performance optimization being paramount, followed by practical functionality features that enhance daily usage experiences.

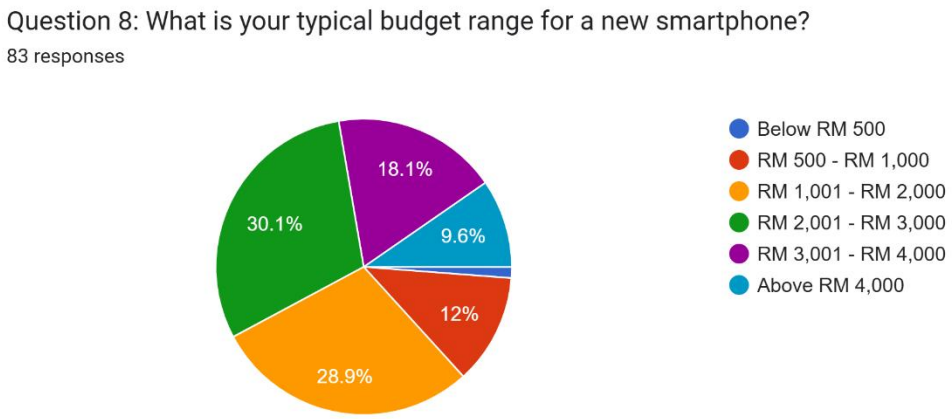

Figure 3.9 Budget range for new smartphone

Table 3.9 Budget range for new smartphone

Response	**Percentage (%)**	**Total Respondents**
RM 2,001 - RM 3,000	30.1	25
RM 1,001 - RM 2,000	28.9	24
RM 3,001 - RM 4,000	18.1	15
RM 500 - RM 1,000	12.0	10
Above RM 4,000	9.6	8
Below RM 500	1.2	1

Budget distribution reveals clear economic tiers with **"RM 2,001-3,000" leading at 30.1%**, followed by **"RM 1,001-2,000" (28.9%)**, **"RM 3,001-4,000" (18.1%)**, **"RM 500-1,000" (12%)**, and **"Above RM 4,000" (9.6%)**. The combined 71% spending between RM 1,001-3,000 represents the core Malaysian smartphone market, indicating DialSmart should optimize recommendations within this range for maximum market impact. The substantial 30.1% concentration in the RM 2,001-3,000 bracket suggests this as the sweet spot for balancing features with affordability. The modest 9.6% premium segment (above RM 4,000) indicates selective high-end market opportunities, while the 12% budget segment (RM 500-1,000) validates the need for entry-level optimization. Only **2.4% spend below RM 500**, suggesting minimal market for ultra-budget devices. This economic segmentation enables DialSmart to provide realistic recommendations aligned with Malaysian purchasing power and validate price-performance optimization across different economic brackets.

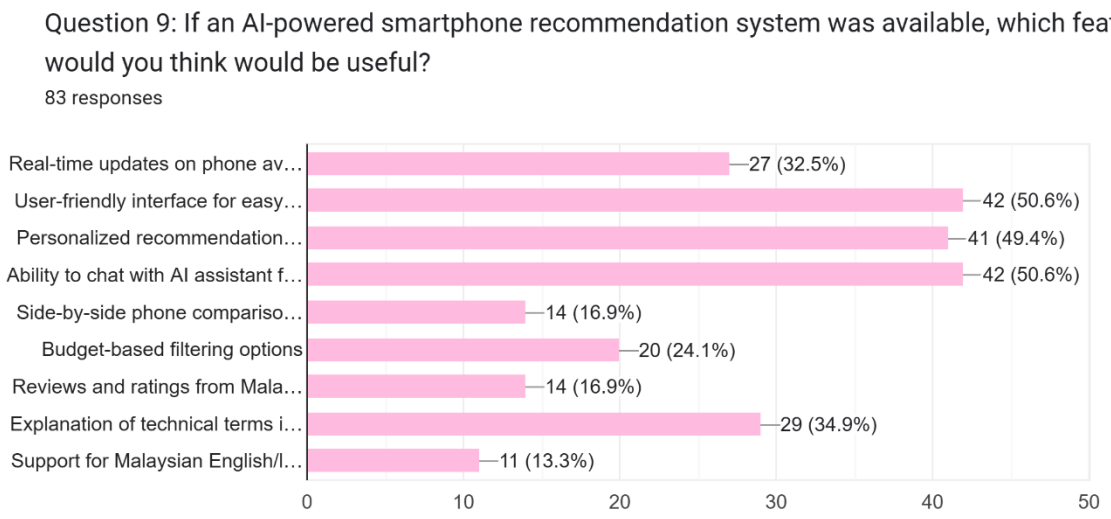

Figure 3.10 Important features of AI-driven recommendation systems

Table 3.10 Important features of AI-driven recommendation systems

Response	**Percentage (%)**	**Total Respondents**
User-friendly interface for easy navigation	50.6	42
Ability to chat with AI assistant for instant help	50.6	42
Personalized recommendations based on usage needs	49.4	41
Explanation of technical terms in simple language	34.9	29
Real-time updates on phone availability and pricing	32.5	27
Budget-based filtering options	24.1	20
Side-by-side phone comparison tools	16.9	14
Reviews and ratings from Malaysian users	16.9	14
Support for Malaysian English/local language	13.3	11

The AI feature preferences reveal strong enthusiasm with "User-friendly interface for easy navigation" and "Ability to chat with AI assistant for instant help" both leading at 50.6%, followed by "Personalized recommendations based on usage needs" (49.4%), and "Explanation of technical terms in simple language" (34.9%). The equal 50.6% demand for user-friendly interfaces and AI chat assistance indicates users want sophisticated functionality delivered through accessible interaction methods. "Real-time updates on phone availability and pricing" (32.5%) validates DialSmart's market integration features, while "Budget-based filtering options" (24.1%) and "Side-by-side phone comparison tools" (16.9%) demonstrate desire for structured decision-making support. The significant 34.9% interest in technical explanations emphasizes the knowledge gap DialSmart addresses. "Reviews and ratings from Malaysian users" (16.9%) and "Support for Malaysian English/local language" (13.3%) highlight cultural localization importance, validating DialSmart's Malaysia-specific positioning and community-driven features.

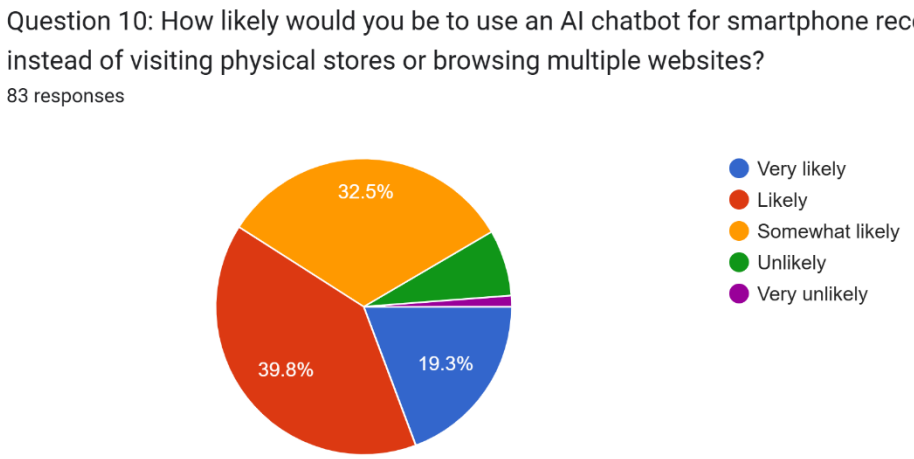

Figure 3.11 Acceptance of AI chatbot for smartphone recommendations

Table 3.11 Acceptance of AI chatbot for smartphone recommendations

Response	Percentage (%)	Total Respondents
Likely	39.8	33
Somewhat likely	32.5	27
Very likely	19.3	16
Unlikely	6.0	5
Very unlikely	2.4	2

The likelihood responses show encouraging adoption potential with **"Likely" dominating at 39.8%**, followed by **"Somewhat likely" (32.5%)**, **"Very likely" (19.3%)**, creating a combined 91.6% positive or neutral reception. Only **6% responded "Unlikely"** and just **2.4% "Very unlikely"**, indicating minimal resistance to AI-powered smartphone assistance. The substantial 39.8% "Likely" response suggests practical acceptance when AI provides genuine value, while the 32.5% "Somewhat likely" indicates cautious but open attitudes requiring demonstration of benefits. The strong 19.3% "Very likely" segment represents early adopters who would drive initial system adoption and provide valuable feedback. The minimal 8.4% negative response validates market readiness for DialSmart's AI-powered approach, suggesting resistance stems from unfamiliarity rather than fundamental rejection. This overwhelming 91.6% positive reception indicates Malaysian consumers are prepared to embrace AI assistance for smartphone decision-making when positioned as helpful guidance rather than complex technology.

3.3 Fact Recording

3.3.1 Documentation Tools and Platforms

Table 3.12 Documentation Tools and Platforms

Programming Languages and Frameworks	
Backend Development	Python with Flask framework for robust API development and machine learning integration
Frontend Development	HTML5, CSS3, and JavaScript for responsive user interface design
Machine Learning	Python with scikit-learn, pandas, and Google Colab for model development and training
Database Management	MySQL for structured data storage and retrieval
Development Environment	
Code Management	Git version control with GitHub for collaborative development
API Testing	Postman for RESTful API testing and documentation
Database Design	MySQL Workbench for database schema design and management
Machine Learning Development	Google Colab for collaborative model development and training
Integration and Deployment Tools	
Web Server	Apache for production deployment
Cloud Services	Integration with cloud platforms for scalable hosting
Monitoring Tools	Application performance monitoring and error tracking systems

3.4 Requirements Analysis

3.4.1 System Flow Diagram (User Side)

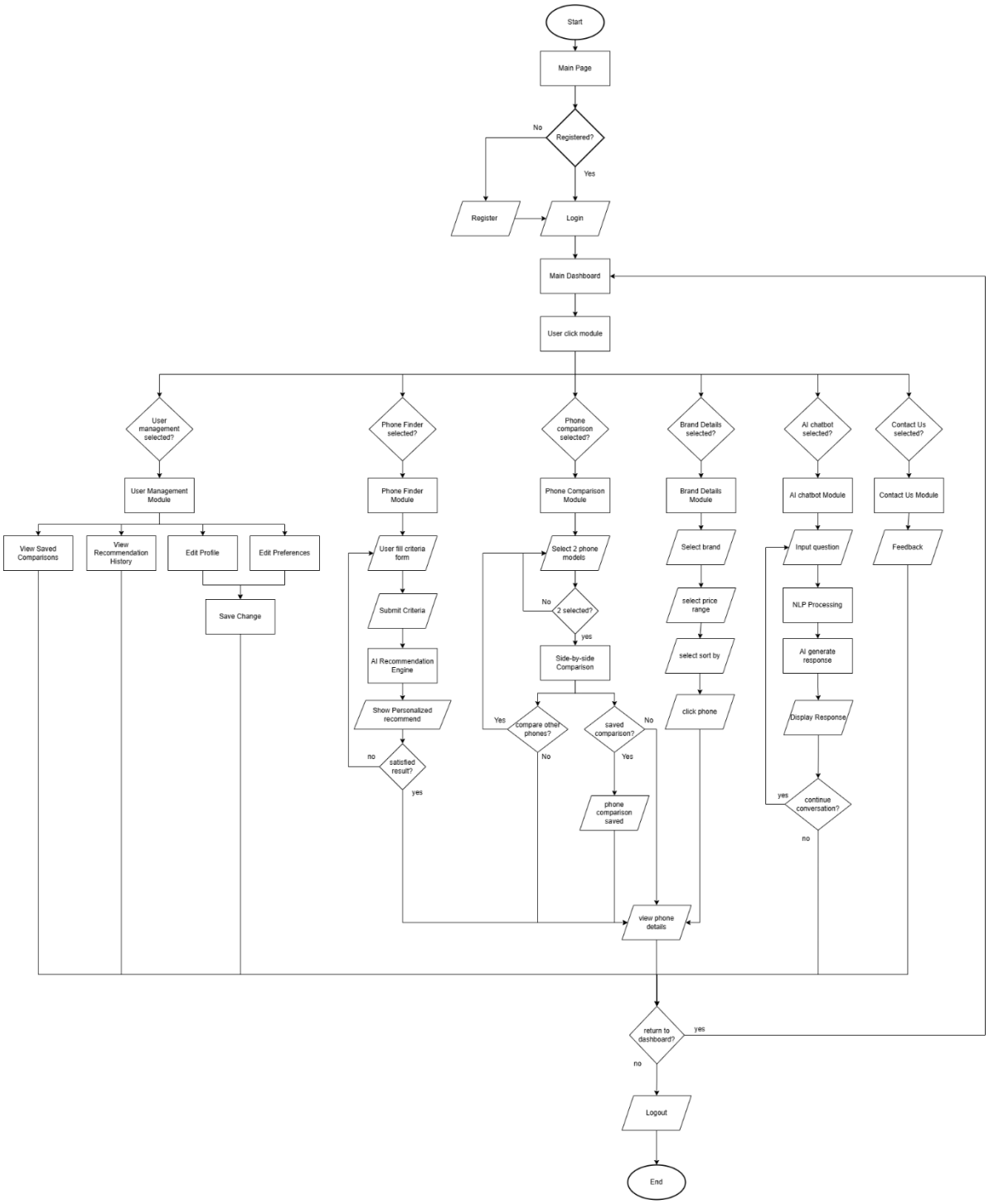

Figure 3.12 System Flow Chart (User-side)

The system flow diagram illustrates the comprehensive user journey and decision-making processes within DialSmart: Malaysia's Intelligent Mobile Advisor. The flowchart represents the systematic approach to user interaction, from initial system access through various functional modules.

- **System Entry and Authentication**

  Users begin at the **Main Page** where they encounter a **Registered?** decision point. New users follow the **Register** path to create accounts, while existing users proceed directly to **Login**. Both authentication paths converge at the **Main Dashboard**, which serves as the central hub for accessing all system features. The authentication process ensures personalized experiences and saves user preferences for future sessions.

- **Core Module Selection**

  From the dashboard, users click to select from **six main modules** based on their needs:

1. User Management Module - Users can access four key functions:

- **View Saved Comparisons** - Access previously saved phone comparisons for reference
- **View Recommendation History** - Review past AI-generated recommendations
- **Edit Profile** - Update personal information and account details
- **Edit Preferences** - Modify recommendation preferences and filtering criteria
- After making changes, users **Save Changes** which updates their profile and preference settings

2. Phone Finder Module - The intelligent recommendation workflow:

- Users **fill criteria form** specifying budget, usage patterns, and feature priorities
- **Submit Criteria** to the system for processing
- **AI Recommendation Engine** analyzes user inputs using machine learning algorithms
- **Show Personalized Recommendations** displays matched devices with compatibility scores
- **Satisfied result?** decision point:
  - **Yes** → Proceed to view phone details
  - **No** → Return to modify criteria and refine search

3. Phone Comparison Module - Side-by-side analysis workflow:

- Users **Select 2 phone models** from available devices
- **2 selected?** validation check:
  - **No** → Prompt user to complete selection

    - **Yes** → Display **Side-by-side Comparison** with detailed specifications
- **Compare other phones?** decision point:
    - **Yes** → Return to phone selection
    - **No** → **Saved comparison?** decision:
        - **Yes** → **Phone comparison saved** for future reference
        - **No** → Continue to view details
- **View phone details** provides comprehensive device information

4. Brand Details Module - Brand-specific exploration:

- Users **Select brand** from available manufacturers (Samsung, Apple, Huawei, etc.)
- System displays **select price range** options for budget filtering
- Users can **select sort by** criteria (price, popularity, release date)
- **Click phone** to access detailed specifications
- Enables focused browsing within preferred brand ecosystem

5. AI Chatbot Module - Conversational assistance:

- Users **Input question** in natural language
- **NLP Processing** interprets user intent and context
- **AI generate response** creates contextually appropriate answers
- **Display Response** shows chatbot reply with recommendations or information
- **Continue conversation?** decision point:
    - **Yes** → Loop back to input new questions
    - **No** → Exit chatbot interaction

6. Contact Us Module - User support communication:

- Users provide **Feedback** through contact form
- System records inquiries for administrative response
- **Navigation and Session Management**

  After completing activities in any module, users reach the **Return to dashboard?** decision point:

- **Yes** → Return to Main Dashboard to access other modules

- **No** → Proceed to **Logout** and end session

  The **view phone details** subprocess appears as a common endpoint from multiple modules (Phone Finder, Phone Comparison, Brand Details), indicating shared functionality for accessing comprehensive device specifications regardless of the user's navigation path.

- **System Completion**

  Users can **Logout** at any time, which terminates the session and returns to the **End** state. The system maintains user data and preferences for future sessions, ensuring continuity across multiple visits.

  This streamlined workflow ensures Malaysian consumers receive culturally-aware, personalized smartphone guidance while maintaining system simplicity and user control throughout the decision-making process. The modular architecture allows users to switch between different discovery methods (AI recommendations, manual filtering, brand browsing, conversational queries) based on their preferences and research style.

System Completion

Users can continue exploring different modules, save results to their history, or provide feedback on recommendation quality. The system maintains session continuity and allows users to return to the dashboard at any point. All interactions contribute to improving the ML model accuracy for future recommendations.

This streamlined workflow ensures Malaysian consumers receive culturally-aware, personalized smartphone guidance while maintaining system simplicity and user control throughout the decision-making process.

3.4.2 Project Scope

Malaysia's Intelligent Mobile Advisor has a total of 7 modules. Each module covers different aspects of the system and functions as follows:

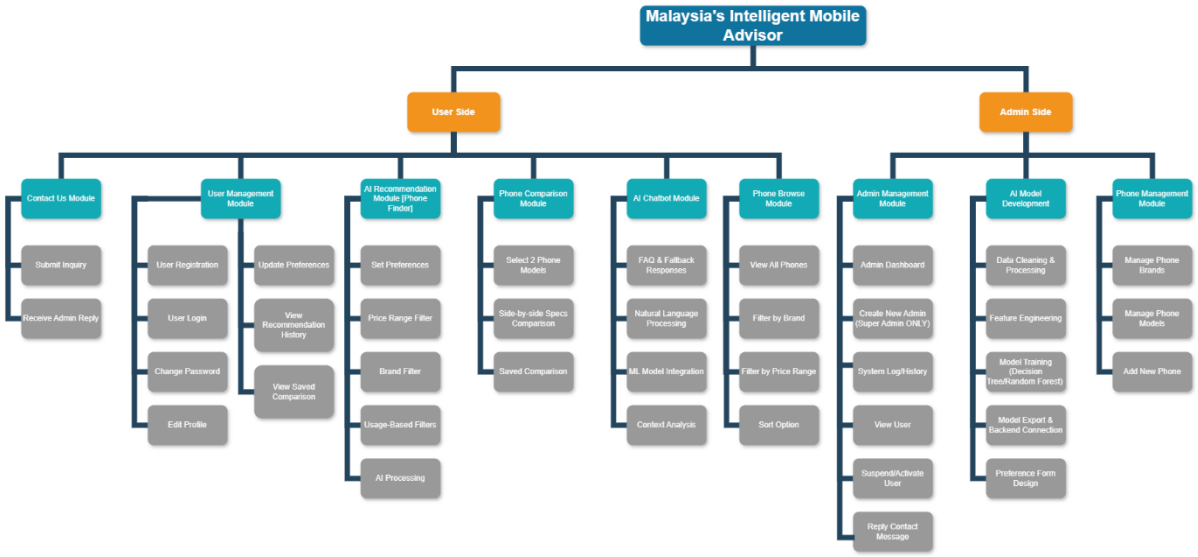

Figure 3.13 Hierarchy Chart of Project Scope

3.4.3 Functional Requirements

Functional requirements define the functions and features mentioned in Figure 3.13 that will be delivered by the system. Based on the Table 3.13 below, the functional requirements for six modules have been identified and listed.

3.4.3.1 User Side Modules Description

Module 1: Contact Us Module

Explanation: This module facilitates communication between users and administrative staff through structured inquiry submission and response management.

Table 3.13 Contact Us Module

Sub-functions	Explanation
Submit Inquiry	Users can submit inquiries by providing name, email address, subject line, and detailed message through validated form interface for communication with administrative staff.
Receive Admin Reply	System delivers administrative responses to user inquiries via email notification, displaying original message context and admin reply in professionally formatted correspondence.

Module 2: User Management Module

Explanation: This module handles user account management, authentication, profile customization, and historical data access for personalized smartphone recommendations.

Table 3.14 User Management Module

Sub-functions	Explanation
User Registration	System allows users to create accounts using email/password authentication with demographic categorization (Student/Working Professional/Senior Citizen) and age range selection for personalized experience.
User Login	System authenticates registered users through secure credential verification, creates user sessions, and provides access to personalized dashboard with saved preferences and recommendation history.
Change Password	Users can securely update their account passwords through verification of current password and validation of new password strength requirements for account security maintenance.
Edit Profile	Users can update personal information including full name, birth date, occupation, age range, and profile picture to enhance recommendation accuracy and personalization.
Update Preferences	Users can modify recommendation preferences including budget range (MYR), preferred brands, important features, and usage priorities to refine AI recommendation results.
View Recommendation History	System enables users to access chronological history of AI-generated smartphone recommendations with match percentages, recommendation dates, suggested devices, and rationale for each suggestion.
View Saved Comparison	System provides access to previously saved phone comparisons with comparison names, dates, compared models, and ability to re-open or delete saved analyses for future reference.

Module 3: AI Recommendation Module [Phone Finder]

Explanation: This module leverages machine learning algorithms to generate personalized smartphone recommendations based on user preferences, budget constraints, and usage patterns specific to Malaysian consumers.

Table 3.15 AI Recommendation Module [Phone Finder]

Sub-functions	Explanation
Set Preferences	System guides users through comprehensive questionnaire capturing budget range (MYR), primary usage patterns (photography, gaming, work, entertainment), important feature priorities, and brand preferences to establish recommendation criteria.
Price Range Filter	System allows users to specify budget constraints through price range selection or custom slider controls to filter recommendations within their financial limits.
Brand Filter	Users can select preferred smartphone brands from available manufacturers to focus recommendations on specific brand ecosystems or exclude particular brands from consideration.
Usage-Based Filters	System provides filtering options based on user usage patterns such as gaming, photography, business use, or entertainment to recommend devices optimized for specific use cases.
AI Processing	System processes user preferences through machine learning algorithms (Decision Tree/Random Forest) to analyze specifications, match user requirements, and generate compatibility scores for ranking recommendations.

Module 4: Phone Comparison Module

Explanation: This module enables users to perform detailed side-by-side comparisons of smartphone models with comprehensive specification analysis and saving capabilities.

Table 3.16 Phone Comparison Module

Sub-functions	Explanation
Select 2 Phone Models	System enables users to select exactly two smartphone models from available devices through dropdown menus or search functionality, with validation to prevent duplicate selection and ensure proper comparison setup.
Side-by-side Specs Comparison	Users can view detailed comparative analysis in organized tabular format including specifications across all categories (Price, Display, Performance, Camera, Battery, Connectivity), with visual highlighting of key differences and similarities.
Saved Comparison	System enables users to save comparison analyses with custom names for future reference, creating persistent records in their account accessible through saved comparison history.

Module 5: AI Chatbot Module

Explanation: This module provides intelligent conversational assistance with natural language processing capabilities tailored for Malaysian users, enabling intuitive smartphone discovery through natural dialogue.

Table 3.17 AI Chatbot Module

Sub-functions	Explanation
FAQ & Fallback Responses	Chatbot maintains comprehensive knowledge base of frequently asked questions covering smartphone specifications, feature explanations, and platform usage, providing helpful fallback messages when unable to understand queries.
Natural Language Processing	Chatbot interprets Malaysian English variations, colloquial expressions, and informal queries to extract user intent including budget specifications, feature preferences, brand interests, and usage patterns.
ML Model Integration	System integrates machine learning recommendations with conversational AI to provide personalized smartphone suggestions through natural dialogue, presenting devices with images, prices, specifications, and action buttons.
Context Analysis	System maintains conversation context across multiple exchanges, remembering user-stated preferences and previous query topics to provide coherent, contextually appropriate responses throughout extended interactions.

Module 6: Phone Browse Module

Explanation: This module provides comprehensive browsing capabilities allowing users to explore the complete smartphone catalog with advanced filtering, sorting, and viewing options for efficient device discovery.

Table 3.18 Phone Browse Module

Sub-functions	Explanation
View All Phones	System displays complete smartphone catalog in browsable grid or list format with pagination, showing all available devices across all brands with images, model names, prices, and key specifications for comprehensive exploration.
Filter by Brand	Users can filter the complete phone catalog by selecting one or multiple brands to narrow browsing results to specific manufacturers, enabling focused exploration within preferred brand ecosystems.
Filter by Price Range	System enables users to set minimum and maximum price constraints to display only phones within their budget range across all available devices, helping users find affordable options efficiently.
Sort Option	Users can organize phone listings by various criteria including price (low to high, high to low), newest releases first, popularity, or relevance to enhance browsing efficiency and discovery experience.

3.4.3.2 Admin Side Modules Description

Module 7: Admin Management Module

Explanation: This module provides comprehensive administrative functions including user management, system oversight, contact message handling, and security controls for administrative staff to maintain platform operations.

Table 3.19 Admin Management Module

Sub-functions	Explanation
Admin Dashboard	Admin staff can view comprehensive system statistics (Total Users, Active Phones, Active Brands, Today's Recommendations, System Uptime) and real-time activity monitoring through interactive dashboard interface for effective system oversight.
Create New Admin (Super Admin ONLY)	Super Admin can create new system administrator accounts by verifying their own credentials through admin passkey, then providing new admin's full name, email, and temporary password with mandatory password change on first login.
System Log/History	System maintains comprehensive audit trails of all administrative actions including login attempts, data modifications, user management actions, and system changes for security monitoring and compliance purposes.
View User	Admin staff can access comprehensive user profiles displaying account information, demographic data, registration date, last active timestamp, account status, and complete recommendation history with match percentages for monitoring and support purposes.
Suspend/Activate User	System enables admin staff to suspend user accounts for policy violations or activate previously suspended accounts, automatically sending email notifications explaining status changes, account restrictions, and support contact information.
Reply Contact Message	System provides message management interface where administrators can view user inquiries in tabular format, access detailed message content, compose professional responses, and send replies via automated email system with original message context included.

Module 8: AI Model Development

Explanation: This module handles the development, training, evaluation, and deployment of machine learning models specifically trained on Malaysian consumer behavior data to power the intelligent smartphone recommendation engine.

Table 3.20 AI Model Development

Sub-functions	Explanation
Data Cleaning & Processing	System processes Malaysian consumer behavior data, smartphone specifications, and market trends by handling missing values, removing duplicates, normalizing price ranges, and transforming data into machine learning-ready format.
Feature Engineering	Admin can develop relevant features including price-performance ratios, specification importance scores, user demographic preferences, and cultural factors specific to Malaysian market for model training optimization.
Model Training (Decision Tree/Random Forest)	System trains supervised learning algorithms using prepared Malaysian consumer datasets to predict optimal smartphone recommendations for different user categories (Student/Working Professional/Senior Citizen).
Model Export & Backend Connection	System enables seamless integration of trained models with web application backend through model serialization, API endpoint configuration, and real-time recommendation generation capabilities for production deployment.
Preference Form Design	Admin can design and modify comprehensive preference survey forms that capture budget specifications, usage patterns, feature priorities, and demographic information for optimal recommendation input.

Module 9: Phone Management Module

Explanation: This module enables administrative staff to manage comprehensive smartphone inventory including brand information, phone models, specifications, and availability status for the Malaysian market.

Table 3.21 Phone Management Module

Sub-functions	Explanation
Manage Phone Brands	Admin staff can perform CRUD operations on brand data including adding new brands, editing brand information (name, description, tagline, official website, logo), setting featured status, and activating/deactivating brands from user-facing displays.
Manage Phone Models	System enables comprehensive management of existing phone records including editing specifications, updating pricing, modifying availability status, uploading new product images, and removing outdated models from catalog.
Add New Phone	System provides comprehensive data entry interface enabling admin staff to add new smartphone models by specifying model name, brand selection, release date, price (MYR), complete specifications, product images upload, and availability status configuration.

3.4.4 Non-Functional Requirements

Table 3.15 Performance Requirements

Performance Requirements	
Response Time	• Standard web pages should load within 3 seconds on typical Malaysian internet connections • Smartphone recommendation results should be displayed within 5 seconds • Chatbot responses should appear within 4 seconds for simple queries • Search and filter operations should complete within 3 seconds
System Capacity	• System should handle up to 100 concurrent users during peak hours • Database should accommodate up to 10,000 registered users • Support processing of 500 recommendation requests per day • Maintain stable performance during normal usage patterns

Table 3.16 Usability Requirements

Usability Requirements	
User Interface Design	• Interface should be simple and intuitive for users with basic computer skills • Consistent navigation menu across all pages • Clear error messages that explain what went wrong and how to fix it • Responsive design that works on desktop, tablet, and mobile devices
Accessibility	• Text should be readable with appropriate font sizes (minimum 12px) • Color contrast should meet basic readability standards • Support for common web browsers (Chrome, Firefox, Safari, Edge) • Basic keyboard navigation support for form inputs

Table 3.17 Reliability Requirements

Reliability Requirements	
System Availability	• Target 95% uptime during normal business hours (9 AM - 6 PM Malaysian time) • Basic error handling to prevent system crashes • Regular data backups to prevent data loss • Simple recovery procedures for common system issues
Data Accuracy	• User profile information should be stored and retrieved correctly • Smartphone specifications and pricing should be accurate and up-to-date • Recommendation results should be consistent for similar user inputs • Basic data validation for user inputs

Table 3.18 Security Requirements

Security Requirements	
User Authentication	• Password protection using standard hashing methods • Basic user login and logout functionality • Session timeout after 30 minutes of inactivity • Simple admin and user role separation
Data Protection	• Basic encryption for sensitive user information • Compliance with standard web security practices • Protection against common security threats (SQL injection, XSS) • Secure handling of user personal information

3.5 Chapter Summary and Evaluation

This chapter has successfully established a comprehensive methodology and requirements framework for developing DialSmart: Malaysia's Intelligent Mobile Advisor. The selection of an Incremental Development Model with Agile practices provides the necessary structure for systematic development while maintaining flexibility for cultural adaptation and user feedback integration. The fact-gathering techniques, particularly observation and planned survey methodologies, have revealed critical insights into Malaysian smartphone consumer behavior, including decision paralysis, technical specification confusion, and the need for culturally-aware advisory systems. The systematic fact-recording approach using modern collaborative tools ensures proper documentation and analysis capabilities throughout the development process. The requirements analysis has produced detailed functional and non-functional specifications that address the unique needs of Malaysian consumers while maintaining high standards for performance, security, and usability. The comprehensive system flow diagram and modular requirements structure provide a clear roadmap for implementation phases, ensuring that the final system will effectively bridge the gap between complex smartphone technology and Malaysian consumer understanding while respecting cultural preferences and economic constraints.

Chapter 4

System Design

4 System Design

This chapter provides a comprehensive examination of the system design and architecture for DialSmart: Malaysia's Intelligent Mobile Advisor. The chapter covers the detailed system architecture that supports the AI-powered smartphone recommendation platform, including the modular design approach that separates user-facing components from administrative functions and machine learning operations.

The system design encompasses seven core modules integrated through a scalable web-based architecture: User Management Module for profile customization and authentication, Brand & Phone Details Module for comprehensive smartphone information display, Phone Finder & Filter Module for advanced search capabilities, Phone Comparison Module for side-by-side analysis, AI Chatbot Module for conversational assistance, Admin Management Module for system oversight, and AI Model Development Module for machine learning integration. Each module is designed to operate independently while maintaining seamless data flow and user experience consistency across the platform.

The chapter details the technical architecture that supports both user and administrative workflows, including the frontend interface design using HTML5, CSS3, and JavaScript for responsive user interactions, and the backend infrastructure utilizing Python with Flask framework for robust API development and database management. The design incorporates MySQL for structured data storage, Google Colab integration for machine learning model training, and RESTful API architecture for efficient communication between system components.

Special attention is given to the conversational AI architecture that enables natural language processing for Malaysian English variations, the recommendation engine design that processes user preferences through Decision Tree and Random Forest algorithms, and the cultural localization framework that adapts system responses to Malaysian consumer behavior patterns. The system design also addresses scalability considerations for handling concurrent users, data security protocols for protecting user information, and integration patterns that enable real-time smartphone pricing and availability updates.

Through comprehensive system flow diagrams, database entity relationship models, class diagrams for object-oriented components, and deployment architecture illustrations, this chapter establishes the technical foundation necessary for implementing a sophisticated yet accessible smartphone advisory system tailored specifically for the Malaysian market's unique requirements and cultural context.

4.1 Entity Relationship Diagram (ERD)

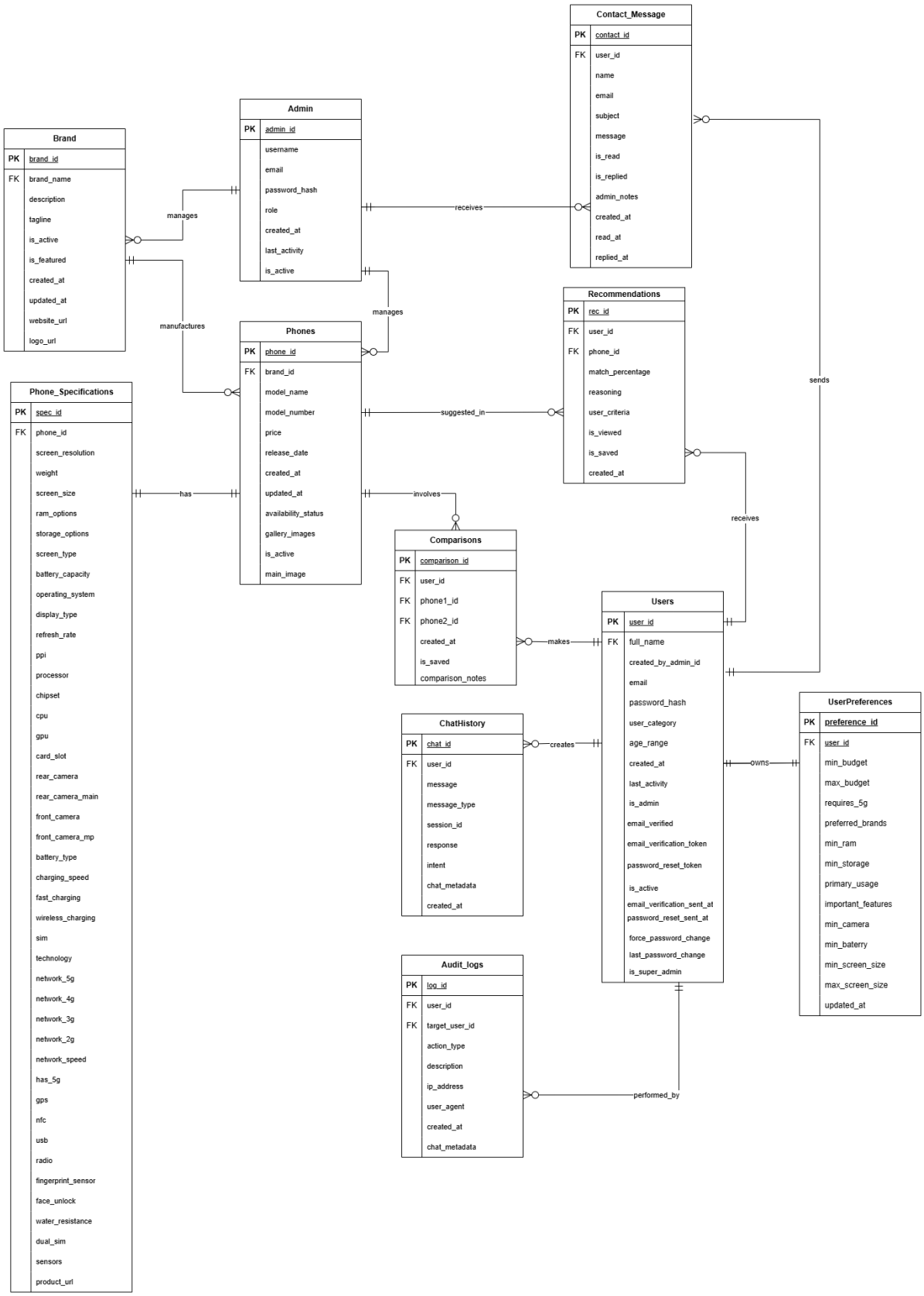

Figure 4.1 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) serves as the foundational blueprint for the DialSmart database architecture, illustrating the relationships between core entities including Users, Smartphones, Brands, Categories, Specifications, Recommendations, Comparisons, ChatHistory, and UserPreferences.

The ERD ensures data integrity and efficient storage by defining how user profiles connect to their preferences, how smartphones relate to their specifications and brands, and how the system tracks user interactions through recommendations and comparisons. This relational structure enables the AI recommendation engine to effectively process user data and smartphone specifications to generate personalized suggestions.

4.2 Class Diagram

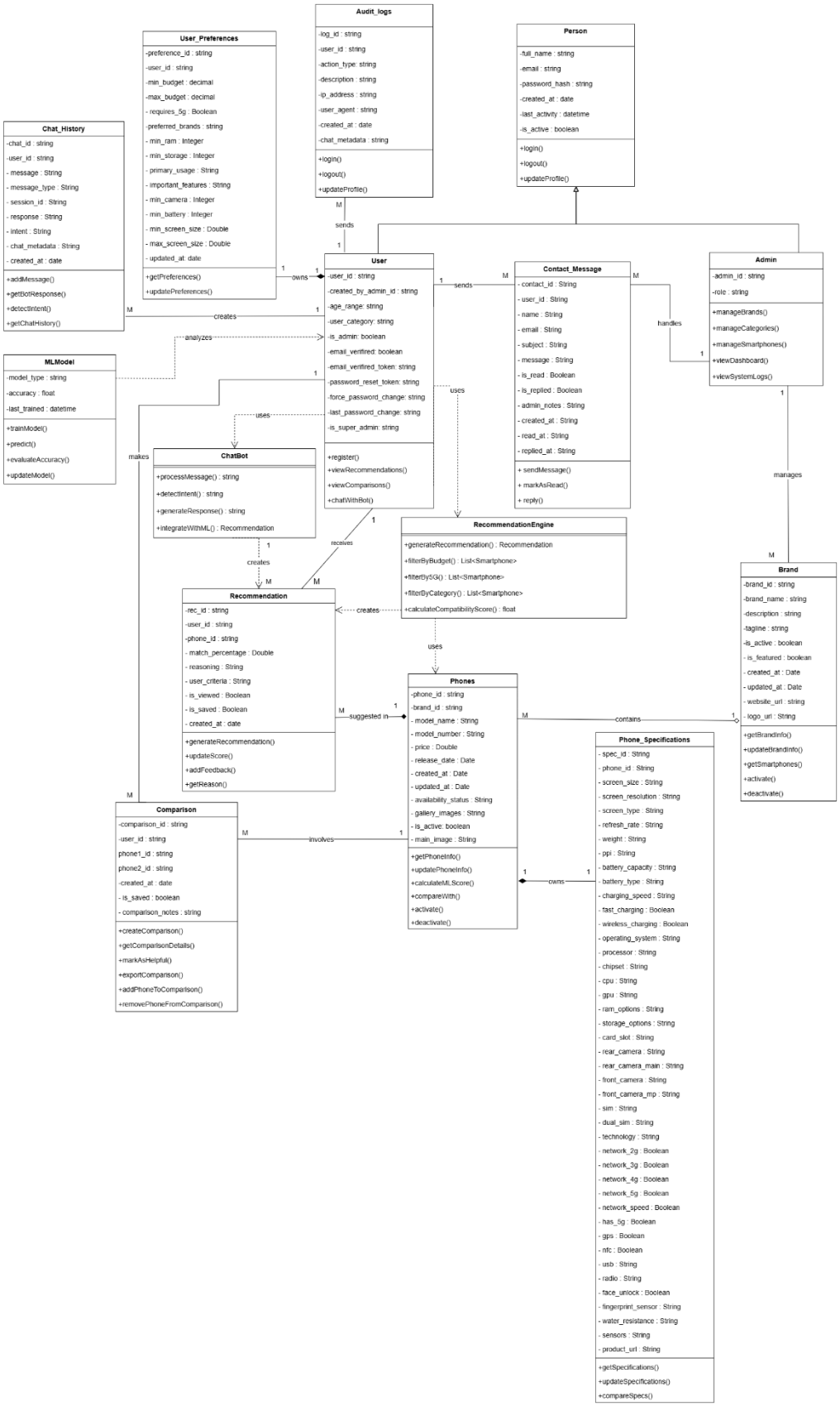

Figure 4.2 Class Diagram

The Class Diagram provides an object-oriented representation of the DialSmart system architecture, defining the classes, attributes, methods, and relationships necessary for implementing the smartphone advisory platform.

It illustrates how different system components interact, including the User class with authentication and preference management methods, the Smartphone class with specification handling capabilities, and the AI recommendation engine classes that process user inputs to generate personalized smartphone suggestions. This diagram guides the development team in implementing consistent object-oriented programming practices and ensures proper encapsulation of functionality across the seven core modules of the system.

4.3 Data Dictionary

4.3.1 User Table

Table 4.1 User Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	USER_ID	STRING	20	Unique identifier for user
FK	FULL_NAME	VARCHAR	50	Username of the user
	CREATED_BY_ADMIN_ID	STRING	20	References USER_ID of the admin who created this user account.
	EMAIL	VARCHAR	100	Email address of user
	PASSWORD_HASH	VARCHAR	255	Encrypted password
	USER_CATEGORY	VARCHAR	50	Category or role of user
	AGE_RANGE	NUMBER	30	User age range
	IS_ADMIN	VARCHAR	100	Occupation of user
	CREATED_AT	DATETIME		Account creation timestamp
	LAST_ACTIVE	DATETIME		Last active/login timestamp.
	IS_ACTIVE	BOOLEAN	1	Shows whether the user account is active (1 = Active, 0 = Inactive).
	EMAIL_VERIFIED	NUMBER	1	Whether the email is verified (1 = Verified, 0 = Not verified).
	EMAIL_VERIFICATION_TOKEN	VARCHAR	255	Token sent to user for verifying email.
	EMAIL_VERIFICATION_SENT_AT	TIMESTAMP	6	Timestamp when verification email was sent.
	PASSWORD_RESET_TOKEN	VARCHAR	100	Token for password reset process.
	PASSWORD_RESET_SENT_AT	TIMESTAMP	6	Timestamp when password reset email was sent.
	FORCE_PASSWORD_CHANGE	NUMBER	1	Forces user to change password on next login (1 = Yes, 0 = No).

4.3.2 Admin Table

Table 4.2 Admin Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	admin_id	STRING	20	Unique identifier for admin
	username	VARCHAR	50	Username of admin
	email	VARCHAR	100	Email address of admin
	password_hash	VARCHAR	255	Encrypted password
	role	VARCHAR	50	Admin role
	created_at	DATETIME		Account creation timestamp
	last_activity	DATETIME		Last activity timestamp
	is_active	BOOLEAN	1	Admin account status (T/F)

4.3.3 Contact_Message Table

Table 4.3 Contact_Message Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	contact_id	STRING	20	Unique identifier for contact message
FK	user_Id	VARCHAR	100	References the user who submitted the message.
	name	VARCHAR	100	Name of the person submitting the contact message.
	email	VARCHAR	150	Email of the sender for follow-up or replies.
	subject	VARCHAR	200	Subject or title of the message.
	message	CLOB		Full message content submitted by the user.
	is_read	NUMBER	1	Indicates whether the message has been read (1 = Yes, 0 = No).
	is_replied	NUMBER	1	Indicates whether an admin has replied (1 = Yes, 0 = No).
	admin_notes	CLOB		Internal notes added by admin
	created_at	DATE		Date when the message was created
	read_at	DATE		Date when the message was marked as read.
	replied_at	DATE		Date when the admin replied to the message.

4.3.4 Brand Table

Table 4.4 Brand Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	brand_id	VARCHAR	20	Unique identifier for brand
FK	name	VARCHAR	100	Name of smartphone brand
	description	VARCHAR	100	Short description about the brand.
	tagline	VARCHAR	200	Brand slogan or tagline used for marketing.
	is_active	NUMBER	1	Indicates whether the brand is active in the system (1 = Active, 0 = Inactive).
	IS_FEATURED	NUMBER	1	Marks whether the brand is displayed as a featured brand (1 = Yes, 0 = No).
	CREATED_AT	DATETIME		Brand creation timestamp
	UPDATED_AT	DATETIME		Indicating the last update to the brand record.
	WEBSITE_URL	VARCHAR	500	Official website link for the brand.
	LOGO_URL	VARCHAR	255	URL of the brand's logo image.

4.3.5 Phone Table

Table 4.5 Phone Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	phone_id	STRING	20	Unique identifier for smartphone
FK	brand_id	STRING	20	Reference to brand table
	model_name	VARCHAR	150	Model name of smartphone
	model_number	VARCHAR	50	Manufacturer-assigned model number.
	price	FLOAT	10,2	Price in Malaysian Ringgit
	release_date	DATE		Official release date
	created_at	DATE		Phone creation timestamp
	updated_at	DATE		Timestamp of the most recent update to the record.
	availability_status	VARCHAR	50	Current availability status
	gallery_images	CLOB		JSON or text list storing URLs of multiple product images.
	is_active	NUMBER	1	Indicates if the product is currently active/available (1 = Active, 0 = Inactive).
	main_image	VARCHAR	255	URL/path to product image

4.3.6 Phone_Specifications Table

Table 4.6 Phone_Specifications Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	SPEC_ID	STRING	38	Unique identifier for specifications
FK	PHONE_ID	STRING	38	Reference to phone table
	SCREEN_SIZE	STRING	38	Screen size in inches
	WEIGHT	VARCHAR	350	Weight of phone in grams
	SCREEN_RESOLUTION	VARCHAR	50	Resolution of the display
	RAM_OPTIONS	VARCHAR	350	RAM capacity in GB
	STORAGE_OPTIONS	VARCHAR	350	Storage capacity in GB
	SCREEN_TYPE	VARCHAR	350	Type of display panel
	BATTERY_CAPACITY	VARCHAR	350	Battery capacity in mAh
	OPERATING_SYSTEM	VARCHAR	350	Operating system version
	DISPLAY_TYPE	VARCHAR	350	Additional screen details
	REFRESH_RATE	VARCHAR	350	Screen refresh rate
	PPI	NUMBER	38	Pixel density
	PROCESSOR	VARCHAR	350	Processor name/model.
	CHIPSET	VARCHAR	350	Chipset model used in the device.
	CPU	VARCHAR	350	CPU configuration and clock speeds
	GPU	VARCHAR	350	Graphics processing unit.
	CARD_SLOT	NUMBER	38	Card slot type
	REAR_CAMERA	VARCHAR	350	Overall rear camera setup description.
	REAR_CAMERA_MAIN	VARCHAR	350	Main rear camera megapixel count.
	FRONT_CAMERA	VARCHAR	350	Front camera description.
	FRONT_CAMERA_MP	VARCHAR	350	Front camera megapixel count.
	BATTERY	VARCHAR	38	Battery type
	FAST_CHARGING	VARCHAR	50	Fast charging support (Yes/No).
	WIRELESS_CHARGING	VARCHAR	50	Wireless charging support (Yes/No).

Table 4.6 Phone_Specifications Table (continued)

	SIM	VARCHAR	350	SIM details
	TECHNOLOGY	VARCHAR	350	Network technology supported
	NETWORK_5G	VARCHAR	50	Supports 5G (Yes/No).
	NETWORK_4G	VARCHAR	50	Supports 4G (Yes/No).
	NETWORK_3G	VARCHAR	50	Supports 3G (Yes/No).
	NETWORK_2G	VARCHAR	50	Supports 2G (Yes/No).
	NETWORK_SPEED	VARCHAR	350	Maximum network speed supported.
	HAS_5G	VARCHAR	50	Whether device includes 5G capability (Yes/No).
	GPS	VARCHAR	50	GPS support (Yes/No).
	NFC	VARCHAR	50	NFC support (Yes/No).
	USB	VARCHAR	350	USB type
	RADIO	VARCHAR	50	FM radio support (Yes/No).
	FINGERPRINT_SENSOR	VARCHAR	50	Type & location of fingerprint sensor.
	FACE_UNLOCK	VARCHAR	350	Whether phone supports face unlock.
	WATER_RESISTANCE	VARCHAR	350	IP rating or splash resistance details.
	DUAL_SIM	NUMBER	38	Dual SIM support (Yes/No).
	SENSORS	VARCHAR	350	List of sensors included
	PRODUCT_URL	VARCHAR	350	External product webpage or reference link.

4.3.7 Recommendation Table

Table 4.8 Recommendation Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	rec_id	STRING	20	Unique identifier for recommendation
FK	user_id	STRING	20	Reference to users table
FK	phone_id	STRING	20	Reference to phone table
	match_percentage	NUMBER	5,2	Recommendation match percentage
	reasoning	VARCHAR	380	Explanation of why this phone was recommended.
	user_criteria	CLOB		Stores user's preference details that influenced the recommendation
	is_viewed	NUMBER	1	Indicates whether the user has viewed the recommendation (1 = Yes, 0 = No).
	is_saved	NUMBER	1	Indicates if the user saved/liked the recommendation (1 = Yes, 0 = No).
	created_at	DATE		Timestamp when the recommendation was generated.

4.3.8 Comparison Table

Table 4.9 Comparison Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	comparison_id	STRING	20	Unique identifier for comparison
FK	user_id	STRING	20	Reference to users table
FK	phone1_id	STRING	20	First phone in comparison
FK	phone2_id	STRING	20	Second phone in comparison
	is_saved	NUMBER	1	Indicates whether the comparison is saved by the user (1 = Yes, 0 = No).
	comparison_notes	VARCHAR	200	Optional notes or comments added by the user about the comparison.
	created_at	DATE		Timestamp when the comparison record was created.

4.3.9 ChatHistory Table

Table 4.10 ChatHistory Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	chat_id	STRING	20	Unique identifier for chat record
FK	user_id	STRING	20	Reference to users table
	message	CLOB		Message text sent by the user.
	session_ID	VARCHAR	100	Chat session identifier used to group multiple messages into one session.
	response	CLOB		Chatbot-generated response to the user
	intent	VARCHAR	100	Detected user intent
	Chat_metadata	CLOB		Additional metadata
	Created_at	DATETIME		Timestamp when the chat message was created

4.3.10 UserPreferences Table

Table 4.11 UserPreferences Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	preference_id	STRING	20	Unique identifier for preferences
FK	user_id	STRING	20	Reference to users table
	min_budget	NUMBER	38	Minimum budget in MYR
	max_budget	NUMBER	38	Maximum budget in MYR
	requires_5g	NUMBER	1	Indicates whether the user requires 5G support (1 = Yes, 0 = No).
	preferred_brands	VARCHAR	255	Comma-separated brand preferences
	min_ram	NUMBER	38	Preferred minimum RAM capacity in GB
	min_storage	NUMBER	38	Preferred minimum storage capacity in GB
	Primary_usage	CLOB		User's primary phone usage description
	Important_features	CLOB		List of important features prioritized by the user
	Min_camera	NUMBER	38	Minimum preferred camera resolution
	Min_baterry	NUMBER	38	Minimum preferred battery capacity
	Min_screen_size	NUMBER	38	Minimum preferred screen size
	Max_screen_size	NUMBER	38	Maximum preferred screen size
	Updated_at	DATE		Timestamp indicating the last update to the user's preference

4.3.11 Audit_logs Table

Table 4.12 Audit_logs Table

P.F	Attribute	Data Type	Field Size	Explanation
PK	log_id	STRING	20	Unique identifier for audit log record
FK	user_id	STRING	20	Reference to users table
	action_type	VARCHAR	100	Type of action performed (e.g., login, update, delete, recommendation viewed).
	description	CLOB		Detailed explanation of the action recorded in the log.
	ip_address	VARCHAR	50	Indicates whether the action involves IP or location details (e.g., "IP Logged").
	user_agent	VARCHAR	255	Device, browser, or platform information captured during the action.
	created_at	DATETIME		Timestamp indicating when the audit log record was created.
	chat_metadata	CLOB		Additional metadata stored in JSON (e.g., context, session info, model details).

4.4 Use Case Diagram

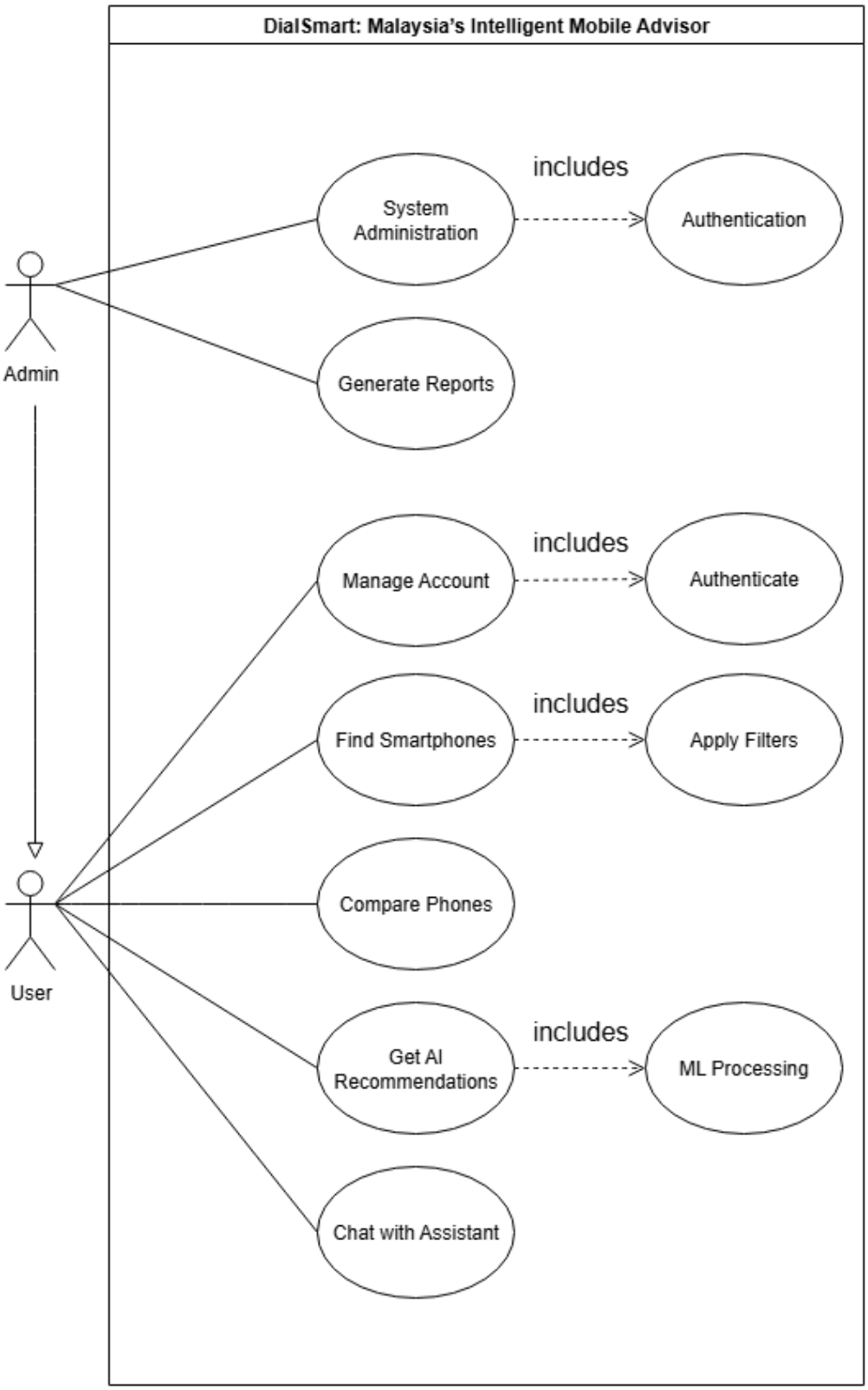

Figure 4.3 Overall Use Case Diagram

Two Main Actors:

- User - Malaysian consumers seeking smartphone guidance
- Admin - System administrators managing the platform

User Functions:

- Manage Account - Registration, login, profile management (includes Authentication)
- Find Smartphones - Search and filter phones by criteria (includes Apply Filters)
- Compare Phones - Side-by-side phone comparison analysis
- Get AI Recommendations - Personalized suggestions (includes ML Processing)
- Chat with Assistant - Conversational AI guidance

Admin Functions:

- System Administration - Manage brands, models, users (includes Authentication)
- Generate Reports - System analytics and performance insights
- All User Functions - Admin can also access all user-side features for testing and system verification

Key Relationships:

- <<include>> relationships show essential sub-processes
- Authentication required for both user and admin access
- ML Processing powers the recommendation engine
- Apply Filters enhances smartphone search functionality
- **Admin inherits User capabilities** - shown by the connecting line, allowing admins to test and experience the system from user perspective

4.5 Detailed Use Case Diagram

4.5.1 User Management Module

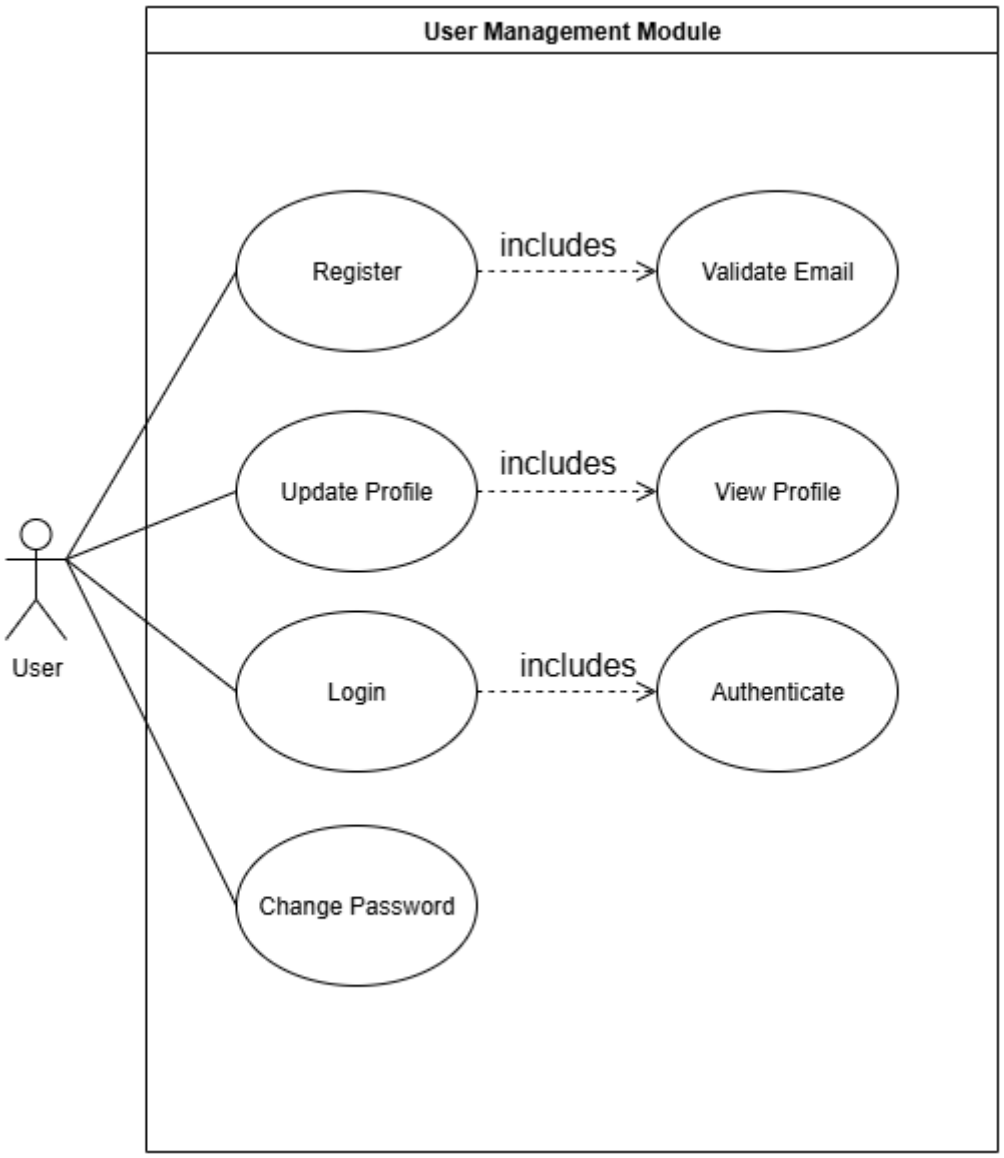

Figure 4.4 Detail Use Case Diagram – Account Management Module

4.5.1.1 User Registration

Table 4.12 Use Case Description – User Registration

Use case name	**User Registration**	
Scenario	New user creates account with personal details	
Triggering event	User clicks register button on homepage	
Brief description	New user provides personal information, email, and password to create an account with category selection for personalized recommendations	
Actors	User	
Related use cases	Validate Email, Authenticate	
Stakeholders	User, System	
Preconditions	User has internet access and valid email	
Postconditions	User account is created and activated	
Flow of Activities	**Actor**	**System**
	1. User clicks register 3. User fills registration details 5. User submits registration 9. User receives verification email 10. User clicks verification link	 2. Display registration form 4. Validate form fields 6. Check email uniqueness 7. Hash password securely 8. Generate verification token 11. Store user in database 12. Display success message
Exception Conditions	6. Email already exists - show error and redirect to login 7. Weak password - display password requirements 8. Verification timeout - allow resend verification	

4.5.1.2 Update Profile

Table 4.13 Use Case Description – Update Profile

Use case name	**Update Profile**	
Scenario	User modifies personal information and preferences	
Triggering event	User clicks edit profile from dashboard	
Brief description	Authenticated user updates personal details, demographic information, and preference settings to improve recommendation accuracy	
Actors	User	
Related use cases	View Profile, Authenticate	
Stakeholders	User, System	
Preconditions	User is logged in with valid session	
Postconditions	User profile information is updated successfully	
Flow of Activities	**Actor**	**System**
	1. User accesses profile page 4. User modifies personal details 6. User saves changes 10. View updated profile	2. Load current profile data 3. Display profile form 5. Validate form fields 7. Update database 8. Refresh user session 9. Display success confirmation
Exception Conditions	5. Invalid data format - show validation errors 7. Database update failed - retry or show error 8. Session expired - redirect to login	

4.5.1.3 User Login

Table 4.14 Use Case Description – User Login

Use case name	**User Login**	
Scenario	Existing user authenticates to access system	
Triggering event	User clicks login button	
Brief description	Registered user provides credentials to authenticate and access personalized dashboard with saved preferences and history	
Actors	User	
Related use cases	Authenticate, Change Password	
Stakeholders	User, System	
Preconditions	User has valid registered account	
Postconditions	User is authenticated with active session	
Flow of Activities	**Actor**	**System**
	2. User enters email and password 3. User clicks login button 5. User accesses member dashboard	1. Display login form 4. System validates user input/ credentials exist 6. Create user session & update last login time
Exception Conditions	4. Invalid credentials / account not registered - display error message	

4.5.1.4 Change Password

Table 4.15 Use Case Description – Change Password

Use case name	**Change Password with Security Validation**	
Scenario	User updates their account password for security purposes	
Triggering event	User accesses password change settings or receives security prompt	
Brief description	System allows users to securely change their password with current password verification and new password strength validation	
Actors	User	
Related use cases	Authenticate (for current password verification)	
Stakeholders	User, Security System, Database	
Preconditions	User is logged in, knows current password	
Postconditions	Password is successfully changed and user account remains secure	
Flow of Activities	**Actor**	**System**
	1. User accesses change password 2. User enters current password 4. User enters new password 6. User confirms new password 8. User submits password change 13. User receives confirmation	 3. System verifies current password 5. System validates password strength 7. System checks password match 9. System encrypts new password 10. System updates password in database 11. System logs security event 12. System confirms password change
Exception Conditions	3. Incorrect current password 5. Weak new password 7. Passwords don't match 10. Database update failure 11. Security validation error	

4.5.2 AI Recommendation Module

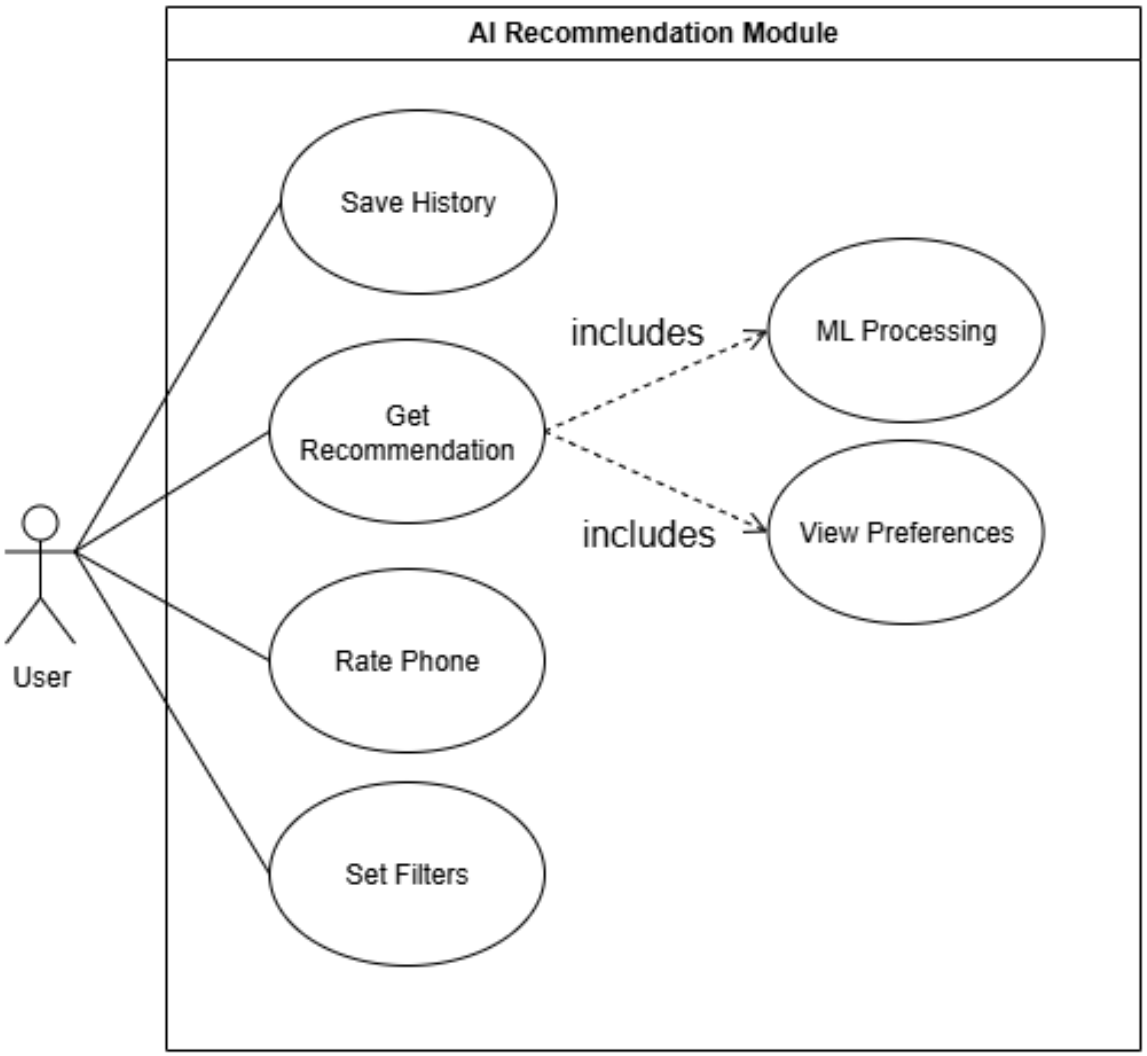

Figure 4.5 Detail Use Case Diagram – AI Recommendation Module

4.5.2.1 Save History

Table 4.16 Use Case Description – Save Recommendation History

Use case name	**Save Recommendation History**	
Scenario	User saves their recommendation results for future reference and comparison	
Triggering event	User clicks "Save to History" button after viewing recommendations	
Brief description	System stores user's recommendation session including selected phones, preferences used, and timestamp for future access	
Actors	User	
Related use cases	Get Recommendation, View Preferences	
Stakeholders	User, System	
Preconditions	User has received recommendations, user is logged in	
Postconditions	Recommendation session is saved to user's history	
Flow of Activities	Actor	System
	1. User views recommendations 2. User clicks "Save to History" 6. User receives confirmation	 3. System captures current recommendation data 4. System saves preferences, phones, and timestamp 5. System confirms save operation
Exception Conditions	4. database save error, storage limit exceeded	

4.5.2.2 Get AI Recommendation

Table 4.17 Use Case Description – Get AI Recommendation

Use case name	**Get AI Recommendation with ML Processing**	
Scenario	User requests personalized smartphone recommendations	
Triggering event	User request smartphone recommendations	
Brief description	System processes user preferences through ML model to generate personalized smartphone suggestions with scoring and explanations	
Actors	User	
Related use cases	ML Processing, View Preferences, Set Filters	
Stakeholders	User, ML Model, System	
Preconditions	User preferences are set, phone database is populated	
Postconditions	Personalized recommendations are generated and displayed	
Flow of Activities	Actor	System
	1. User requests recommendation 2. User set filer preferences 7. User reviews recommendations 8. User clicks phone details 9. User saves recommendation	 3. System loads user preferences 4. System processes data through ML model 5. System generates ranked recommendations 6. System displays recommendations with rationale 10. Save to database
Exception Conditions	4. ML model unavailable, insufficient user data, no matching phones found	

4.5.2.3 Rate phone

Table 4.18 Use Case Description – Rate Phone

Use case name	**Rate Phone Recommendation**	
Scenario	User provides feedback on recommended phones to improve future AI recommendations	
Triggering event	User clicks rating stars or feedback button on recommended phone	
Brief description	System collects user ratings and feedback on recommended phones to enhance ML model accuracy and personalization	
Actors	User	
Related use cases	Get Recommendation, ML Processing	
Stakeholders	User, ML Model, System	
Preconditions	User has viewed recommendations, phone is displayed	
Postconditions	User rating is recorded and used for ML model improvement	
Flow of Activities	Actor	System
	1. User views recommended phone 2. User clicks rating interface 3. User selects star rating (1-5) 4. User provides optional feedback 9. User receives thank you message	5. System validates rating input 6. System records rating with phone and user ID 7. System updates ML training data 8. System confirms rating submission
Exception Conditions	4. ML model unavailable, insufficient user data, no matching phones found	

4.5.2.4 Set Filters

Table 4.19 Use Case Description –Set Filters

Use case name	**Set Recommendation Filters**	
Scenario	User configures filtering criteria to refine recommendation results	
Triggering event	User accesses filter settings or modifies existing filter preferences	
Brief description	System allows user to set filtering criteria such as price range, brand preferences, features, and usage patterns to customize recommendations	
Actors	User	
Related use cases	Get Recommendation, View Preferences	
Stakeholders	User, System	
Preconditions	User has access to filter interface	
Postconditions	Filter preferences are saved and applied to future recommendations	
Flow of Activities	Actor	System
	1. User accesses filter settings 2. User sets price range 3. User selects preferred brands 4. User chooses feature priorities 5. User confirms filter settings	6. System validates filter inputs 7. System saves filter preferences 8. System applies filters to recommendation engine 9. System confirms filter update
	10. User receives confirmation	
Exception Conditions	6. Invalid filter combinations 7. conflicting preferences, save operation failure	

4.5.3 Phone Comparison Module

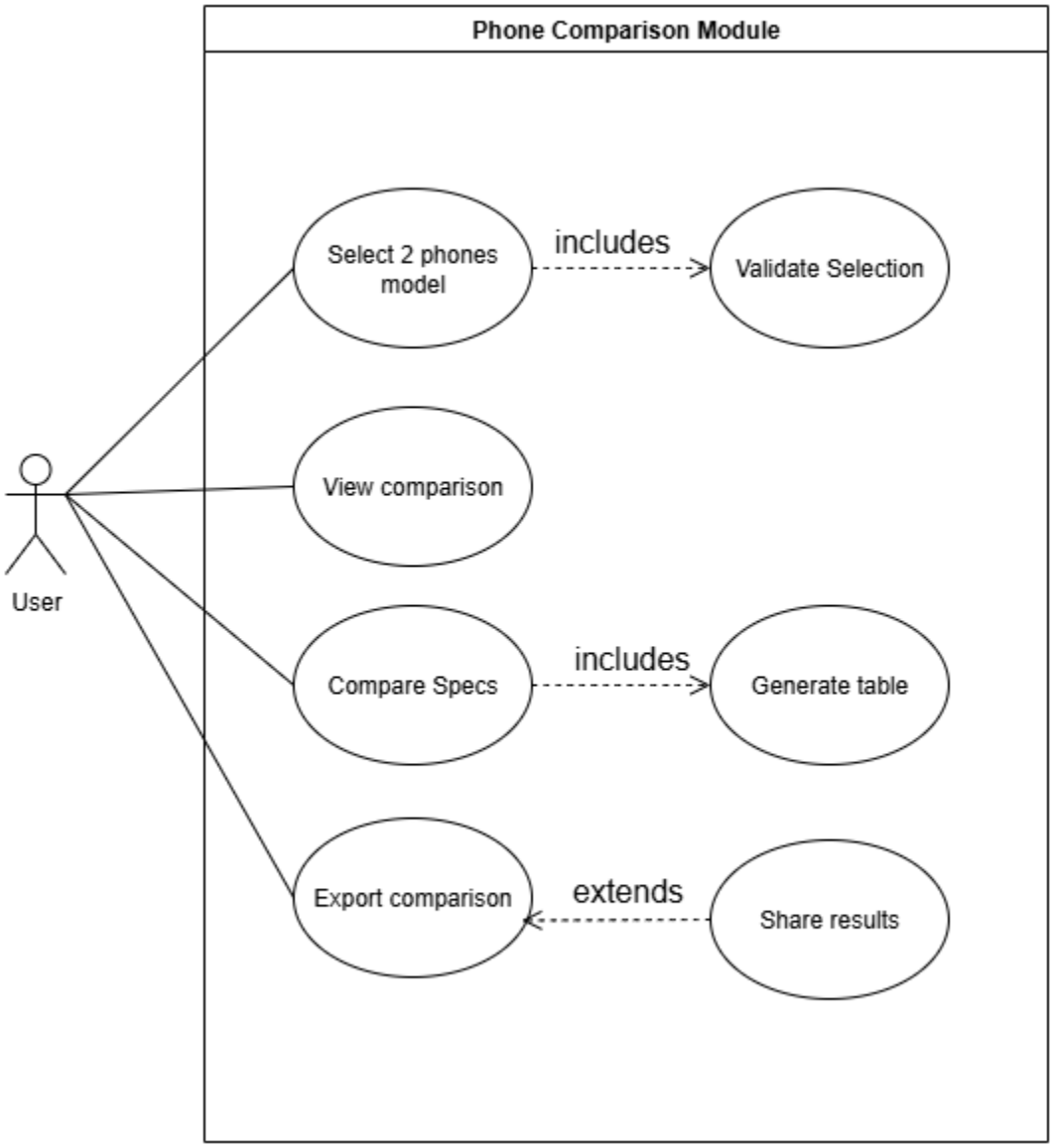

Figure 4.6 Detail Use Case Diagram – Phone Comparison Module

4.5.3.1 Select 2 Phones Comparison

Table 4.20 Use Case Description – Select 2 phones comparison

Use case name	**Select 2 Phones Comparison**	
Scenario	User selects two smartphone models for detailed side-by-side comparison	
Triggering event	User clicks on phone selection interface or "Compare" button	
Brief description	System allows user to select exactly two smartphone models from available options with validation to ensure proper comparison setup	
Actors	User	
Related use cases	Validate Selection, View Comparison	
Stakeholders	User, System	
Preconditions	Phone database is populated, user has access to comparison interface	
Postconditions	Two valid phones are selected and ready for comparison	
Flow of Activities	**Actor**	**System**
	1. User accesses comparison interface 2. User selects first phone 4. User selects second phone 8. User confirms selection	 3. System validates first selection 5. System validates second selection 6. System checks for duplicate selection 7. System enables comparison functionality 9. System prepares comparison data
Exception Conditions	5. Same phone selected twice, invalid phone selection, phone data unavailable	

4.5.3.2 View Comparison

Table 4.21 Use Case Description – View Comparison

Use case name	**View Side-by-Side Phone Comparison**	
Scenario	User views detailed comparison between two selected smartphone models	
Triggering event	User clicks "View Comparison" after selecting two phones	
Brief description	System displays comprehensive side-by-side comparison of specifications, features, and pricing	
Actors	User	
Related use cases	Select 2 Phones Model, Compare Specs	
Stakeholders	User, System	
Preconditions	Two phones are selected and validated	
Postconditions	Comparison table is displayed with all relevant information	
Flow of Activities	**Actor**	**System**
	1. User requests comparison view 5. User reviews comparison data 6. User scrolls through specifications	2. System retrieves phone specifications 3. System formats comparison layout 4. System displays side-by-side comparison
Exception Conditions	Phone data incomplete, comparison formatting error, display timeout	

4.5.3.3 Compare Specs

Table 4.22 Use Case Description – Compare Specs

Use case name	**Compare Specs**	
Scenario	User performs detailed technical specification comparison between selected phones	
Triggering event	User accesses detailed specs tab in comparison view	
Brief description	System generates comprehensive comparison table highlighting differences and similarities in technical specifications	
Actors	User	
Related use cases	View Comparison, Generate Table	
Stakeholders	User, System	
Preconditions	Two phones selected, specifications data available	
Postconditions	Detailed specification comparison table is generated and displayed	
Flow of Activities	**Actor**	**System**
	1. User clicks "Compare Specs" 6. User analyzes specifications 7. User identifies preferred features	2. System extracts technical specifications 3. System generates comparison table 4. System highlights key differences 5. System displays performance metrics
Exception Conditions	2. Incomplete specification data 3. Table generation error, performance calculation failure	

4.5.3.4 Export Comparison

Table 4.23 Use Case Description – Export Comparison

Use case name	Export Comparison with Share Results	
Scenario	User exports their phone comparison analysis for offline reference or sharing with others	
Triggering event	User clicks "Export" or "Share" button after completing phone comparison	
Brief description	System generates exportable comparison report in multiple formats and provides sharing options via email, social media, or direct download	
Actors	User	
Related use cases	Share Results, View Comparison, Compare Specs	
Stakeholders	User, Export System, File Generation Service	
Preconditions	Phone comparison is completed and displayed, both phones have valid data	
Postconditions	Comparison report is generated and made available for download or sharing	
Flow of Activities	**Actor**	**System**
	1. User completes phone comparison 2. User clicks "Export Comparison" 3. User selects export format (PDF) 7. User chooses sharing method 10. User downloads file or shares link	 4. System validates comparison data 5. System generates formatted report 6. System creates downloadable file 8. System processes sharing request 9. System generates shareable link 11. System logs export activity 12. System confirms successful export
Exception Conditions	2. Incomplete specification data 3. Table generation error, performance calculation failure	

4.5.4 Chatbot Module

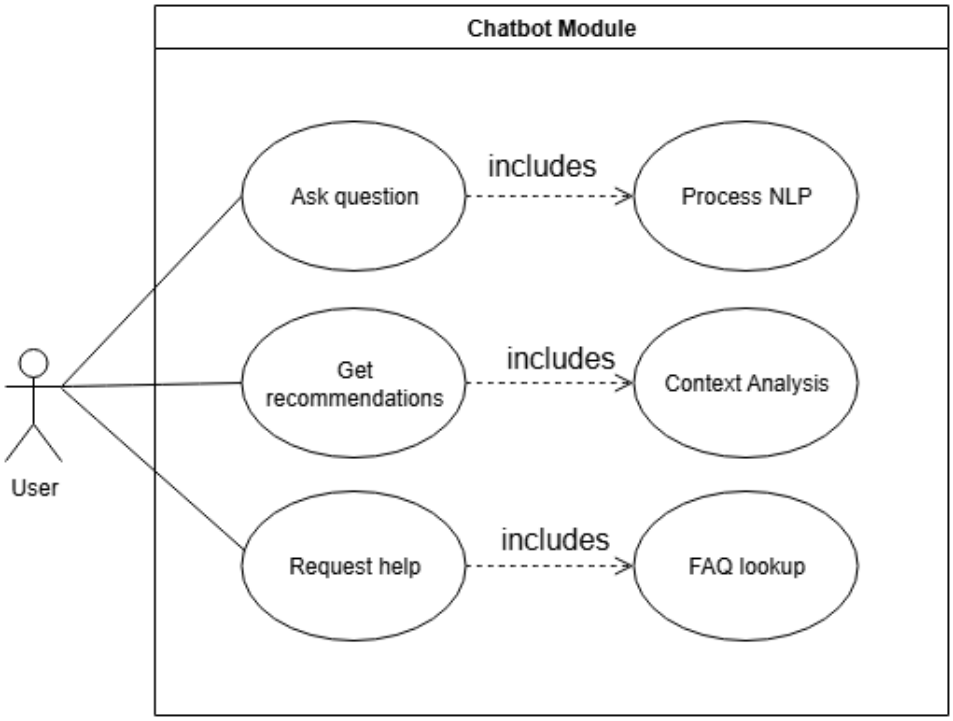

Figure 4.7 Detail Use Case Diagram – Chatbot Module

4.5.4.1 Ask Questions

Table 4.24 Use Case Description – Ask Questions

Use case name	**Ask Question with NLP Processing**	
Scenario	User asks smartphone-related questions and receives intelligent responses	
Triggering event	User types question and presses send or enter	
Brief description	System processes user queries using natural language processing to understand intent and provide relevant answers	
Actors	User	
Related use cases	Process NLP, Start Chat	
Stakeholders	User, NLP Engine, Chatbot System	
Preconditions	Chat session is active, NLP service is running	
Postconditions	User receives relevant answer to their question	
Flow of Activities	**Actor**	**System**
	1. User types question 2. User sends message 7. User receives answer 8. User can ask follow-up questions	 3. System processes message through NLP 4. System analyzes user intent 5. System retrieves relevant information 6. System generates response
Exception Conditions	3. NLP processing error, unrecognized query 6. System response timeout	

4.5.4.2 Get Recommendations

Table 4.25 Use Case Description – Get Recommendation

Use case name	**Get Recommendations with Context Analysis**	
Scenario	User requests smartphone recommendations through conversational interface	
Triggering event	User asks for phone recommendations or guidance	
Brief description	System analyzes conversation context and user preferences to provide personalized smartphone recommendations	
Actors	User	
Related use cases	Context Analysis, Ask Question	
Stakeholders	User, ML Engine, Chatbot System	
Preconditions	Chat session active, recommendation engine available	
Postconditions	User receives personalized smartphone suggestions	
Flow of Activities	**Actor**	**System**
	1. User requests recommendations	
		2. System analyzes conversation context 3. System extracts user preferences 4. System queries recommendation engine 5. System formats recommendations
	6. User reviews suggestions 7. User asks for clarifications	
		8. System provides additional details
Exception Conditions	2. Insufficient user data 4. Recommendation engine failure, context analysis error	

4.5.4.3 Request Help

Table 4.26 Use Case Description – Request Help

Use case name	**Request Help with FAQ Lookup**	
Scenario	User requests general help or guidance on using the system	
Triggering event	User types help commands or clicks help button	
Brief description	System provides assistance through FAQ lookup and general guidance on system features	
Actors	User	
Related use cases	FAQ Lookup, Start Chat	
Stakeholders	User, FAQ System, Chatbot	
Preconditions	Chat session active, FAQ database available	
Postconditions	User receives helpful information and guidance	
Flow of Activities	**Actor**	**System**
	1. User requests help 5. User selects help topic 7. User follows guidance	 2. System searches FAQ database 3. System identifies relevant help topics 4. System presents help options 6. System displays detailed information
Exception Conditions	2. FAQ database unavailable 3. No relevant help found, help content outdated	

4.5.5 Admin Management Module

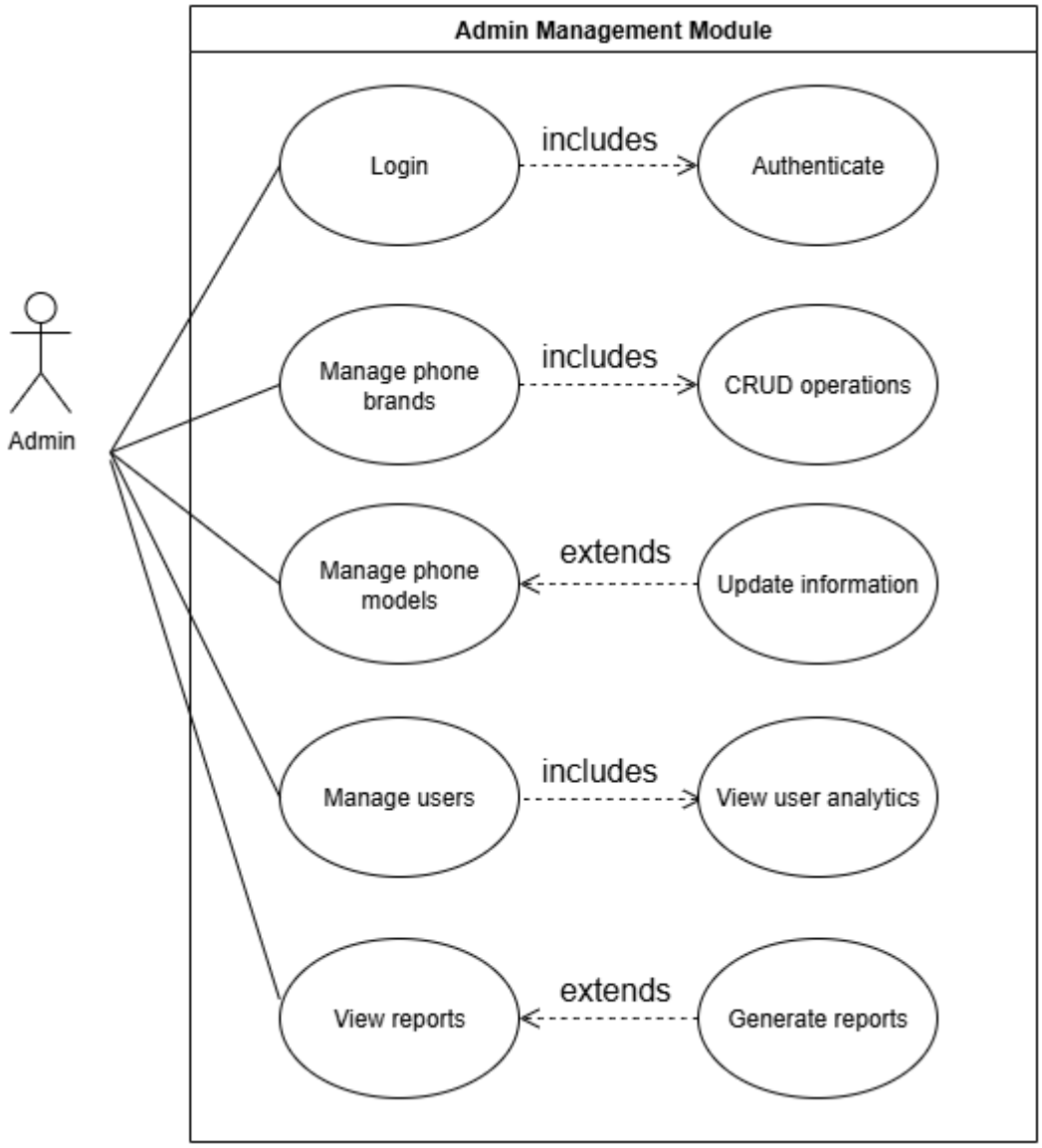

Figure 4.8 Detail Use Case Diagram – Admin Management Module

4.5.5.1 Admin Login

Table 4.27 Use Case Description – Admin Login

Use case name	**Admin Login with Authentication**	
Scenario	Administrator logs into the system to access management functions	
Triggering event	Admin enters credentials and clicks login button	
Brief description	System authenticates admin credentials and grants access to administrative functions	
Actors	Admin	
Related use cases	Authenticate	
Stakeholders	Admin, Security System	
Preconditions	Admin has valid credentials, system is operational	
Postconditions	Admin gains access to administrative dashboard	
Flow of Activities	**Actor**	**System**
	1. Admin enters username 2. Admin enters password 3. Admin clicks login 8. Admin accesses admin functions	 4. System validates credentials 5. System checks admin privileges 6. System creates secure session 7. System grants dashboard access
Exception Conditions	4. Invalid credentials, account locked, system authentication failure	

4.5.5.2 Manage Phone Brands

Table 4.28 Use Case Description – Manage Phone Brand

Use case name	**Manage Phone Brands with CRUD Operations**	
Scenario	Administrator manages smartphone brand information in the system	
Triggering event	Admin accesses brand management section	
Brief description	System allows admin to create, read, update, and delete smartphone brand information	
Actors	Admin	
Related use cases	CRUD Operations	
Stakeholders	Admin, System Database	
Preconditions	Admin is authenticated, brand management module is accessible	
Postconditions	Brand information is updated in the system	
Flow of Activities	**Actor**	**System**
	1 Admin accesses model management 2. Admin selects phone model 4. Admin updates specifications 6. Admin uploads product images 8. Admin saves changes	3. System displays model details 5. System validates input data 7. System processes image files 9. System updates database 10. System refreshes user interface
Exception Conditions	5. Invalid specification data 7. Image upload failure 9. model update error	

4.5.5.3 Manage Phone Models

Table 4.29 Use Case Description – Manage Phone Models

Use case name	**Manage Phone Models with Update Information**	
Scenario	Administrator manages smartphone model details and specifications	
Triggering event	Admin accesses phone model management section	
Brief description	System enables admin to add, modify, and update smartphone model information and specifications	
Actors	Admin	
Related use cases	Update Information	
Stakeholders	Admin, System Database	
Preconditions	Admin authenticated, phone brands exist, model management accessible	
Postconditions	Phone model information is updated and available for users	
Flow of Activities	**Actor**	**System**
	1. Admin accesses brand management 2. Admin selects CRUD operation 4. Admin inputs brand information 6. Admin confirms changes 9. Admin reviews updated information	3. System displays brand interface 5. System validates brand data 7. System updates database 8. System confirms operation success
Exception Conditions	4. Invalid brand data 7. Database update failure, duplicate brand entry	

4.5.5.4 Manage Users

Table 4.30 Use Case Description – Manager Users

Use case name	**Manage Users with View User Analytics**	
Scenario	Administrator oversees user accounts and analyzes user behavior	
Triggering event	Admin accesses user management dashboard	
Brief description	System provides admin with user management capabilities and analytics on user activity and behavior patterns	
Actors	Admin	
Related use cases	View User Analytics	
Stakeholders	Admin, User Database, Analytics System	
Preconditions	Admin authenticated, user data available	
Postconditions	Admin has current view of user management and analytics	
Flow of Activities	**Actor**	**System**
	1. Admin accesses user management	2. System displays user list 3. System shows user statistics
	4. Admin reviews user accounts 5. Admin selects analytics view	6. System generates user analytics 7. System displays usage patterns
	8. Admin analyzes user behavior 9. Admin takes management actions	10. System implements admin decisions
Exception Conditions	2. User data unavailable 6. Analytics generation failure, insufficient permissions	

4.5.5.5 View Reports

Table 4.31 Use Case Description – View Reports

Use case name	**View Reports with Generate Reports**	
Scenario	Administrator generates and reviews system performance and usage reports	
Triggering event	Admin accesses reports section or requests specific report	
Brief description	System generates comprehensive reports on system usage, user behavior, and performance metrics	
Actors	Admin	
Related use cases	Generate Reports	
Stakeholders	Admin, Reporting System, Database	
Preconditions	Admin authenticated, system data available for reporting	
Postconditions	Reports are generated and available for admin review	
Flow of Activities	**Actor**	**System**
	1. Admin requests reports 2. Admin selects report type 7. Admin analyzes reports 8. Admin exports reports	 3. System queries database 4. System processes report data 5. System generates visual reports 6. System displays report results 9. System creates exportable files
Exception Conditions	4. Insufficient data for reports 5. Report generation failure, export error	

4.5.6 Phone Finder & Filter Module

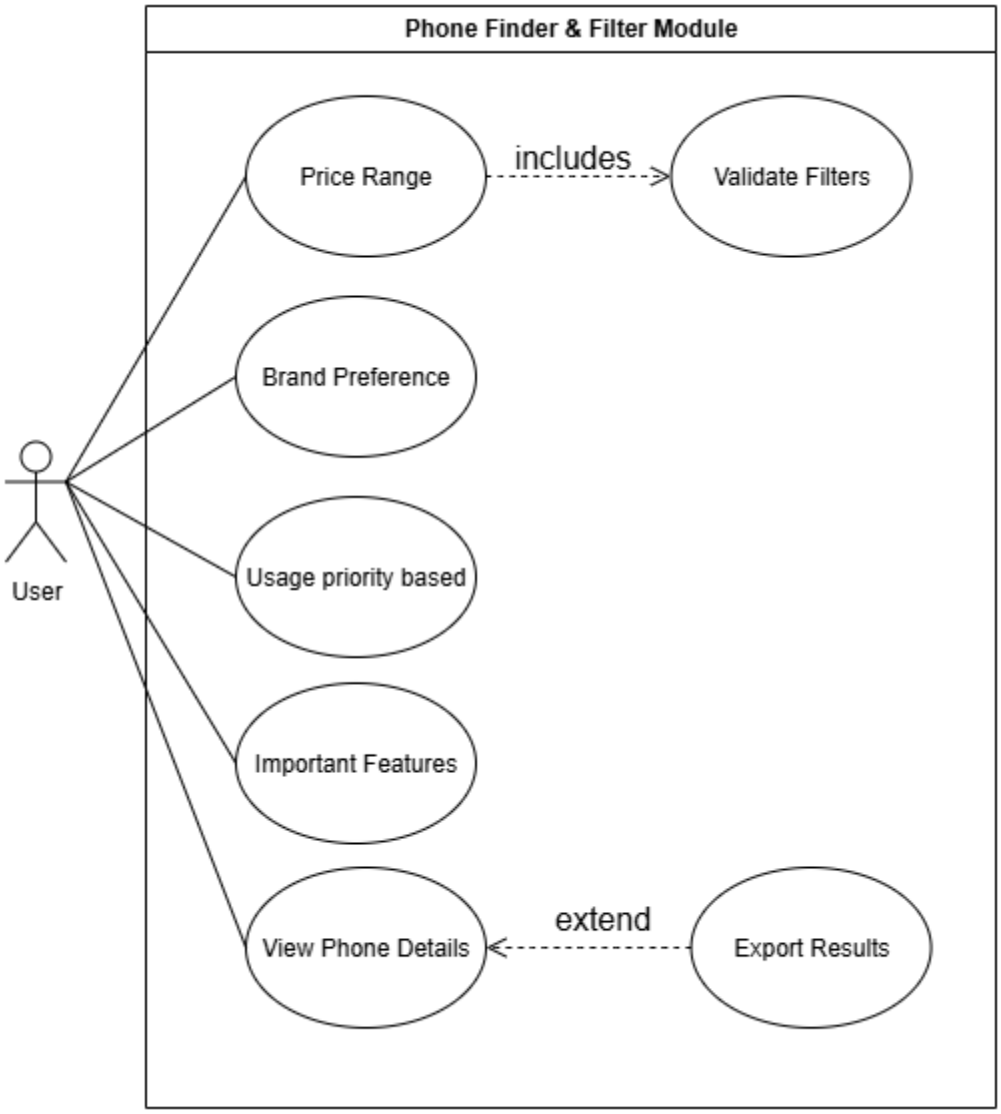

Figure 4.9 Detail Use Case Diagram – Phone Finder & Filter Module

4.5.6.1 Price Range

Table 4.32 Use Case Description – Price Range

Use case name	**Set Price Range with Filter Validation**	
Scenario	User sets budget constraints to filter smartphones within their price range	
Triggering event	User adjusts price range slider or inputs price values	
Brief description	System allows user to define minimum and maximum price limits with validation to filter available smartphones	
Actors	User	
Related use cases	Validate Filters	
Stakeholders	User, Filter System	
Preconditions	Phone finder interface is loaded, price data is available	
Postconditions	Price filter is applied and phone list is updated	
Flow of Activities	**Actor**	**System**
	1. User accesses price filter 2. User sets minimum price 4. User sets maximum price 8. User reviews filtered results	 3. System validates price input 5. System validates price range 6. System applies price filter 7. System updates phone results
Exception Conditions	3. Invalid price range 5. Minimum exceeds maximum 7. No phones in price range	

4.5.6.2 Brand Preference

Table 4.33 Use Case Description – Brand Preference

Use case name	**Set Brand Preference Filter**	
Scenario	User selects preferred smartphone brands to narrow search results	
Triggering event	User checks/unchecks brand checkboxes or selects brand options	
Brief description	System filters smartphones based on user's preferred brands and manufacturers	
Actors	User	
Related use cases	Validate Filters	
Stakeholders	User, Filter System	
Preconditions	Brand filter interface available, brand data loaded	
Postconditions	Brand filter applied and results updated accordingly	
Flow of Activities	**Actor**	**System**
	1. User accesses brand filter 2. User selects preferred brands 4. User confirms brand choices 7. User reviews brand-filtered phones	 3. System validates brand selection 5. System applies brand filter 6. System updates search results
Exception Conditions	3. No brands selected, invalid brand selection 6. No phones for selected brands	

4.5.6.3 Usage Priority Based

Table 4.34 Use Case Description –Usage Priority Based

Use case name	**Set Usage Priority Based Filter**	
Scenario	User defines intended phone usage patterns to get suitable recommendations	
Triggering event	User selects usage categories like photography, gaming, work, or entertainment	
Brief description	System filters smartphones based on usage priorities such as camera quality for photography, performance for gaming, etc.	
Actors	User	
Related use cases	Validate Filters	
Stakeholders	User, Filter System	
Preconditions	Usage filter options available, phone specifications support usage categorization	
Postconditions	Usage-based filter applied and relevant phones displayed	
Flow of Activities	**Actor**	**System**
	1. User accesses usage filter 2. User selects usage priorities 4. User prioritizes usage types 8. User reviews usage-optimized results	 3. System validates usage selection 5. System maps usage to specifications 6. System applies usage-based filter 7. System ranks phones by usage fit
Exception Conditions	3. No usage priorities selected, conflicting usage types, insufficient specification data	

4.5.6.4 Important Features

Table 4.35 Use Case Description – Important Features

Use case name	**Select Important Features Filter**	
Scenario	User specifies essential smartphone features to find matching devices	
Triggering event	User selects important features like 5G, camera quality, battery life, storage	
Brief description	System filters smartphones based on user-specified important features and technical requirements	
Actors	User	
Related use cases	Validate Filters	
Stakeholders	User, Filter System	
Preconditions	Feature filter interface available, feature data for phones exists	
Postconditions	Feature-based filter applied and matching phones displayed	
Flow of Activities	**Actor**	**System**
	1. User accesses feature filter 2. User selects important features 4. User sets feature priorities 8. User reviews usage-optimized results	 3. System validates feature selection 5. System processes feature requirements 6. System applies feature filter 7. System highlights matching features
Exception Conditions	3. No features selected, incompatible feature combinations, feature data missing	

4.5.6.5 View Phone Details

Table 4.36 Use Case Description – View Phone Details

Use case name	**View Phone Details with Export Results**	
Scenario	User views detailed information about filtered smartphones and exports results	
Triggering event	User clicks on phone from filtered results or requests detailed view	
Brief description	System displays comprehensive phone information and provides option to export filtered results for offline reference	
Actors	User	
Related use cases	Export Results	
Stakeholders	User, Phone Database, Export System	
Preconditions	Phone filtering completed, detailed phone data available	
Postconditions	Phone details displayed and optionally exported	
Flow of Activities	**Actor**	**System**
	1. User selects phone from results	
		2. System retrieves detailed information 3. System displays comprehensive specs
	4. User reviews phone details 5. User requests export results	
		6. System generates export file 7. System provides download/share options
	8. User downloads or shares results	
Exception Conditions	2. Phone details unavailable 6. Export generation failure 7. Download error	

4.6 Activity Diagram

4.6.1 User Registration

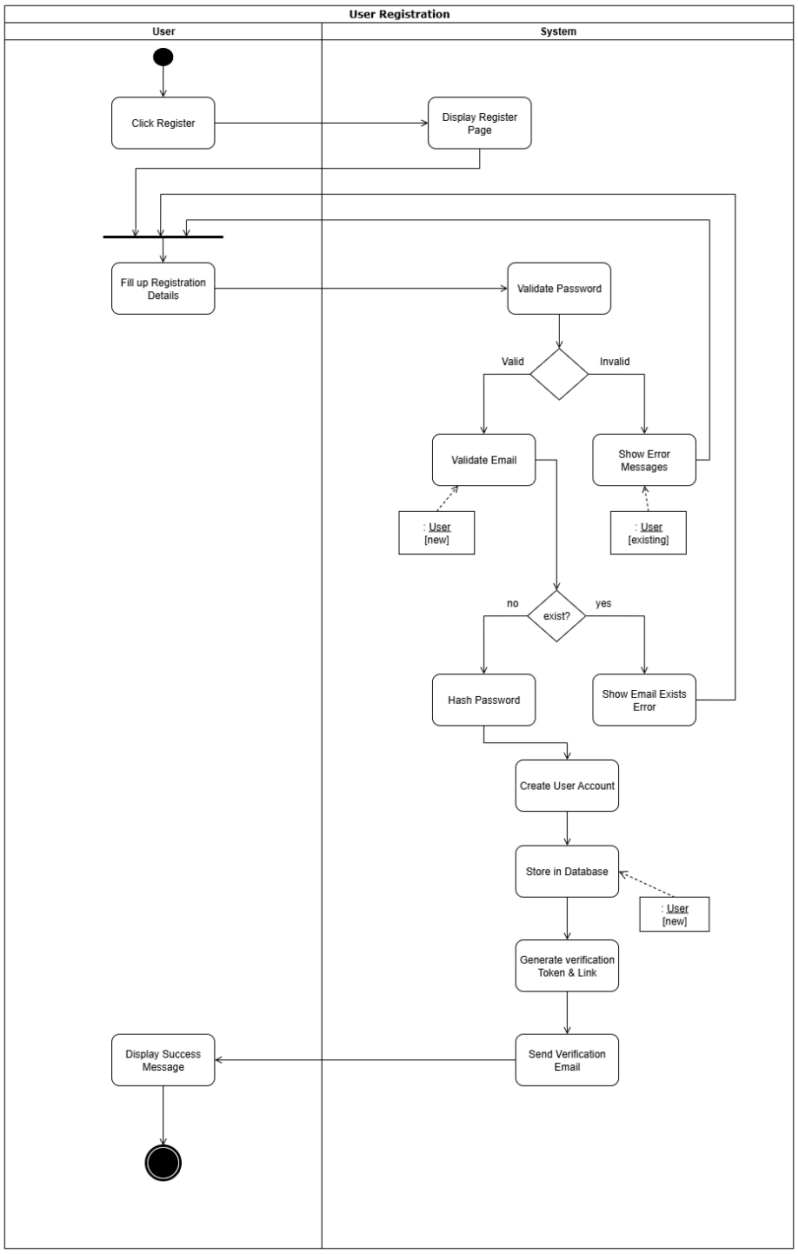

Figure 4.10 Activity Diagram – User Registration

The User Registration activity diagram illustrates the comprehensive workflow for new user account creation in the DialSmart system. The process begins when a user clicks the register button, triggering the display of the registration page.

The system implements a robust validation framework that includes password strength verification, email format checking, and duplicate email detection. Upon successful form submission, the system creates the user account and stores encrypted credentials in the database.

The workflow includes email verification through token generation and link sending, ensuring account security and validity. Error handling paths are integrated throughout the process to manage validation failures, database errors, and network issues, providing users with clear feedback and recovery options.

4.6.2 User Login/Forget Password

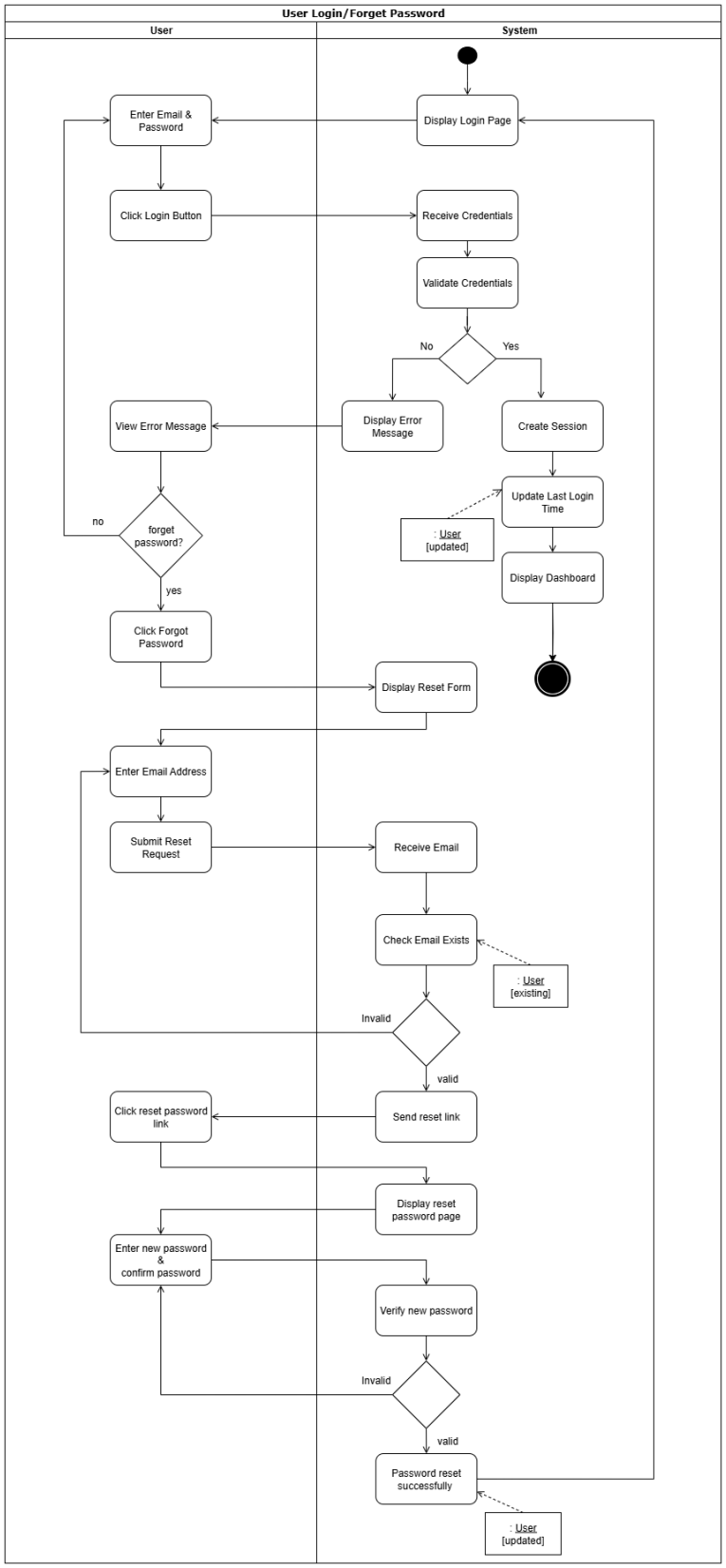

Figure 4.11 Activity Diagram – User Login/Forget Password

The User Login/Forget Password activity diagram demonstrates the dual authentication and password recovery workflows within the system. The primary login path involves credential validation, session creation, and dashboard redirection upon successful authentication.

The forget password branch activates when users cannot remember their credentials, initiating an email-based recovery process that includes reset link generation and secure password update mechanisms.

The diagram showcases decision points for credential validity, account existence checks, and password reset token validation, ensuring comprehensive security measures while maintaining user accessibility and system integrity.

4.6.3 AI Recommendation

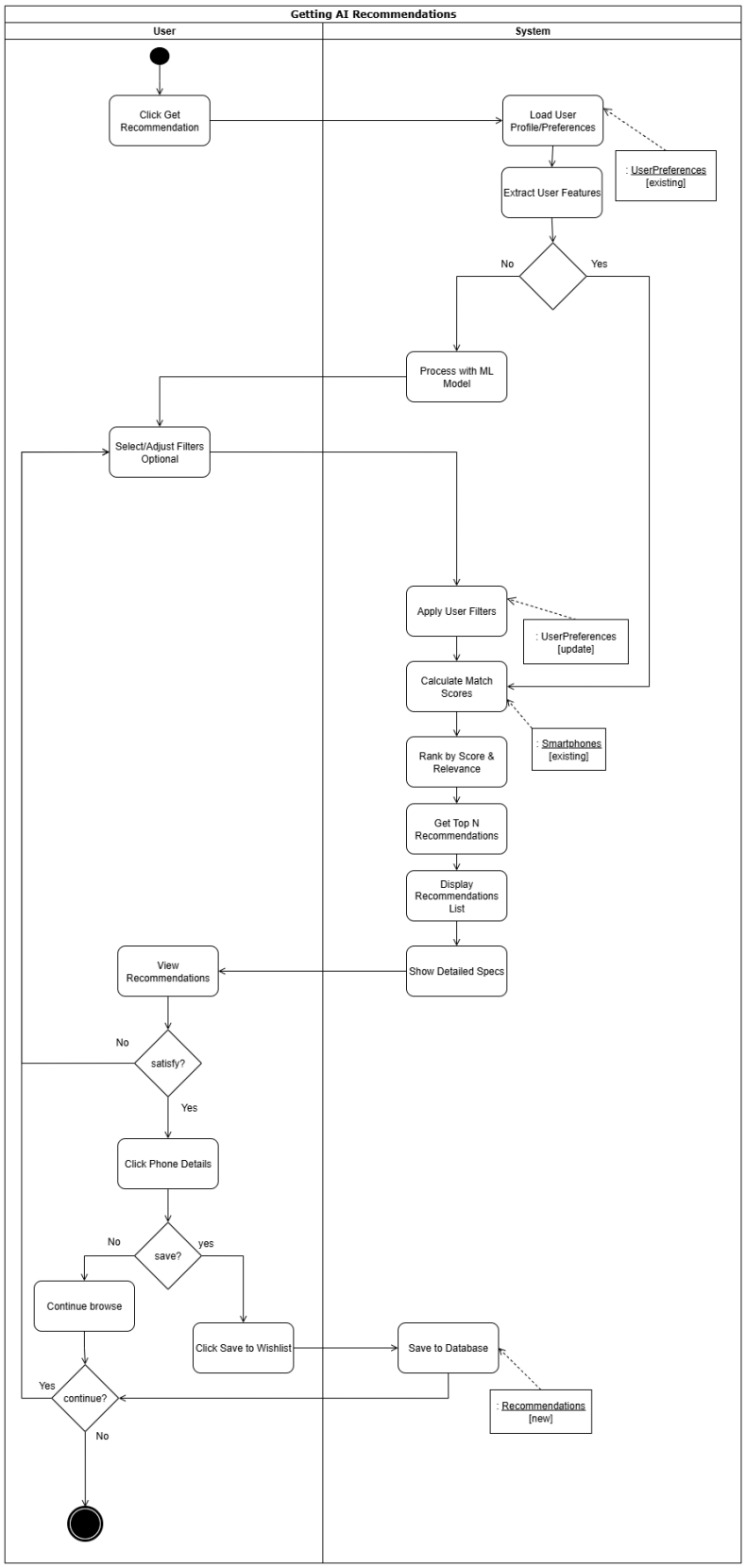

Figure 4.12 Activity Diagram – Getting AI Recommendation

The AI Recommendations activity diagram describes the complex machine learning workflow that generates personalized smartphone recommendations for users. The process starts when a user requests a recommendation, loading their preferences and historical data to be processed thru the ML model.

The system implements a multi-stage recommendation pipeline, including user preference extraction, ML model processing via decision tree and random forest algorithms, and result ranking based on compatibility scores.

The workflow integrates feedback loops for continuous learning and includes decision points such as saving recommendations, viewing detailed specifications, and launching phone comparisons, ensuring that users receive comprehensive guidance throughout the smartphone selection process.

4.6.4 Phone Comparison Module

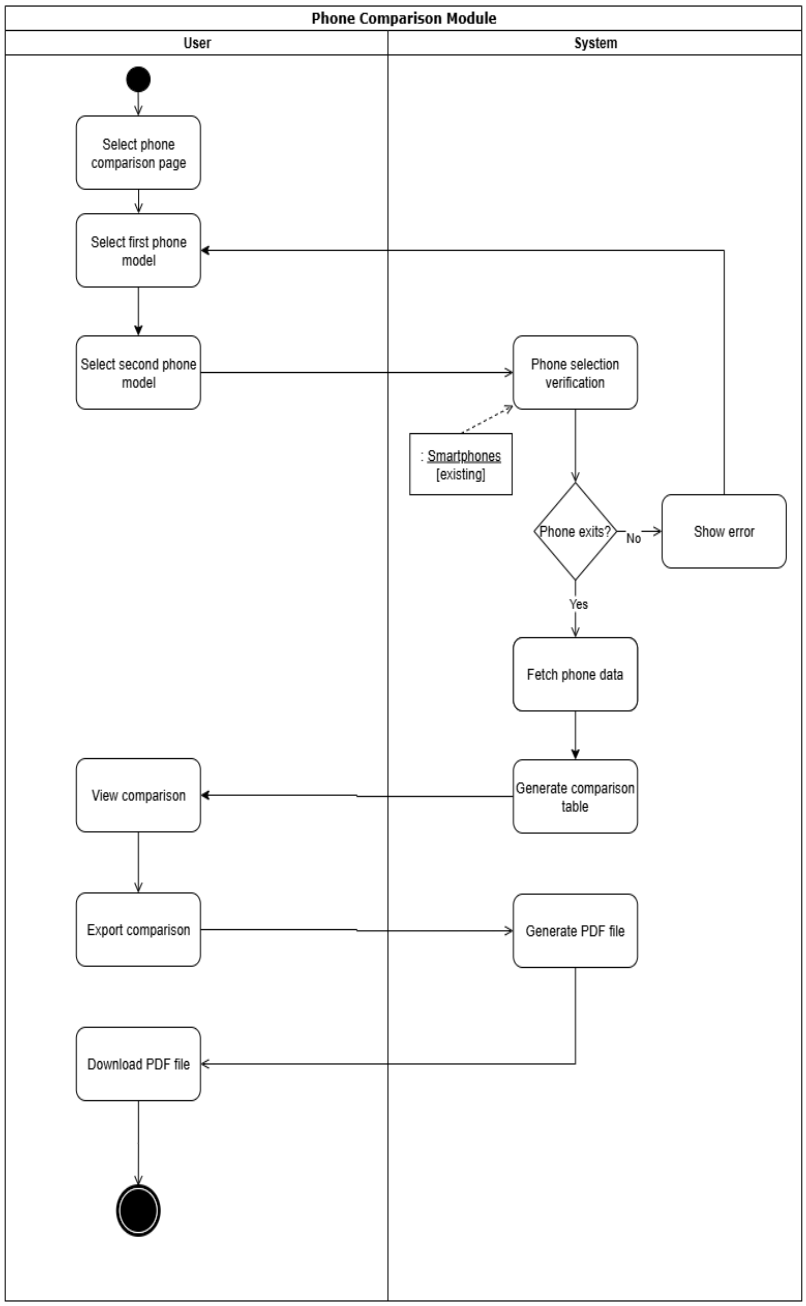

Figure 4.13 Activity Diagram – Phone Comparison

The Phone Comparison activity diagram illustrates the systematic process for side-by-side smartphone analysis within the DialSmart platform.

Users select two phone models through dropdown menus, triggering validation checks to ensure different models are chosen.

The system fetches comprehensive specifications from the database, generates detailed comparison tables highlighting key differences, and presents the analysis in an accessible format.

The workflow includes export functionality for PDF generation and sharing capabilities, enabling users to save their comparison results for future reference or collaborative decision-making processes.

4.6.5 Chatbot Interaction Module

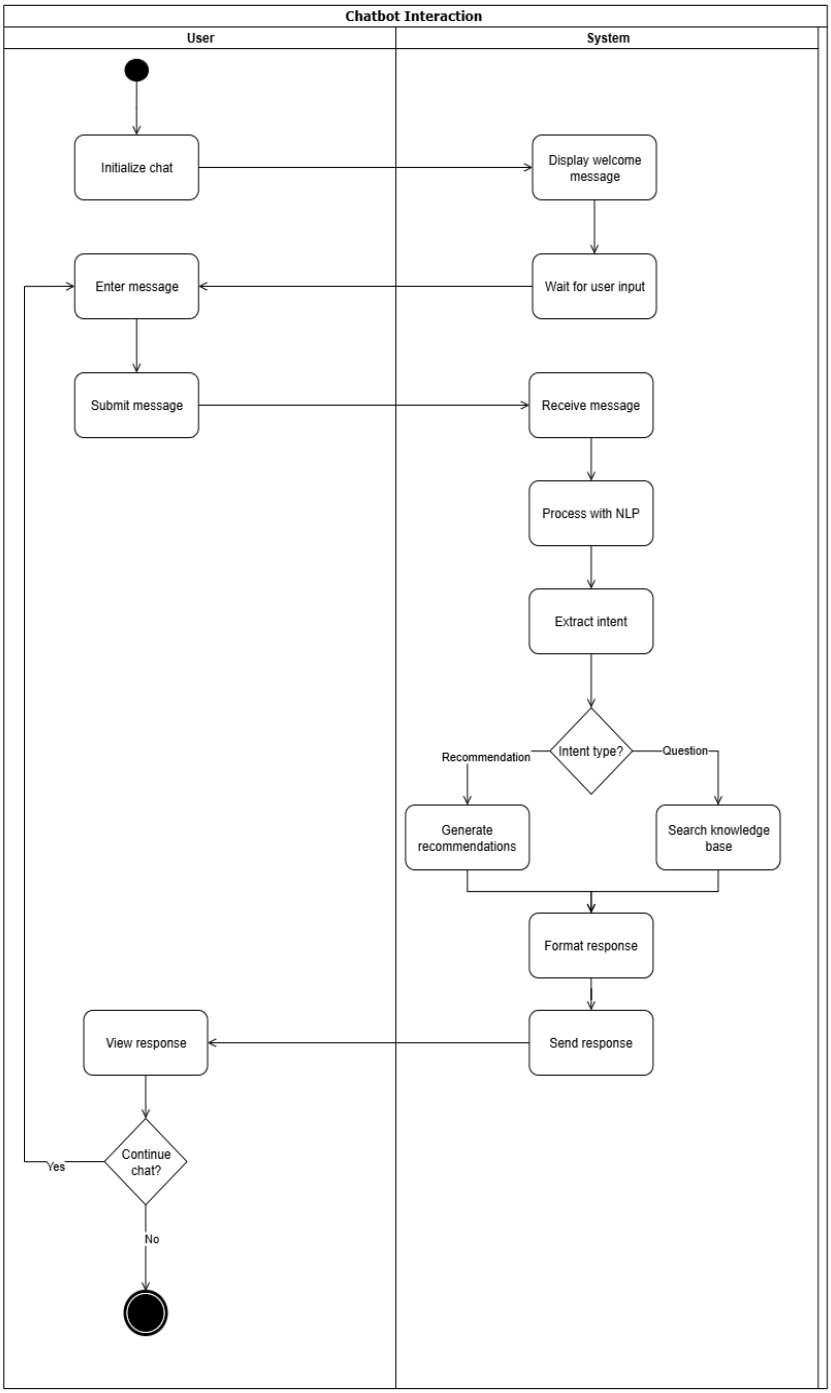

Figure 4.14 Activity Diagram – Chatbot Interaction

The Chatbot Interaction activity diagram represents the conversational AI workflow that enables natural language communication between users and the DialSmart recommendation system.

The process begins with chat initialization and welcome message display, followed by user message input and natural language processing.

The system implements intent extraction and context analysis to determine whether queries require recommendation generation or knowledge base searches.

Response formatting ensures conversational continuity while integrating with the ML recommendation engine for personalized smartphone suggestions delivered through natural dialogue.

4.6.6 Phone Finder & Filter Module

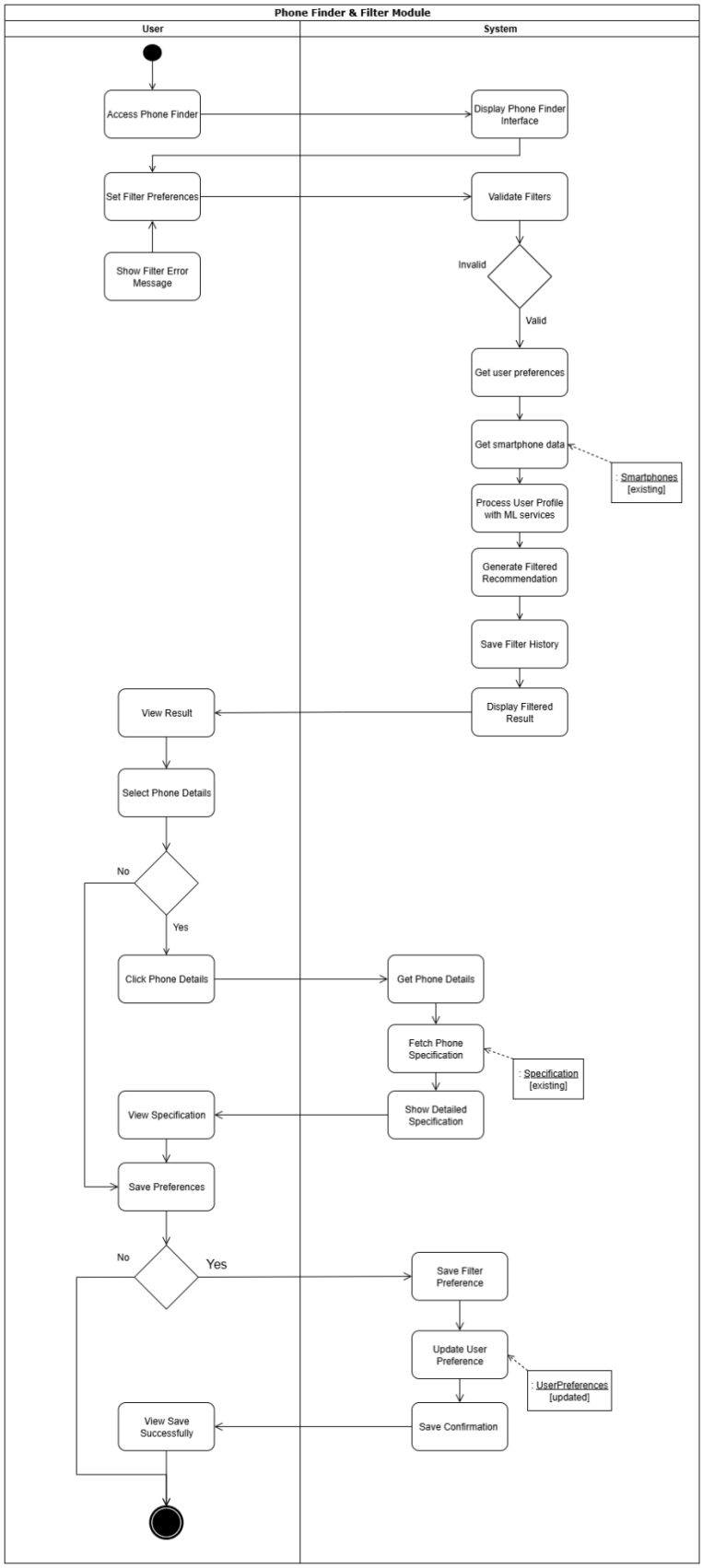

Figure 4.15 Activity Diagram – Phone Finder & Filter

The Phone Finder & Filter activity diagram showcases the advanced search and filtering capabilities that help users navigate the extensive smartphone database.

Users can apply multiple filter criteria including price ranges, brand preferences, usage priorities, and technical specifications.

The system validates filter combinations, processes queries efficiently, and generates filtered recommendations based on user-defined criteria.

The workflow includes preference saving functionality and integrates with detailed phone specification viewing, enabling users to refine their search iteratively until they find suitable smartphone options.

4.6.7 CRUD Module

4.6.7.1 View Phone Operation

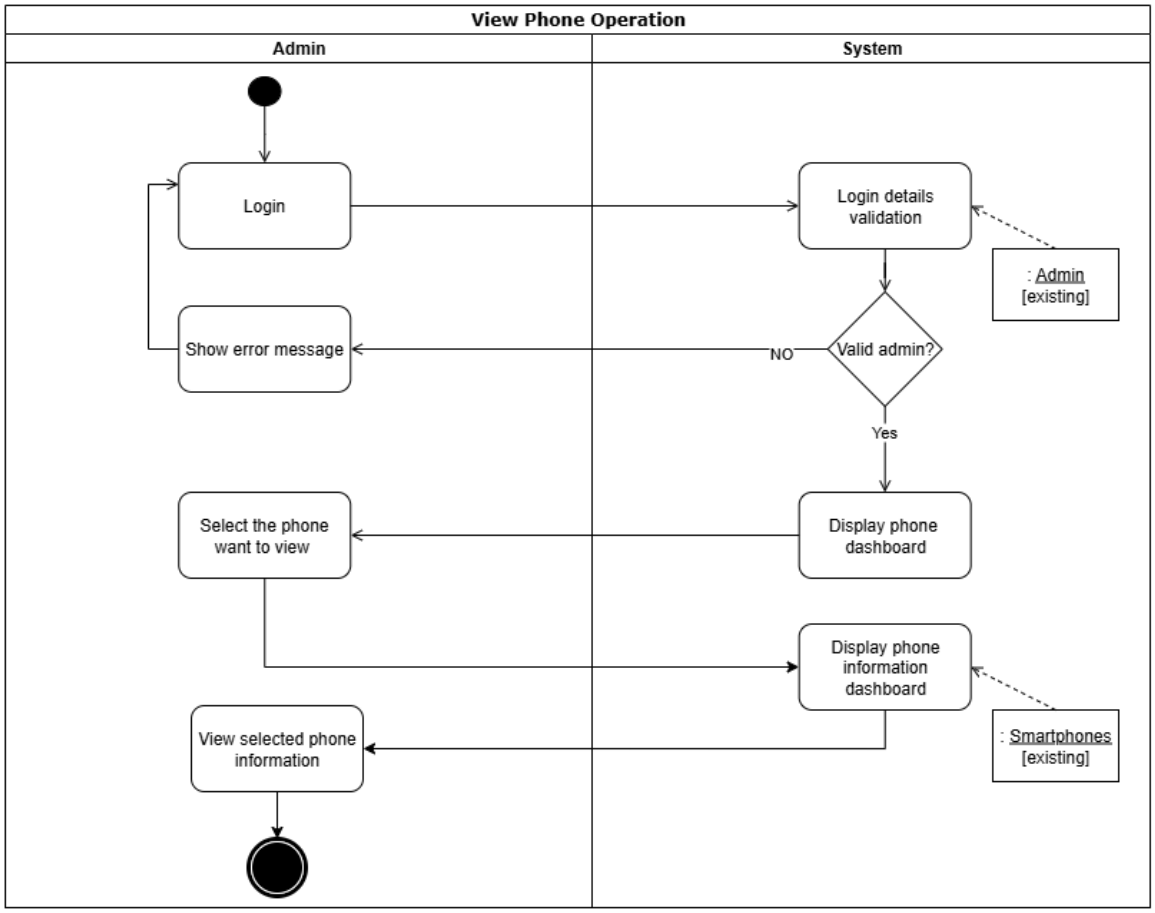

Figure 4.16 Activity Diagram – View Phone Operations

The CRUD (Create, Read, Update, Delete) activity diagrams demonstrate comprehensive administrative functionality for smartphone data management.

The View Phone Operation workflow enables administrators to browse and examine existing phone records with proper authentication and authorization checks.

4.6.7.2 Add Phone Operation

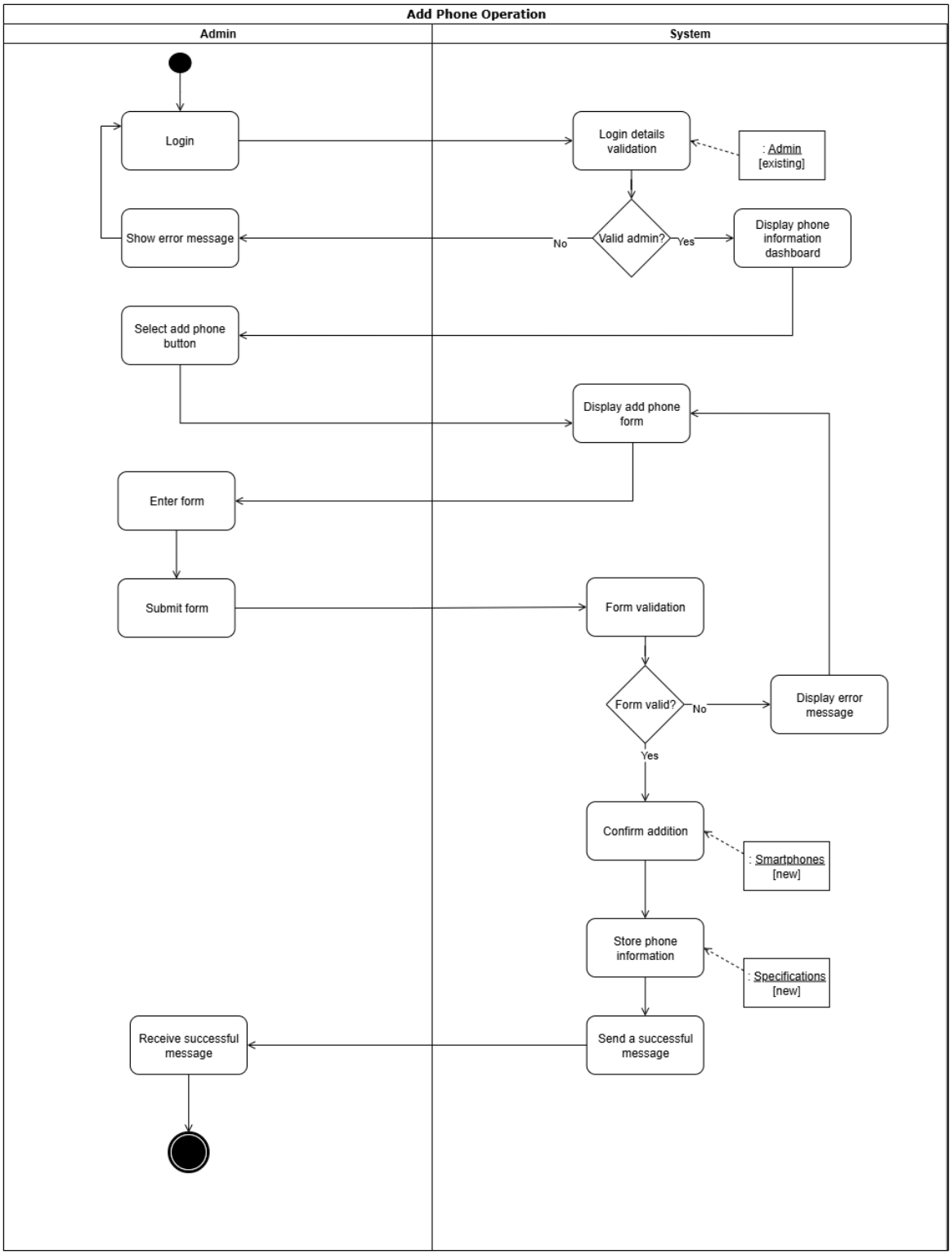

Figure 4.17 Activity Diagram – Add Phone Operations

The Add Phone Operation process includes form validation, data verification, and database insertion with error handling.

4.6.7.3 Update Phone Operation

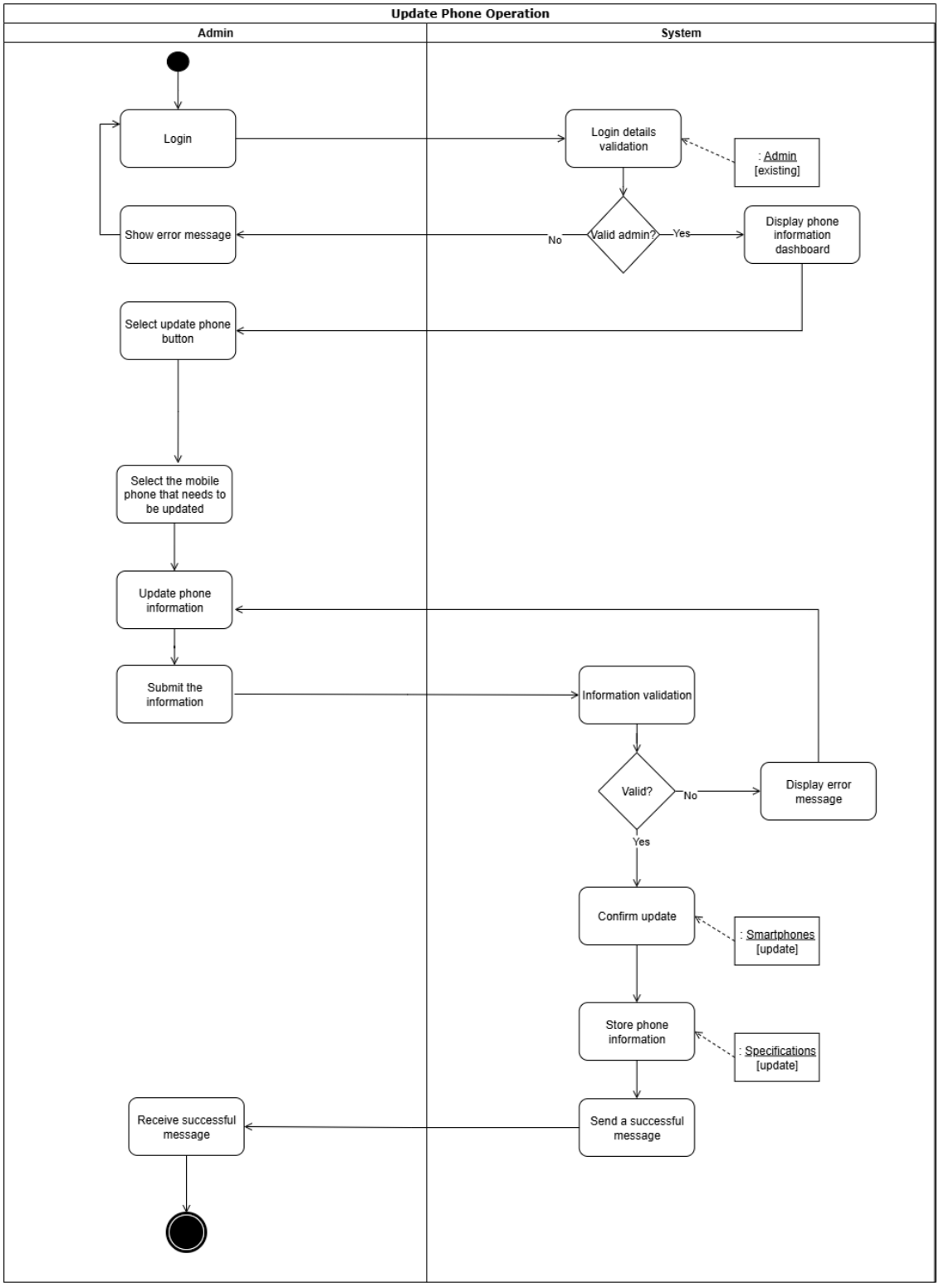

Figure 4.18 Activity Diagram – Update Phone Operations

The Update Phone Operation workflow allows modification of existing records with validation and confirmation steps.

4.6.7.4 Delete Phone Operation

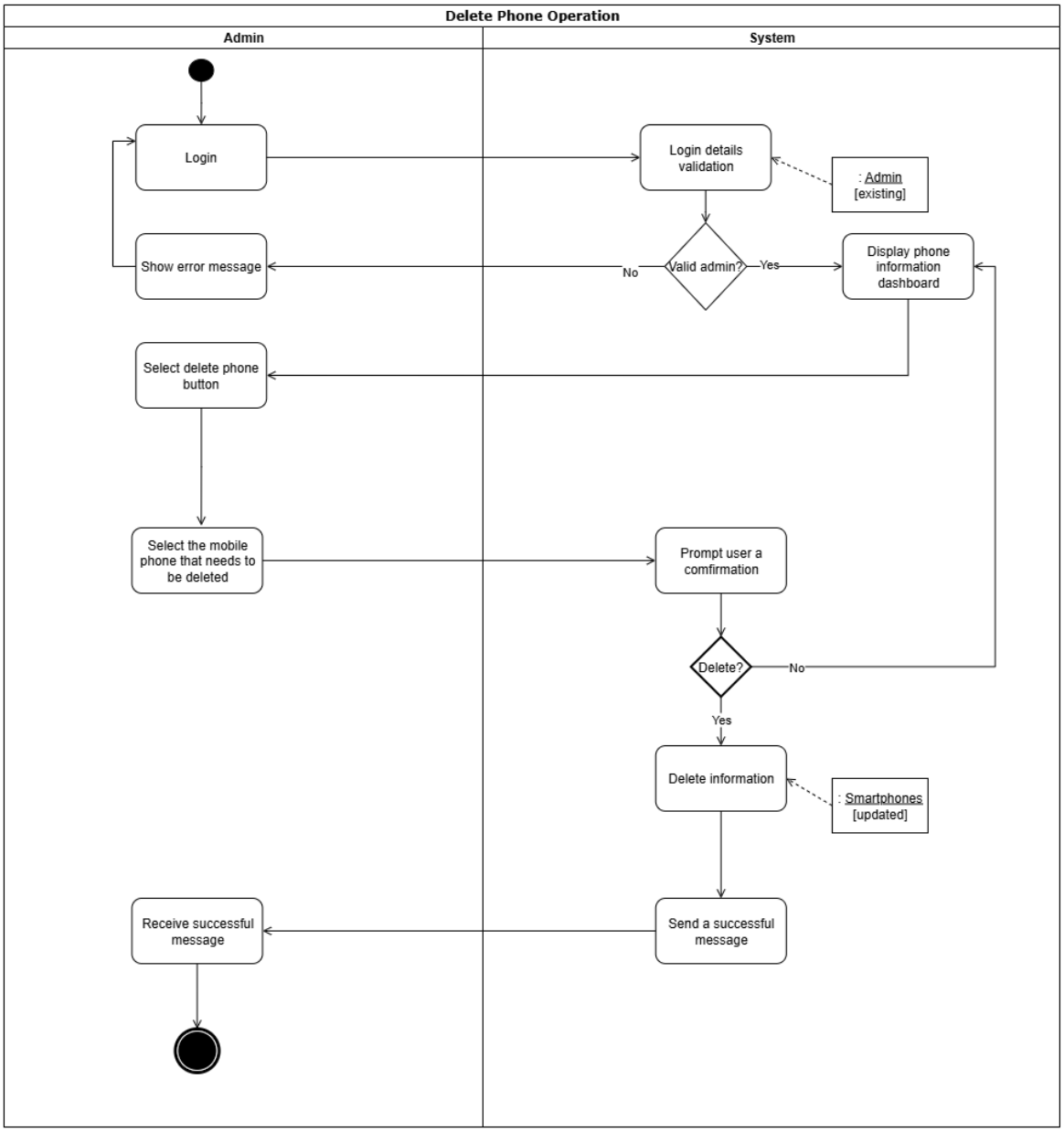

Figure 4.19 Activity Diagram – Delete Phone Operations

The Delete Phone Operation includes confirmation prompts and cascading relationship management to maintain database integrity while removing obsolete records.

4.7 Sequence Diagram

4.7.1 User Registration

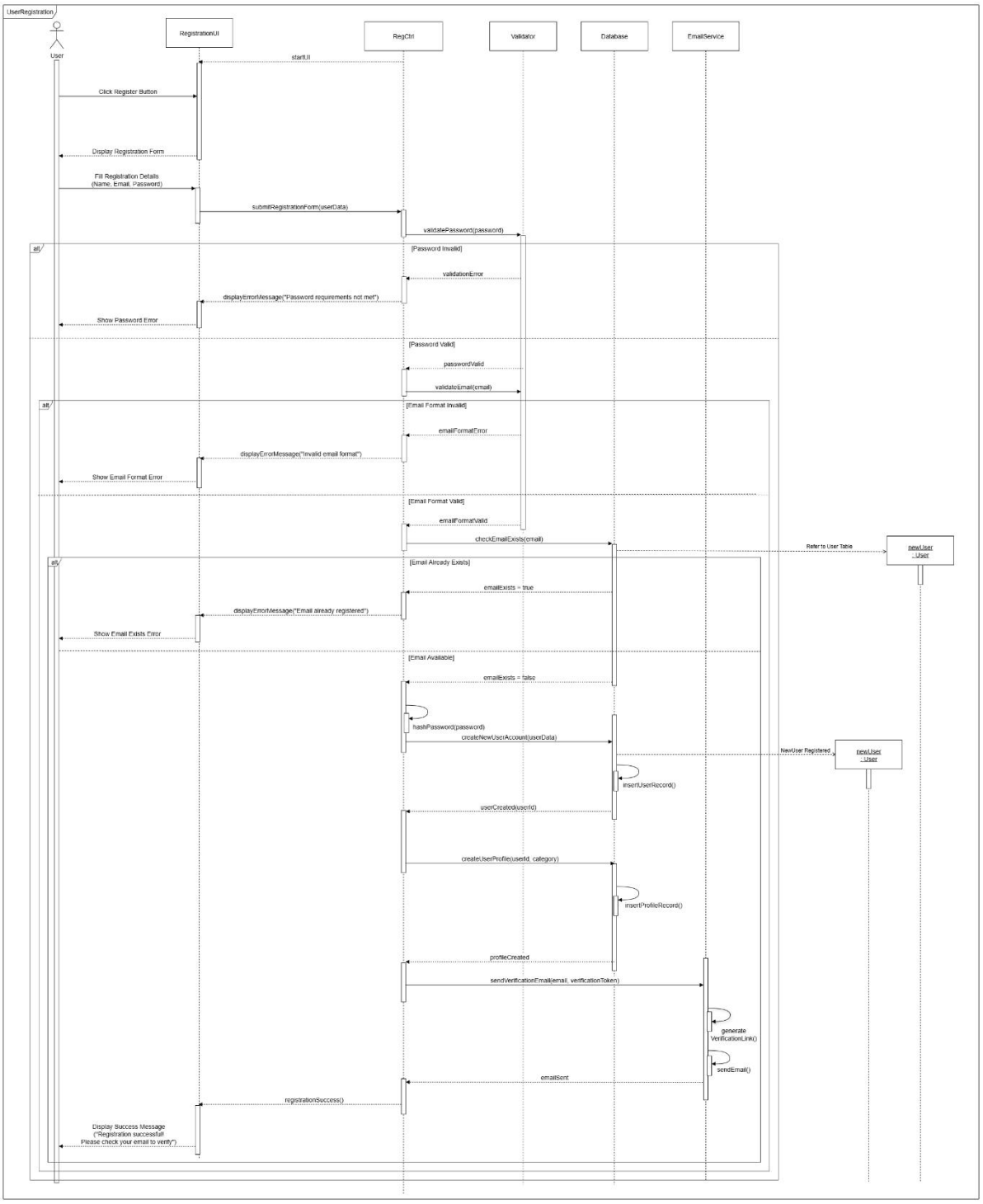

Figure 4.20 Sequence Diagram – User Registration

Component Interaction Flow:

The User Registration sequence demonstrates complex multi-component coordination involving RegistrationUI, RegistrationCTRL, Validator, Database, and EmailService components. The diagram illustrates how user account creation requires systematic validation, database operations, and email verification processes to ensure secure and reliable user onboarding.

Form Validation and Data Processing:

The sequence begins with user form submission triggering comprehensive validation workflows. The RegistrationCTRL receives form data and coordinates with the Validator component to perform email format checking, password strength validation, and duplicate account detection. This multi-step validation ensures data integrity before any database operations occur, preventing invalid or duplicate registrations from compromising system security.

Database Transaction Management:

Following successful validation, the sequence demonstrates secure database interaction patterns. The system performs email availability checks, creates encrypted password hashes, and executes user account creation transactions. The database operations include proper error handling and rollback mechanisms to maintain data consistency during the registration process.

Email Verification Integration:

The registration sequence incorporates email verification through the EmailService component, sending verification tokens and links to confirm user email addresses. This asynchronous email process ensures account security while allowing users to complete registration efficiently. The sequence handles both successful email delivery and potential email service failures gracefully.

4.7.2 AI Recommendation

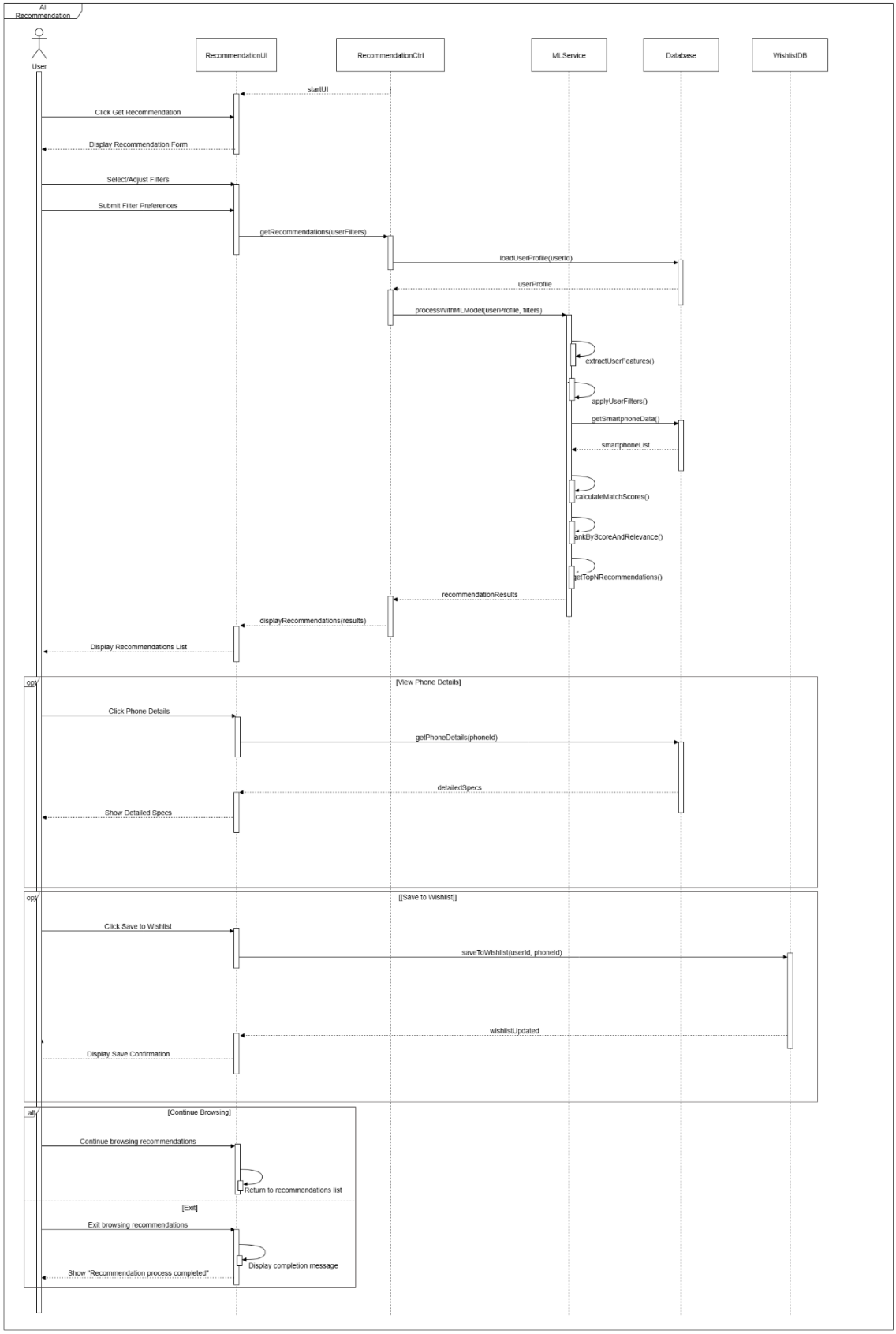

Figure 4.21 Sequence Diagram – AI Recommendation

Machine Learning Integration Architecture:

The AI Recommendation sequence illustrates sophisticated integration between the recommendation interface, ML Service, Database, and ResultsDB components. The diagram demonstrates how user preferences are processed through machine learning algorithms to generate personalized smartphone recommendations with scoring and ranking capabilities.

User Preference Processing Workflow:

The sequence begins with user preference collection and validation through the RecommendationUI component. The RecommendationCTRL coordinates preference extraction, user history analysis, and demographic categorization to create comprehensive user profiles. This data preprocessing ensures the ML model receives properly formatted and complete user information for accurate recommendation generation.

ML Model Execution and Scoring:

The core recommendation generation involves complex ML Service interactions, including model loading, feature processing, and prediction generation. The sequence shows how user preferences are transformed into feature vectors, processed through trained Decision Tree and Random Forest algorithms, and converted into ranked smartphone recommendations with confidence scores and explanation rationales.

Result Formatting and Presentation:

Following ML processing, the sequence demonstrates result formatting and presentation workflows. The system retrieves detailed phone specifications from the Database, combines them with ML scores, and formats comprehensive recommendation lists. The RecommendationUI receives structured recommendation data including phone details, matching percentages, and reasoning explanations for user understanding.

4.7.3 Phone Comparison

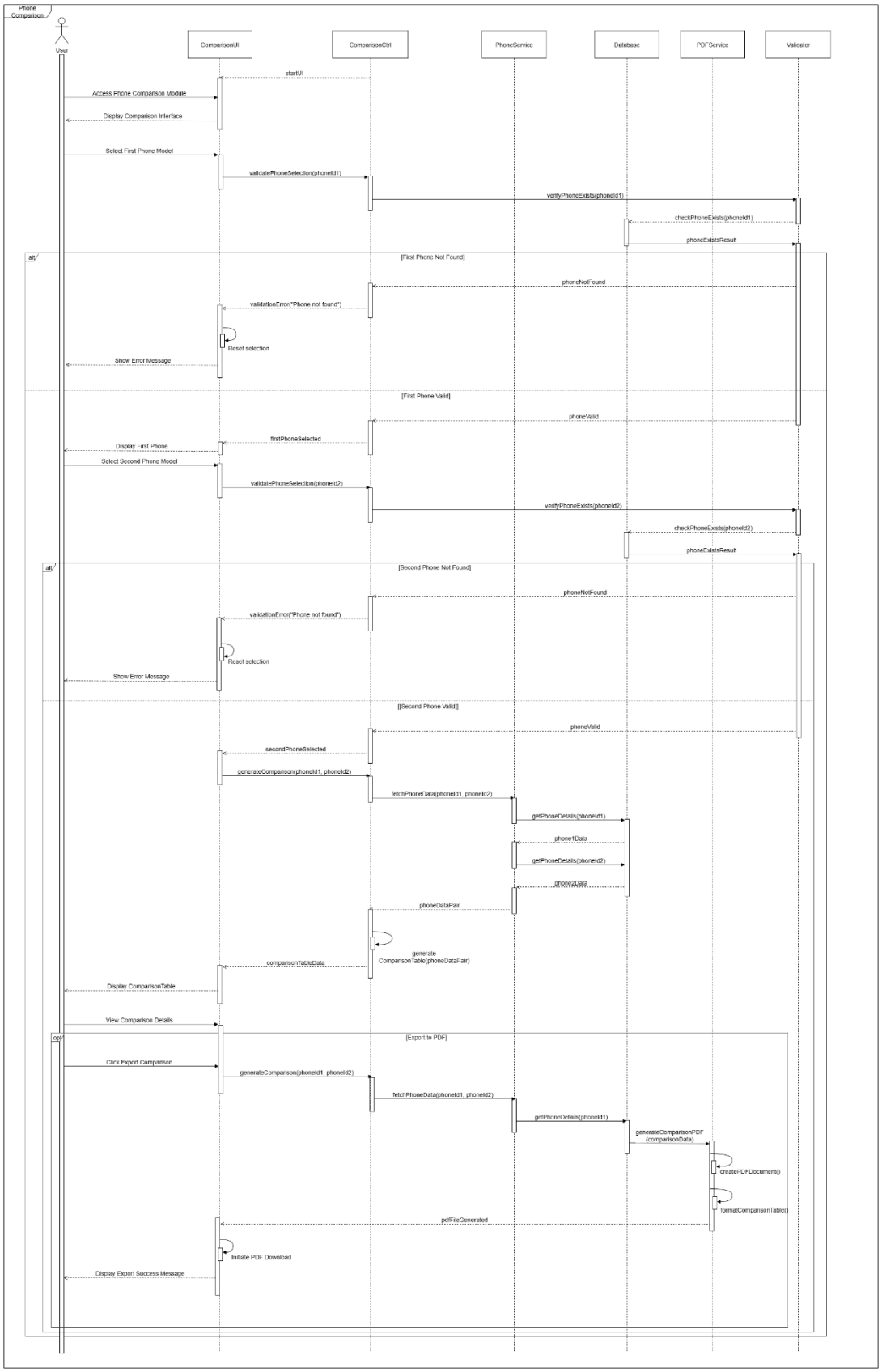

Figure 4.22 Sequence Diagram – Phone Comparison

Multi-Component Comparison Architecture:

The Phone Comparison sequence demonstrates systematic coordination between ComparisonUI, ComparisonCTRL, PhoneService, Database, and PDFService components. The diagram illustrates how two-phone comparison requires comprehensive data retrieval, specification analysis, and formatted presentation generation for effective user decision-making support.

Phone Selection and Validation Process:

The sequence initiates with user phone selection validation through the ComparisonUI component. The ComparisonCTRL coordinates phone availability checking, specification retrieval, and comparison feasibility analysis. This validation ensures both selected phones have complete specification data and are suitable for meaningful comparison analysis.

Specification Retrieval and Analysis:

The core comparison process involves detailed specification extraction from the Database through the PhoneService component. The sequence demonstrates how technical specifications, pricing information, feature lists, and performance metrics are systematically retrieved and organized for side-by-side analysis. The system handles missing specifications gracefully while maintaining comparison accuracy.

Comparison Table Generation and Export:

The final phase involves comparison table formatting and PDF generation through the PDFService component. The sequence shows how specification data is transformed into user-friendly comparison formats, highlighting differences and similarities between selected phones. The export functionality enables users to save and share comparison results for future reference and collaborative decision-making.

4.7.4 Chatbot Interaction

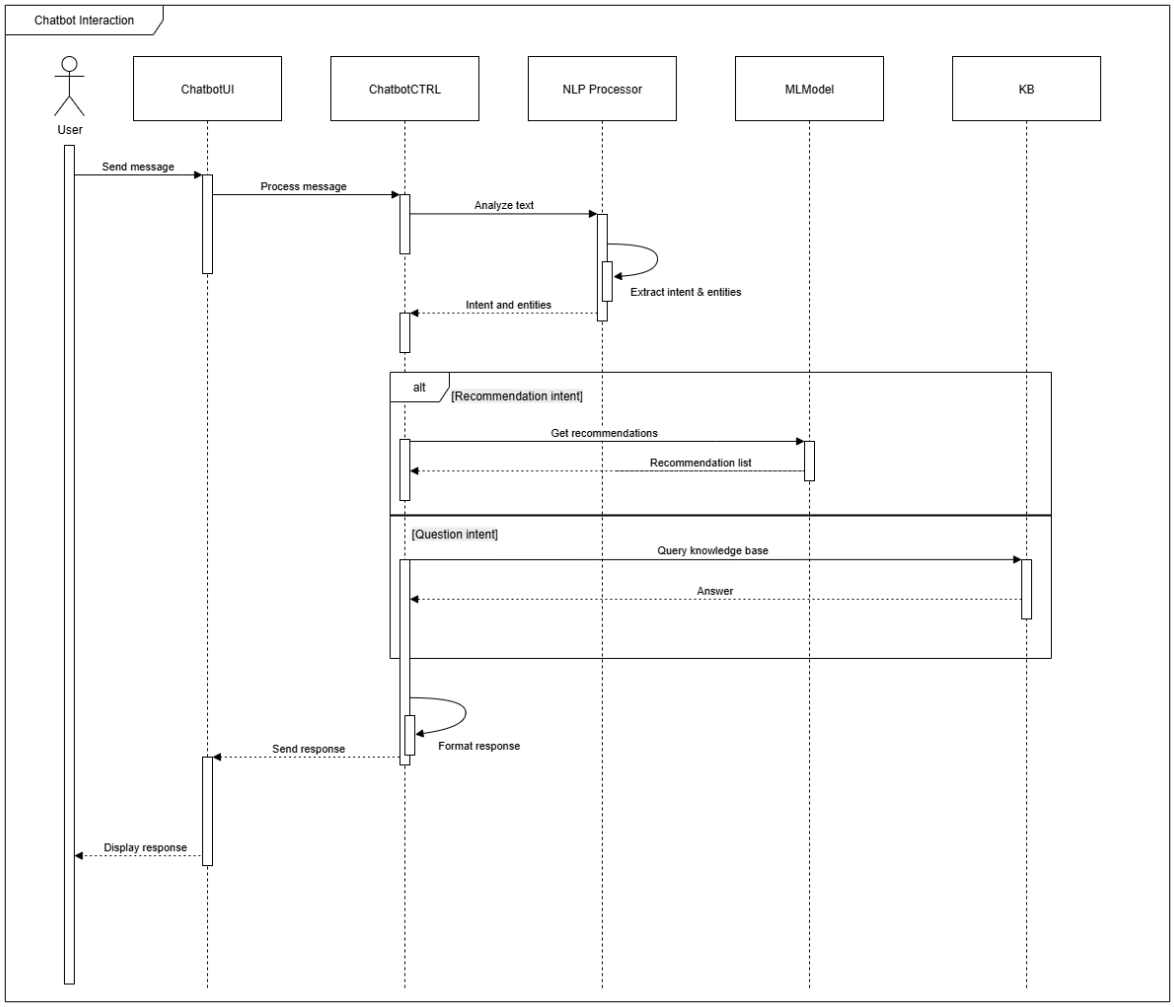

Figure 4.23 Sequence Diagram – Chatbot Interaction

Natural Language Processing Architecture: The Chatbot Interaction sequence demonstrates sophisticated integration between ChatbotUI, ChatbotCTRL, NLP Processor, ML Model, and Knowledge Base (KB) components. The diagram illustrates how conversational AI processes user queries through natural language understanding, intent extraction, and context-aware response generation specifically tailored for smartphone advisory interactions.

Message Processing and Intent Analysis: The sequence begins with user message submission triggering comprehensive NLP processing workflows. The ChatbotCTRL receives user input and coordinates with the NLP Processor to perform text analysis, intent detection, and entity extraction. This multi-layered analysis identifies whether users are seeking recommendations, asking questions, or requesting specific information, enabling contextually appropriate response generation.

ML Model Integration and Recommendation Generation: Following intent analysis, the sequence demonstrates dynamic ML Model integration for recommendation requests. The system processes user preferences extracted from conversational context, queries the ML Model for personalized suggestions, and retrieves recommendation lists with scoring rationales. This integration enables the chatbot to provide AI-powered smartphone recommendations through natural dialogue rather than structured forms.

Knowledge Base Query and Response Formatting: The sequence incorporates Knowledge Base interactions for general queries and FAQ responses. The ChatbotCTRL coordinates knowledge retrieval, answer formatting, and response personalization based on user context and conversation history. The system handles both recommendation scenarios and informational queries while maintaining conversational flow and context awareness throughout extended interactions.

4.7.5 Phone Finder/Filter

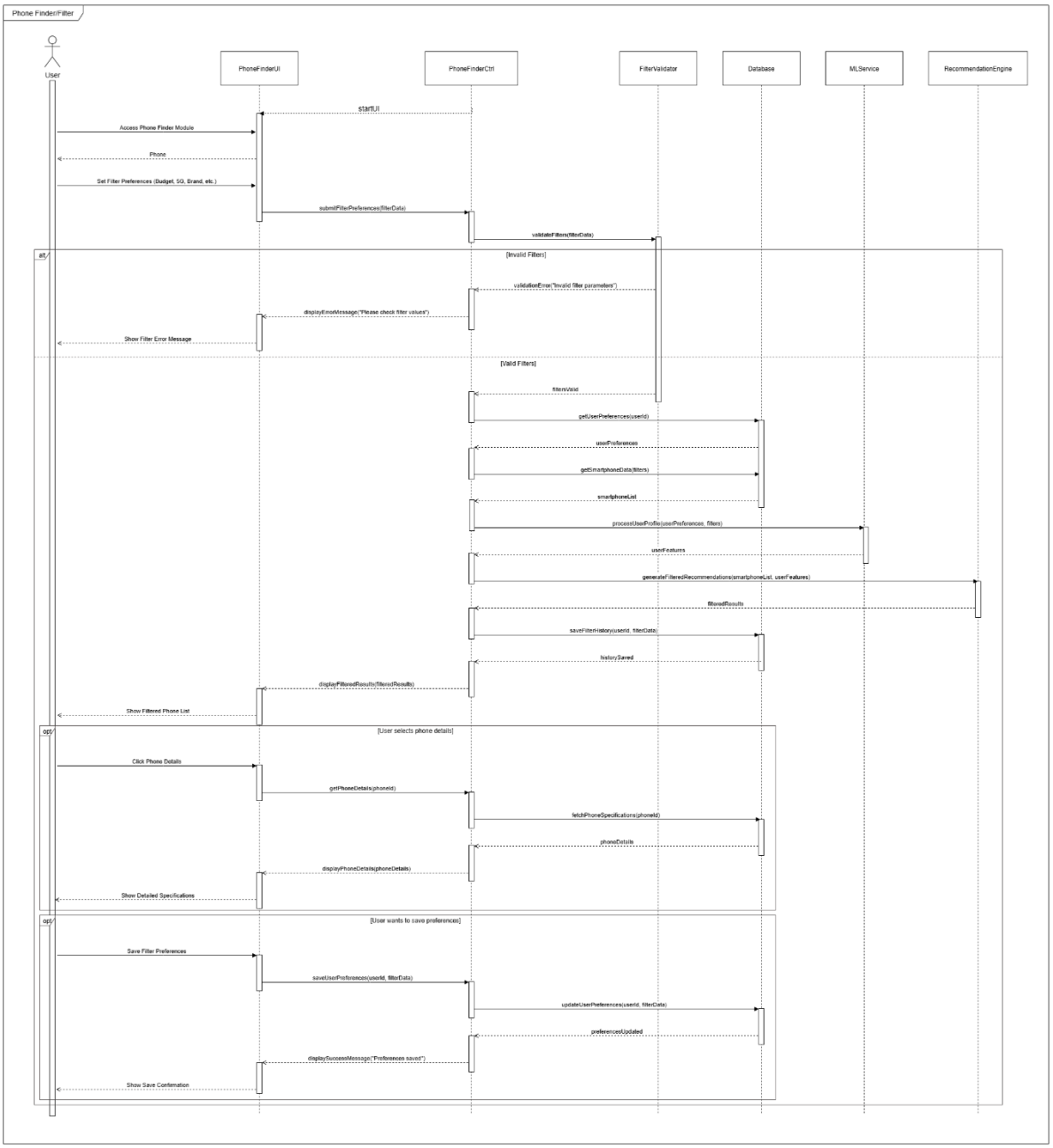

Figure 4.24 Sequence Diagram – Phone Finder/Filter

Multi-Criteria Search Architecture:

The Phone Finder/Filter sequence illustrates comprehensive coordination between PhoneFinderUI, PhoneController, FilterService, Database, and RecommendationEngine components. The diagram demonstrates how complex filtering workflows process multiple user criteria simultaneously to generate refined smartphone search results with intelligent ranking and recommendation integration.

Filter Criteria Validation and Processing:

The sequence initiates with user filter specification validation through the PhoneFinderUI component. The PhoneController coordinates criteria processing including price range validation, brand selection verification,

feature requirement analysis, and usage pattern categorization. This systematic validation ensures filter combinations are logically consistent and technically feasible before database query execution.

Database Query Optimization and Result Generation:

The core filtering process involves sophisticated database query construction through the FilterService component. The sequence shows how multiple filter criteria are translated into optimized database queries, executed against smartphone specifications, and processed to generate initial result sets. The system handles complex query combinations while maintaining response performance and result accuracy.

Recommendation Engine Integration and Result Ranking:

The final phase demonstrates RecommendationEngine integration for intelligent result ranking and personalization. The sequence illustrates how filtered results are enhanced with ML-based scoring, user preference matching, and recommendation rationales. The PhoneFinderUI receives comprehensive result sets including phone specifications, filtering match indicators, recommendation scores, and suggested alternatives, enabling users to make informed decisions based on their specific criteria and preferences.

4.7.6 CRUD Module

4.7.6.1 View Phone Operation

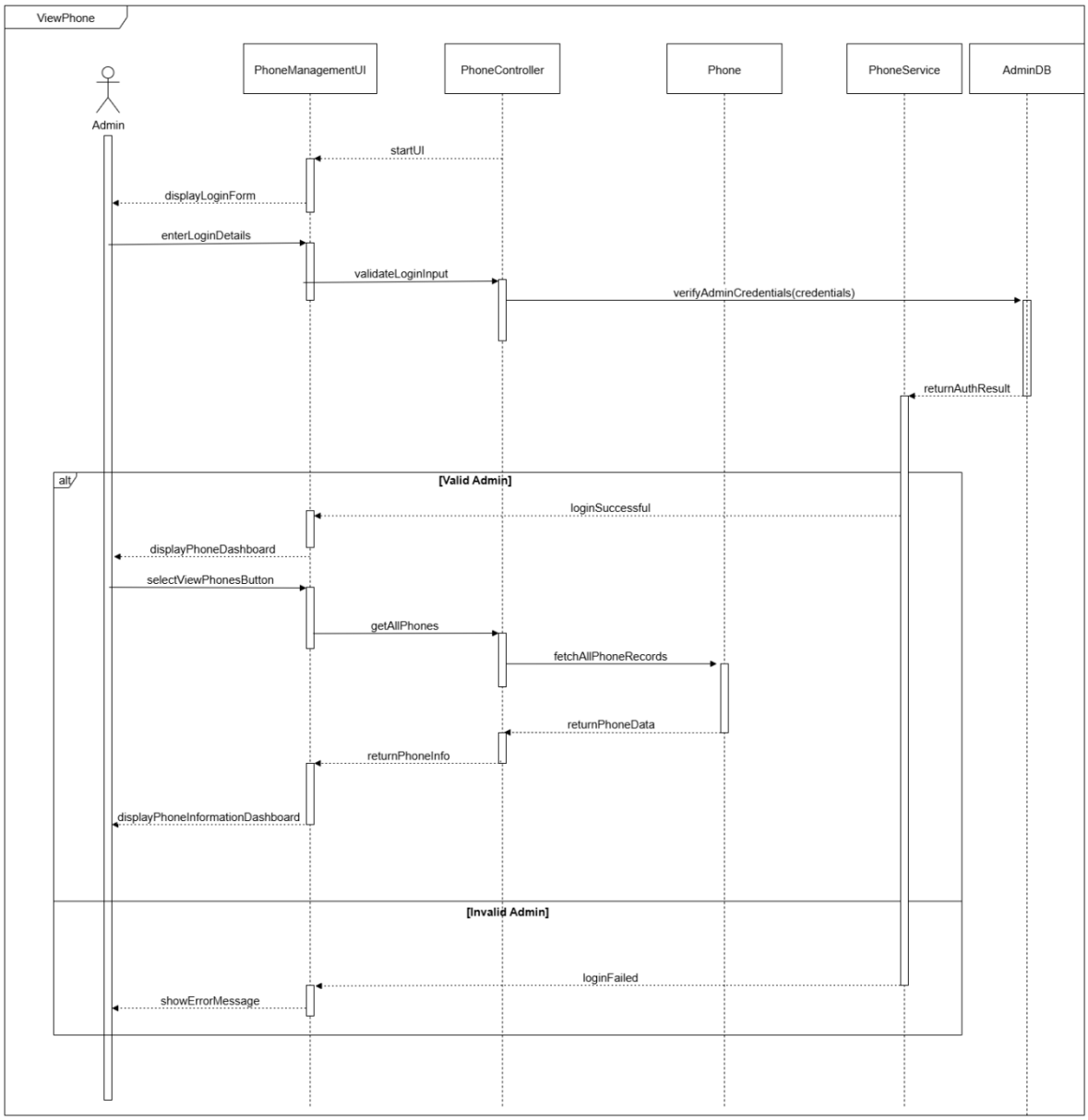

Figure 4.25 Sequence Diagram – View Phone Operation

The CRUD sequence diagrams demonstrate administrative data management workflows through systematic component interactions.

The View Phone Operation sequence shows read-only access patterns with proper authentication and data retrieval.

4.7.6.2 Add Phone Operation

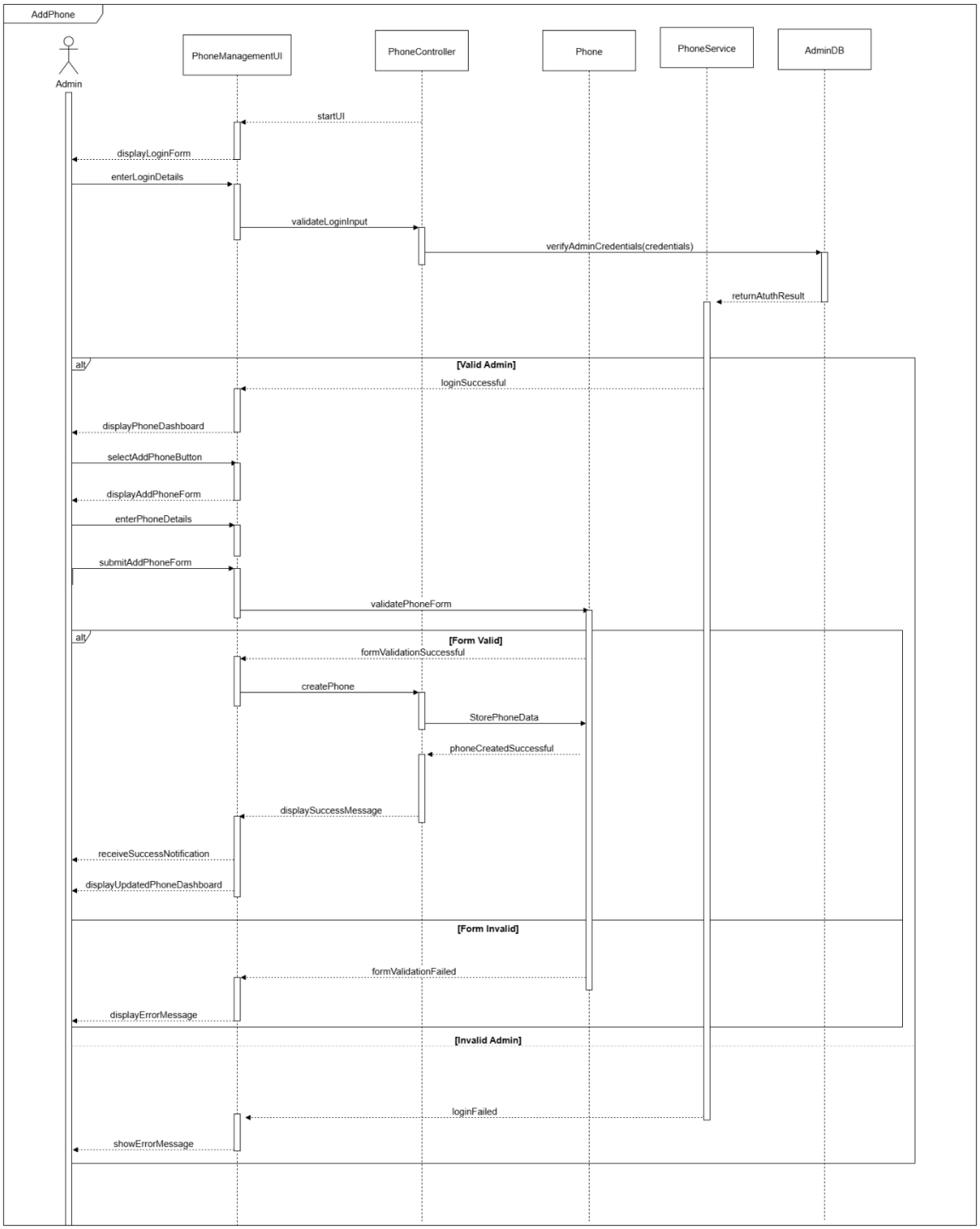

Figure 4.26 Sequence Diagram – Add Phone Operation

The Add Phone Operation sequence illustrates creation workflows with validation, verification, and database insertion.

4.7.6.3 Update Phone Operation

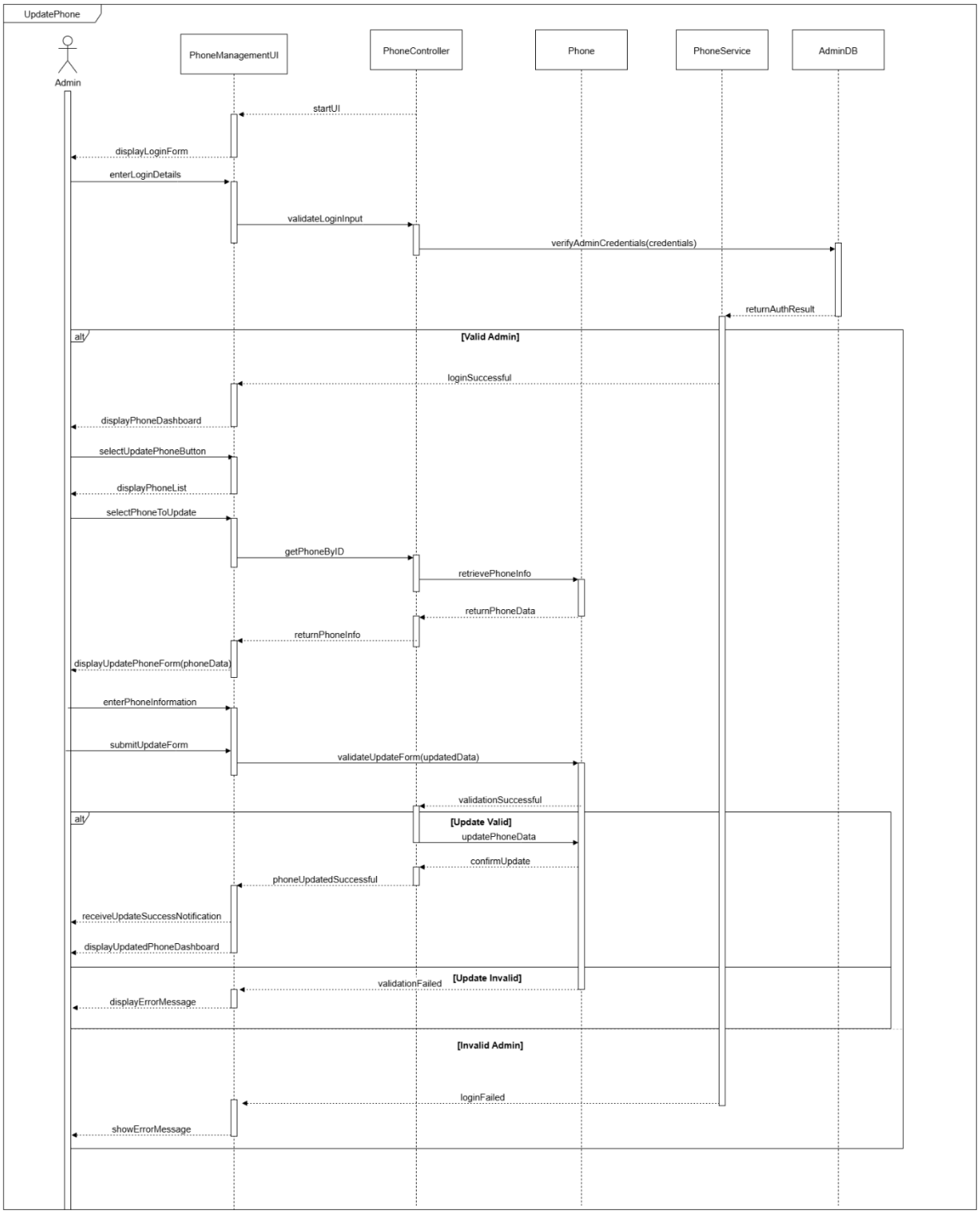

Figure 4.27 Sequence Diagram – Update Phone Operation

The Update Phone Operation sequence demonstrates modification patterns with existing data loading, change validation, and update confirmation.

4.7.6.4 Delete Phone Operation

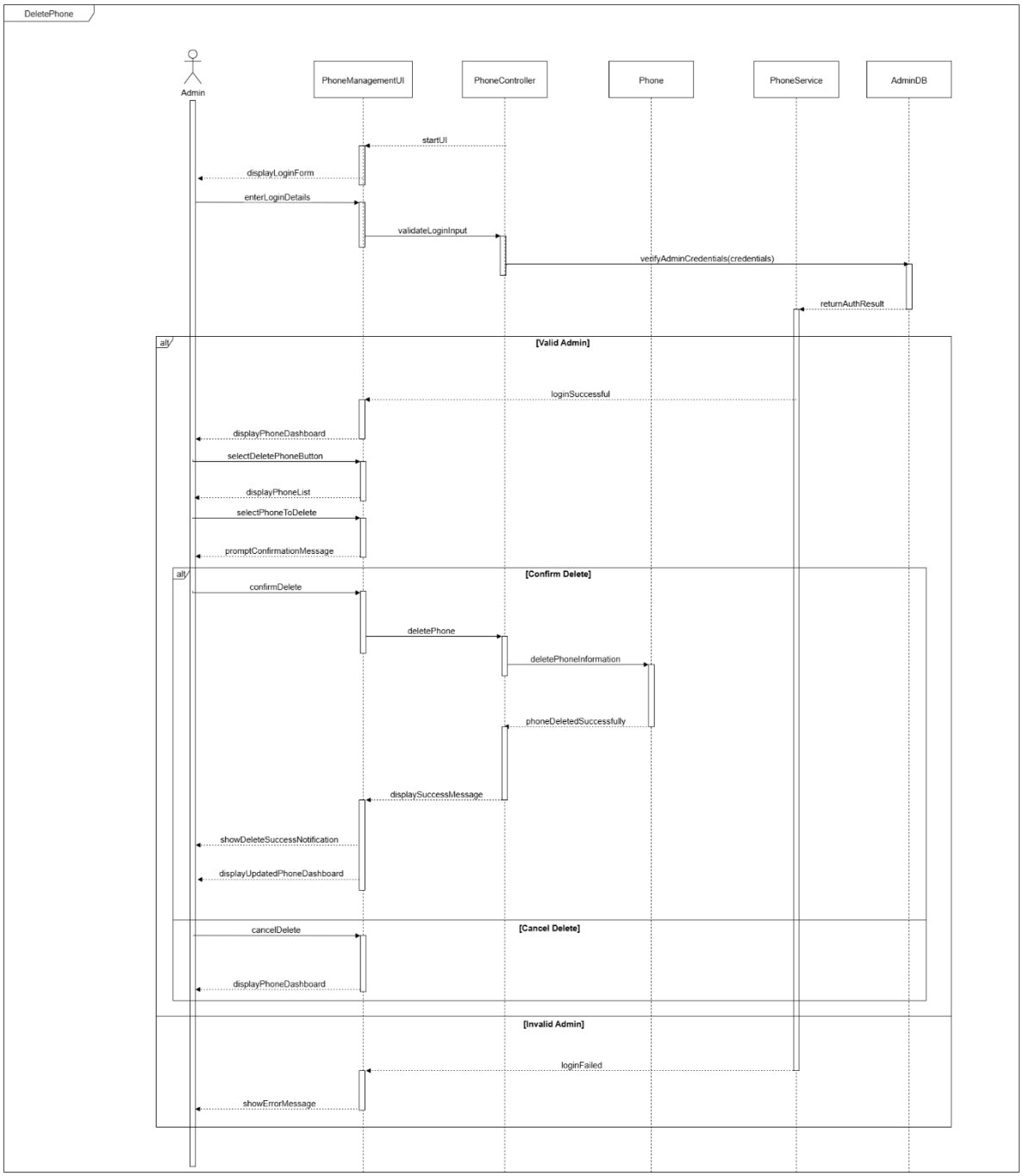

Figure 4.28 Sequence Diagram – Delete Phone Operation

The Delete Phone Operation sequence shows removal workflows with confirmation prompts, relationship checking, and cascading deletion management to maintain database consistency.

4.8 Layered Software Architecture Diagram

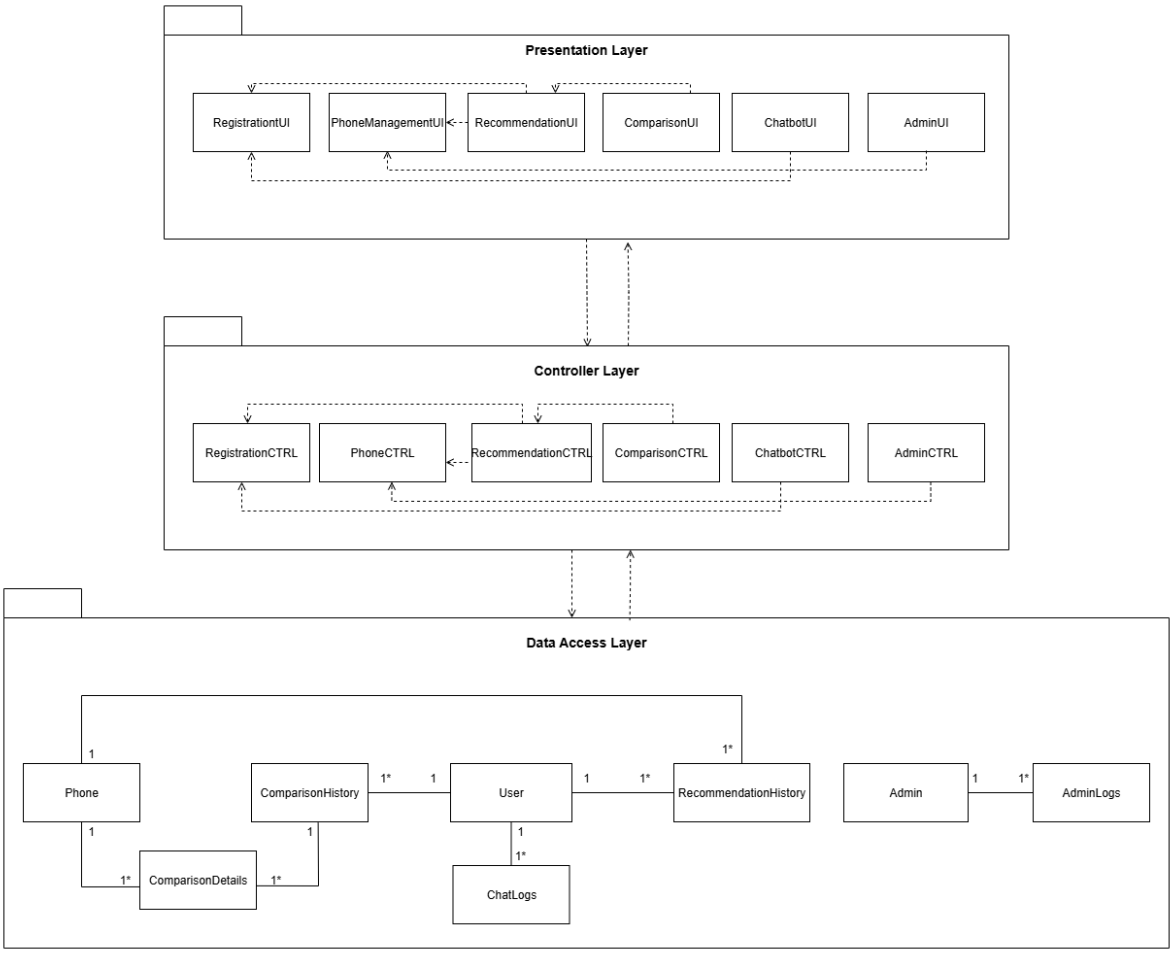

Figure 4.29 Layered Software Architecture Diagram

The Layered Software Architecture diagram presents the systematic organization of DialSmart's technical components across **three distinct architectural tiers** that ensure **separation of concerns**, **maintainability**, and **scalability**. The **Presentation Layer** encompasses all user-facing components including the Registration UI, Phone Management interface, Recommendation display, Comparison tools, Chatbot interface, and Admin panel, providing **intuitive access points for different user roles** while maintaining **consistent user experience standards** across all functional modules.

The **Controller Layer** serves as the **critical intermediary** that orchestrates **business logic** and manages **communication between the presentation and data layers**. This tier includes specialized controllers for Registration, Phone management, Recommendation processing, Comparison generation, Chatbot interactions, and Administrative functions, each implementing **specific business rules** while maintaining **clear separation between user interface concerns and data management operations**. The controller components handle **request validation**, **business logic execution**, and **response formatting** while ensuring **security protocols** and **access control measures** are consistently applied.

The **Data Access Layer** provides **robust data management capabilities** through carefully designed **database entities and relationships** that support the system's comprehensive functionality. The **Phone entity serves as the central data structure** with relationships to Comparison History, User preferences, Recommendation History, Admin management, and Chat logs, ensuring **data integrity** while supporting **complex query requirements**. This architectural approach enables **efficient data retrieval**, supports **scalable performance characteristics**, and facilitates **future system enhancements** while maintaining **clear boundaries between functional responsibilities** across all system components.

4.9 User Interface (UI) Design

4.9.1 Register Page

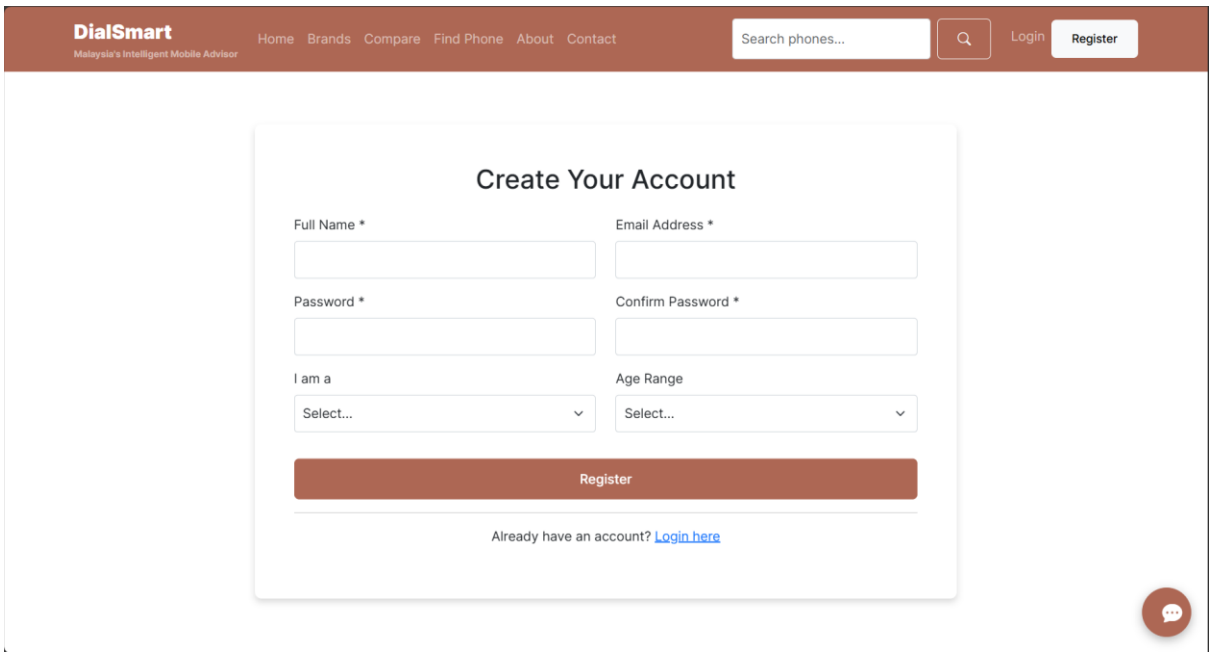

Figure 4.30 Register page

The registration interface enables new users to create accounts by providing essential information including full name, email address, password with confirmation, user category selection (Student/Worker/Elder), and age range. The clean, centered form design ensures straightforward account creation with validation to maintain data integrity.

4.9.2 Forget Password Page

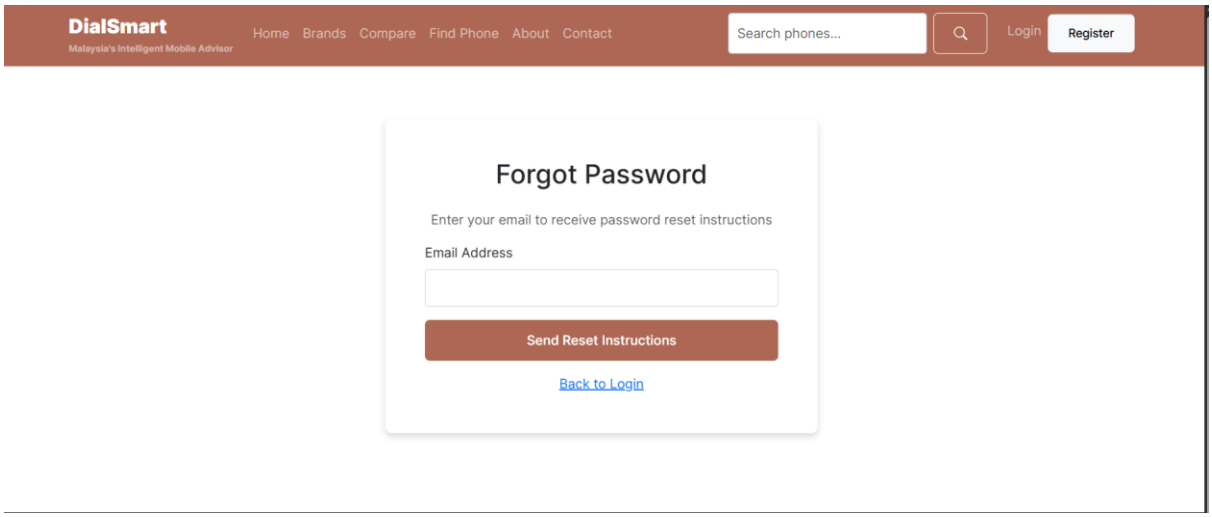

Figure 4.31 Forget Password Page

This password recovery interface allows users who have forgotten their credentials to initiate account recovery by entering their registered email address. The system sends password reset instructions via email, providing a secure mechanism for users to regain access to their accounts without administrative intervention.

4.9.3 Index Page

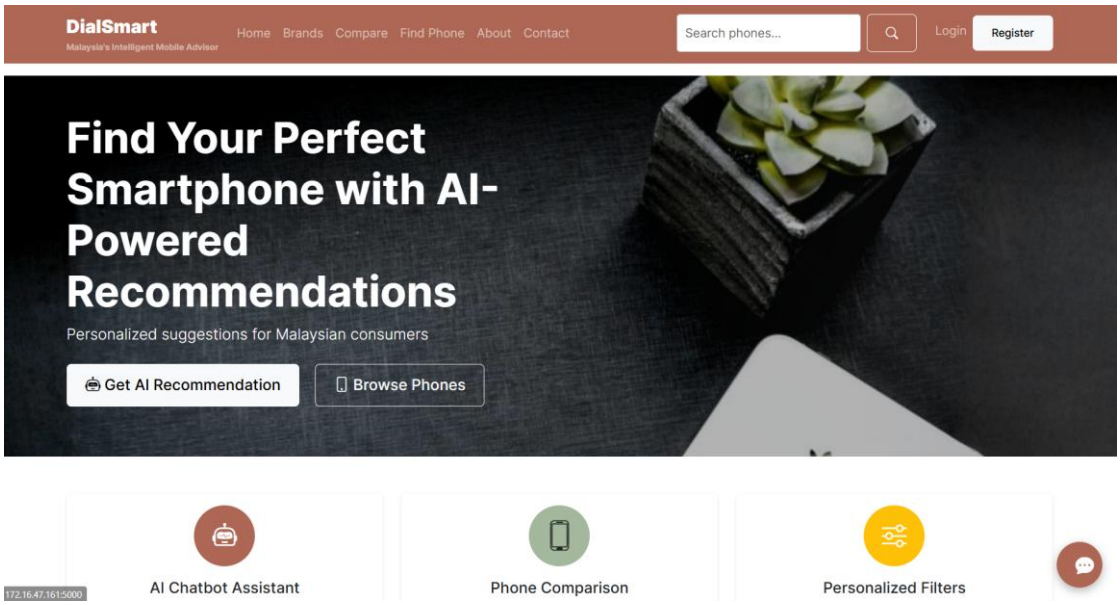

Figure 4.32 Index Page

The homepage serves as the primary entry point, featuring a prominent hero section with the value proposition "Find Your Perfect Smartphone with AI-Powered Recommendations." Users can access core functionalities including AI recommendations, phone comparison tools, and personalized filters, with clear calls-to-action for both registered and guest users.

4.9.3.1 Trusted Partner & Featured Brands - Index Page

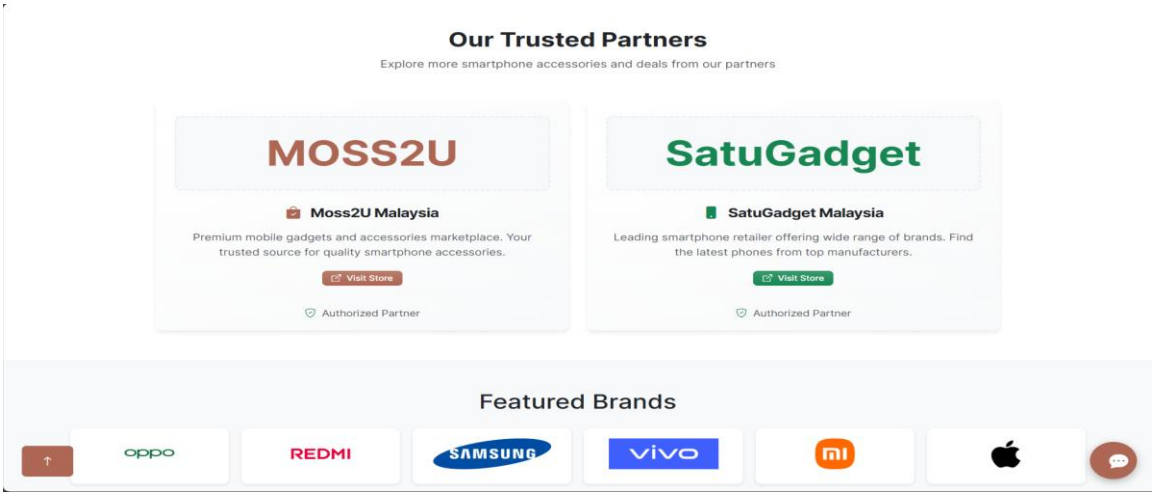

Figure 4.33 Trusted Partner & Featured Brands - Index Page

This section establishes credibility by showcasing partnerships with MOSS2U Malaysia and SatuGadget Malaysia, both authorized retailers for smartphone accessories and devices. The featured brands carousel displays major manufacturers including Oppo, Redmi, Samsung, Vivo, Xiaomi, and Apple, providing quick navigation to brand-specific pages.

4.9.4 Browse Phone Page

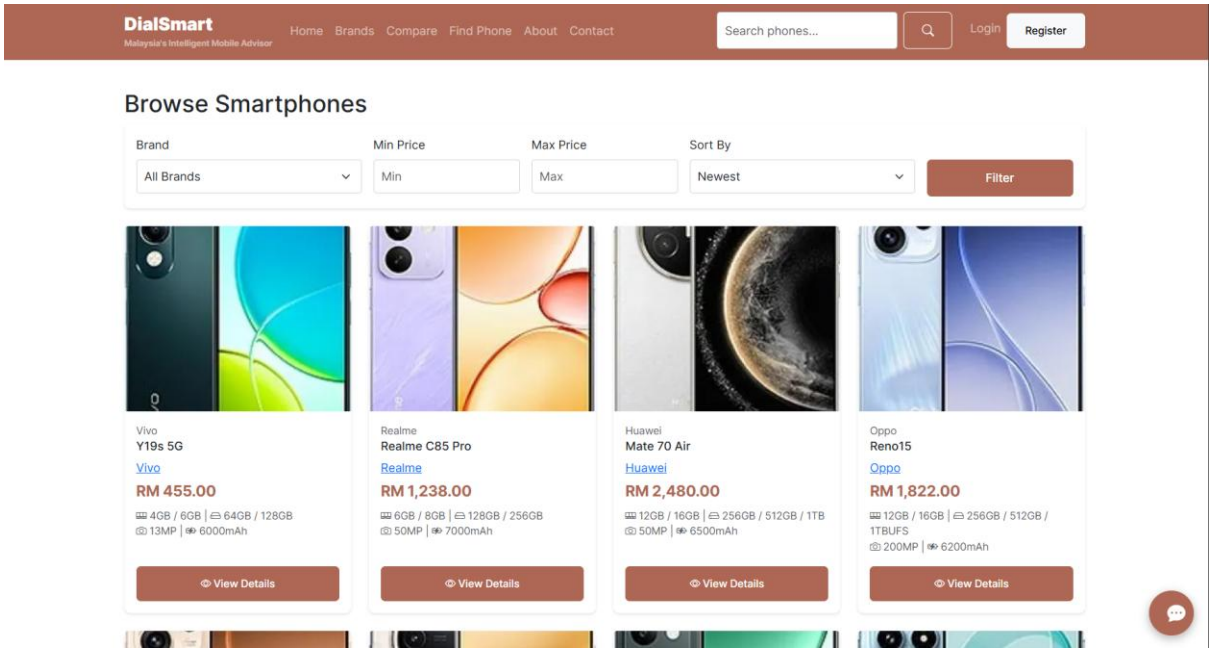

Figure 4.34 Browse Phone Page

The comprehensive phone browsing interface displays smartphones in a grid layout with filtering options for brand selection, price range specification, and sorting by 'Newest', 'Price', and 'Name'. Each phone card shows the device image, model name, brand, pricing in MYR, key specifications (RAM, storage, camera), and a "View Details" button for deeper exploration.

4.9.5 Find Phone Page

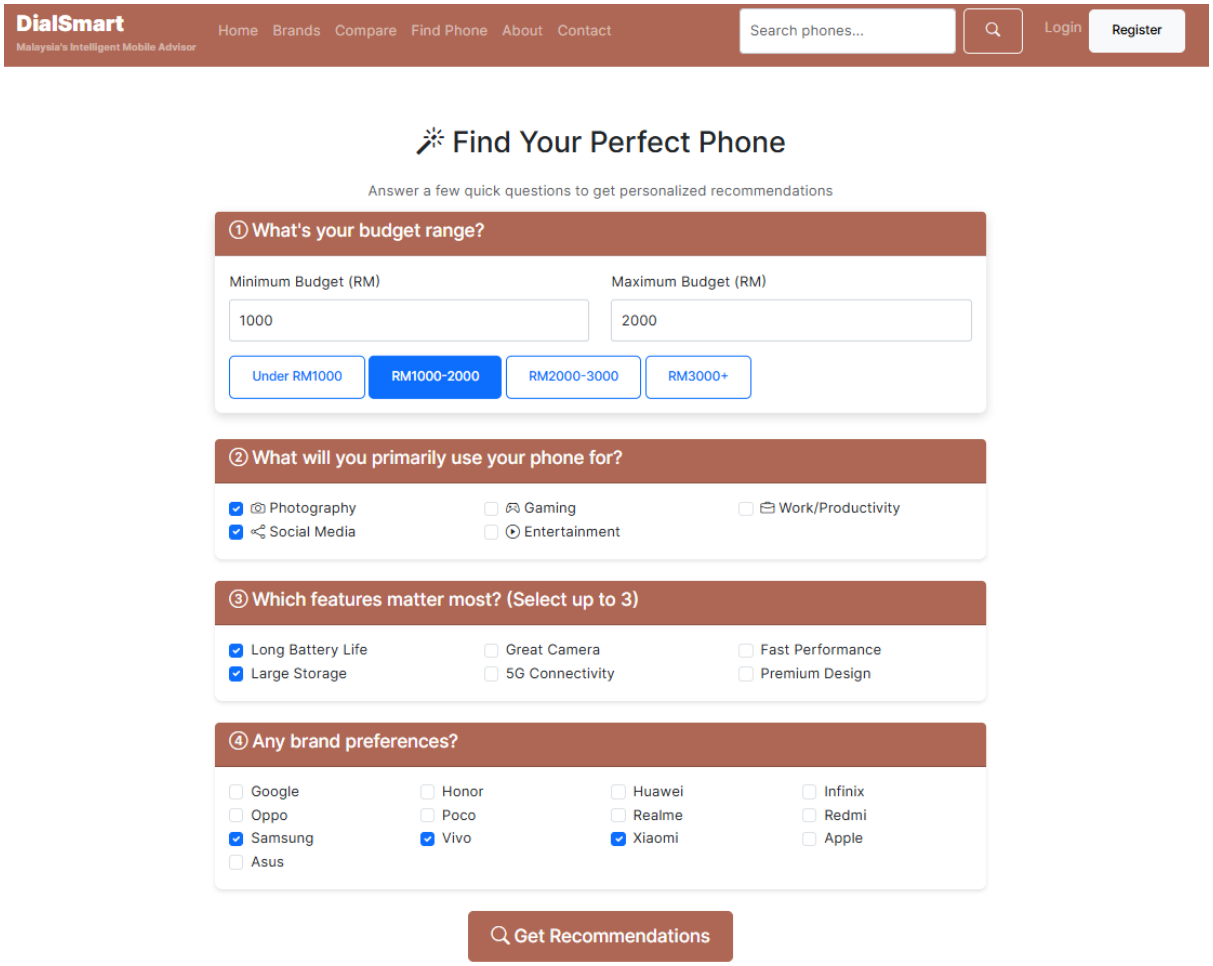

Figure 4.35 Find Phone Page

This intelligent recommendation wizard guides users through a four-step questionnaire to generate personalized suggestions. Users specify their budget range (with preset options and custom sliders), primary usage patterns (Photography, Gaming, Work/Productivity, Social Media, Entertainment), essential features (battery life, camera, performance, storage, 5G, design), and brand preferences to receive tailored recommendations.

4.9.5.1 Find Phone Result - Find Phone Page

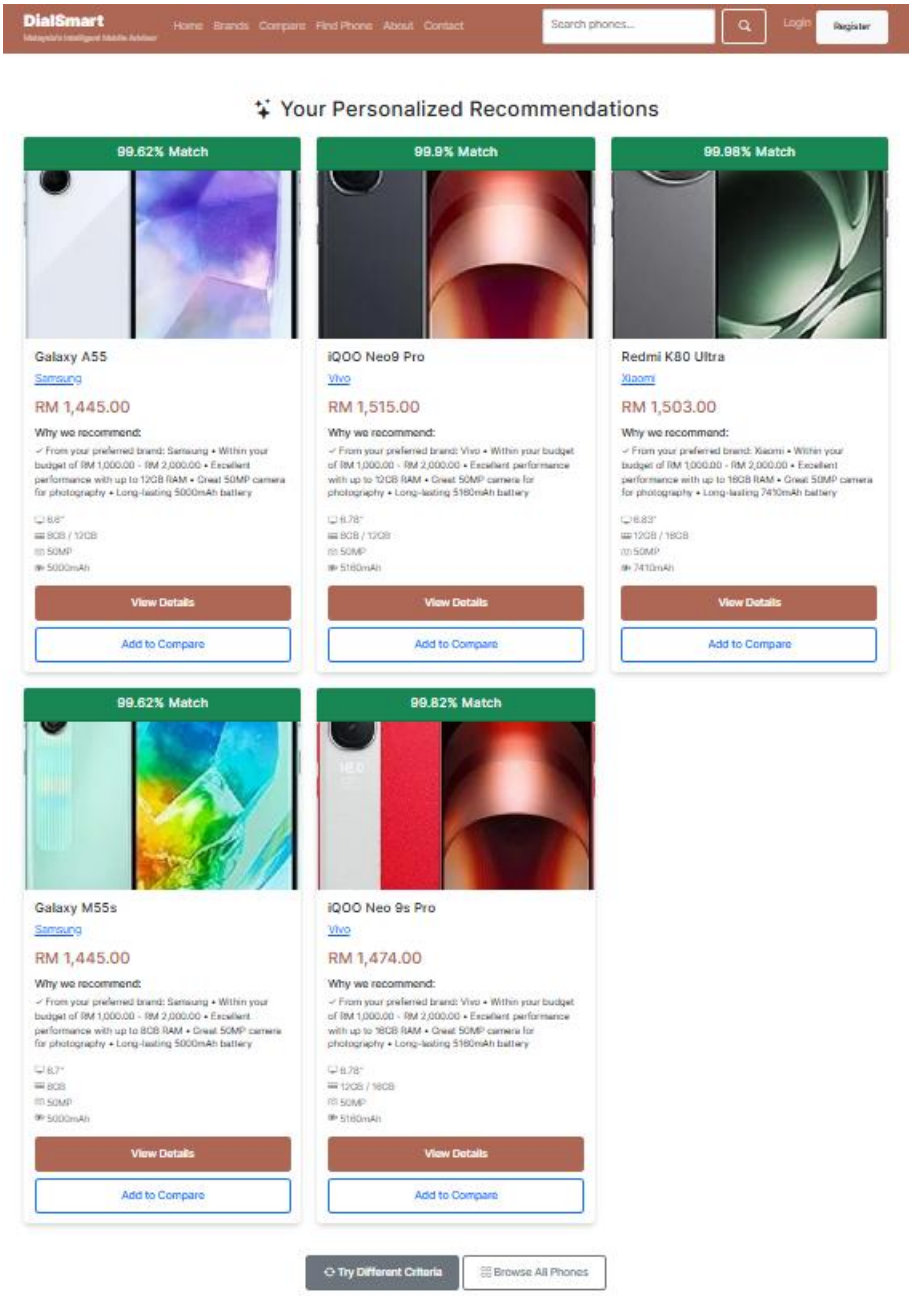

Figure 4.36 Find Phone Result - Find Phone Page

The results page displays personalized smartphone recommendations with match percentages (e.g., 99.92% Match, 99.5% Match) to indicate how well each device aligns with user preferences. Each recommendation card includes the phone image, model name, brand, price, match rationale explaining "Why we recommend" with specific features, key specifications, and options to view details or add to comparison.

4.9.6 Phone Details Page

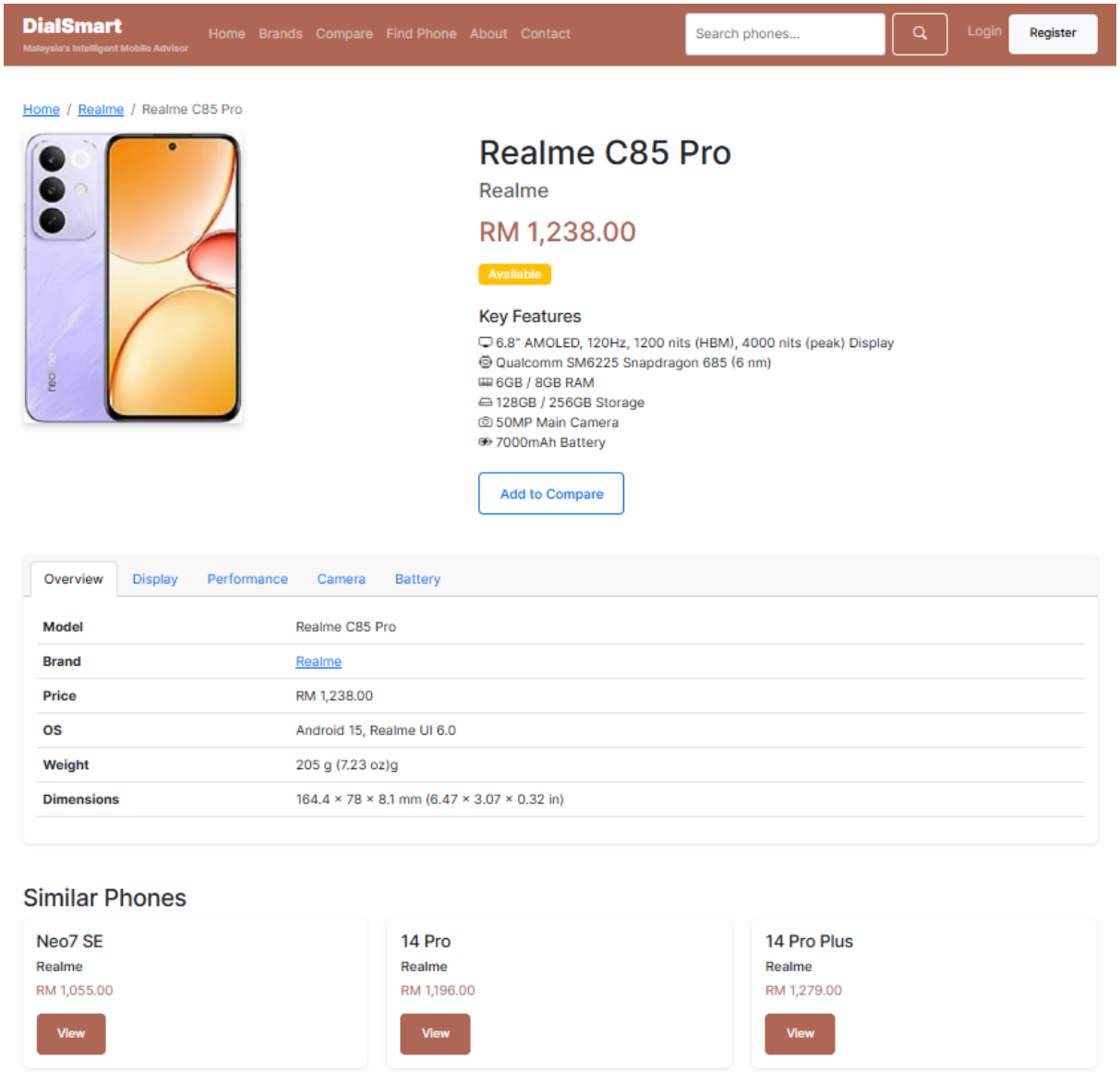

Figure 4.37 Phone Details Page

This comprehensive product page presents detailed smartphone specifications organized into tabbed sections (Overview, Display, Performance, Camera, Battery, Connectivity). The page features high-quality product images, pricing, availability status, key features highlights, and a similar phones section to facilitate comparative research and informed decision-making.

4.9.6.1 Official Brand Page - Phone Details Page

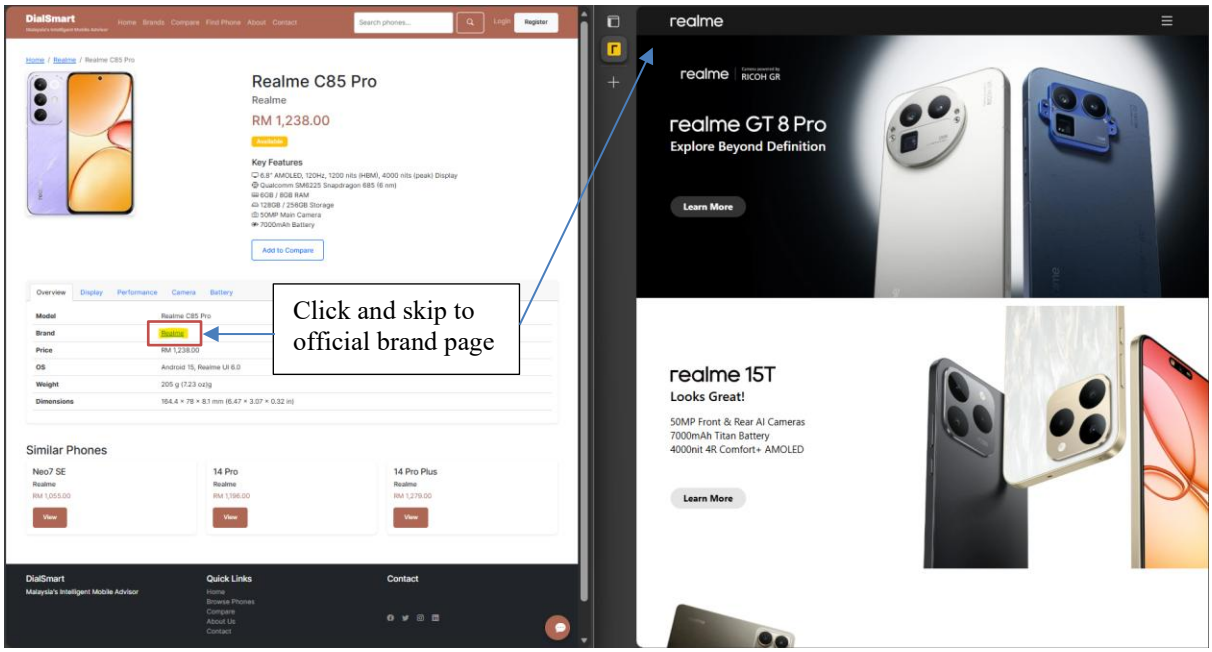

Figure 4.38 Official Brand Page - Phone Details Page

These interfaces provide seamless navigation to official manufacturer websites. Users can click directly to brand pages from phone details, maintaining continuity while accessing authoritative product information.

4.9.6.2 Brand Page - Phone Details Page

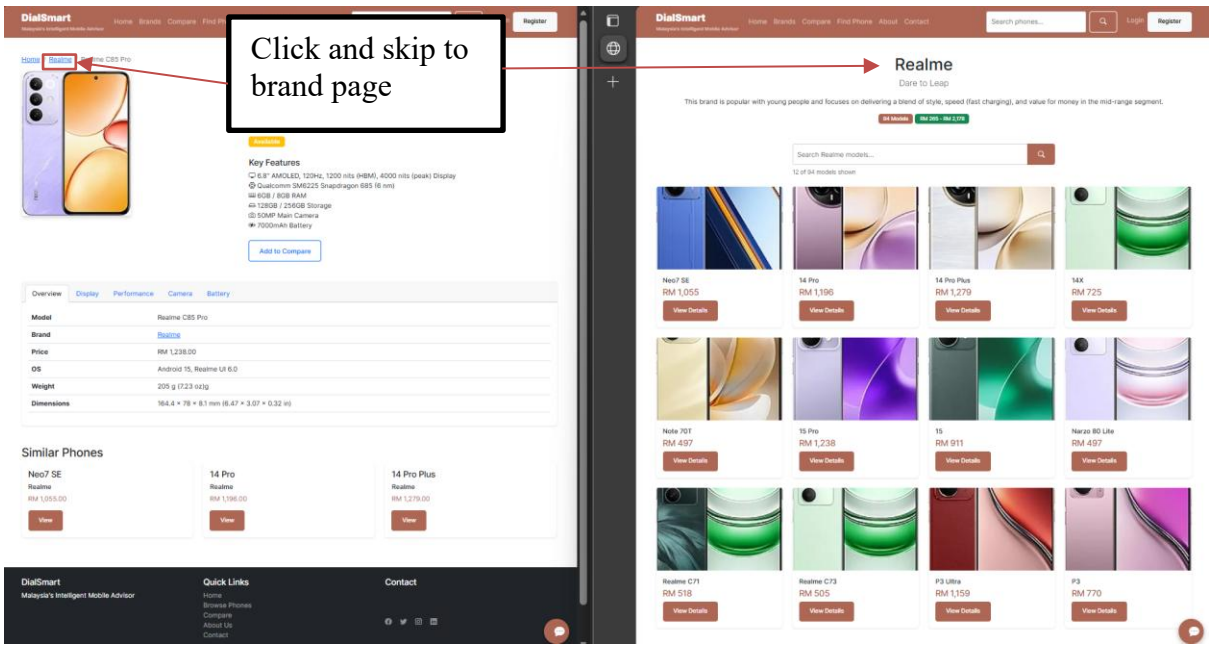

Figure 4.39 Brand Page - Phone Details Page

These interfaces provide seamless navigation to DialSmart platform. . Users can click directly to brand pages from phone details, maintaining continuity while accessing authoritative product information. The brand search functionality within dedicated brand pages allows users to find specific models quickly.

4.9.7 Brand Page

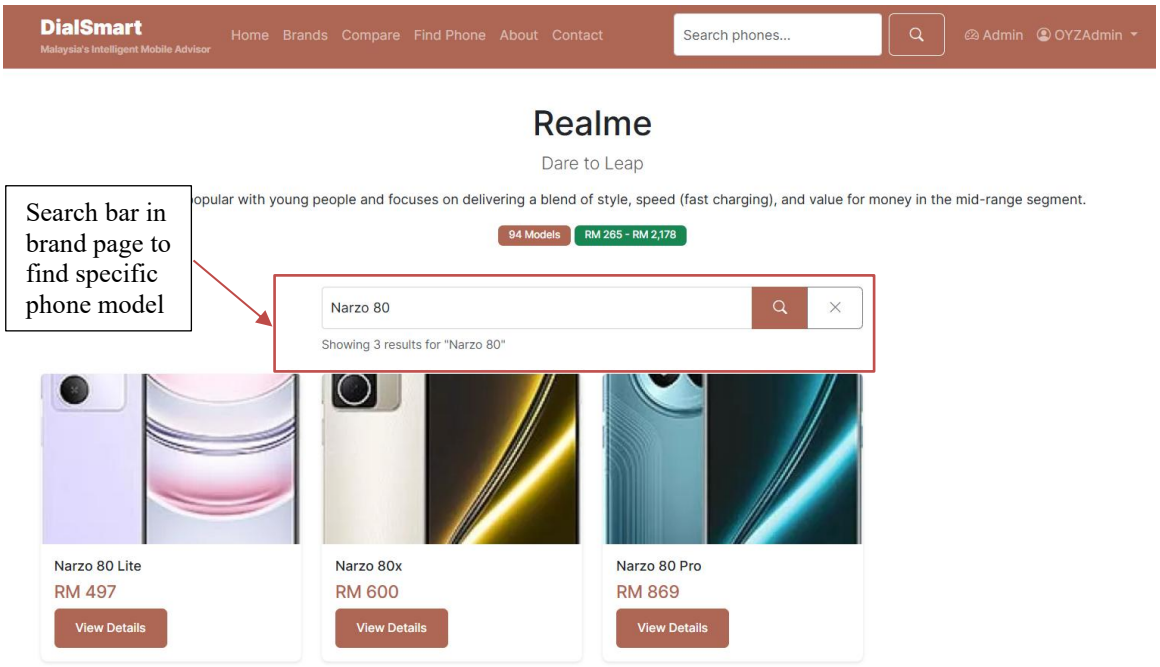

Figure 4.40 Brand Page

Brand-specific pages display all available models from a particular manufacturer (e.g., Realme) with the tagline "Dare to Leap." The page includes a search function for finding specific models within the brand catalog, displaying phone cards with images, model names, pricing, and "View Details" buttons organized in a responsive grid layout.

4.9.8 Search Bar

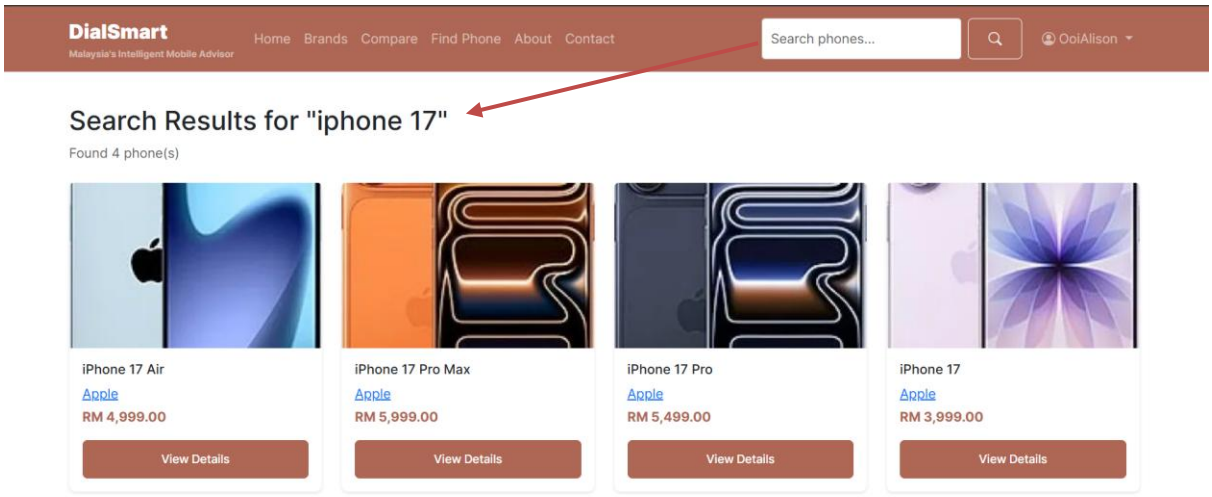

Figure 4.41 Search Bar

The global search functionality enables users to find specific phone models across all brands by entering search terms like "iPhone 17." Results display matching devices with images, model names, brand affiliations, pricing, and direct access to detailed specifications, streamlining the device discovery process.

4.9.9 Chatbot Assistant

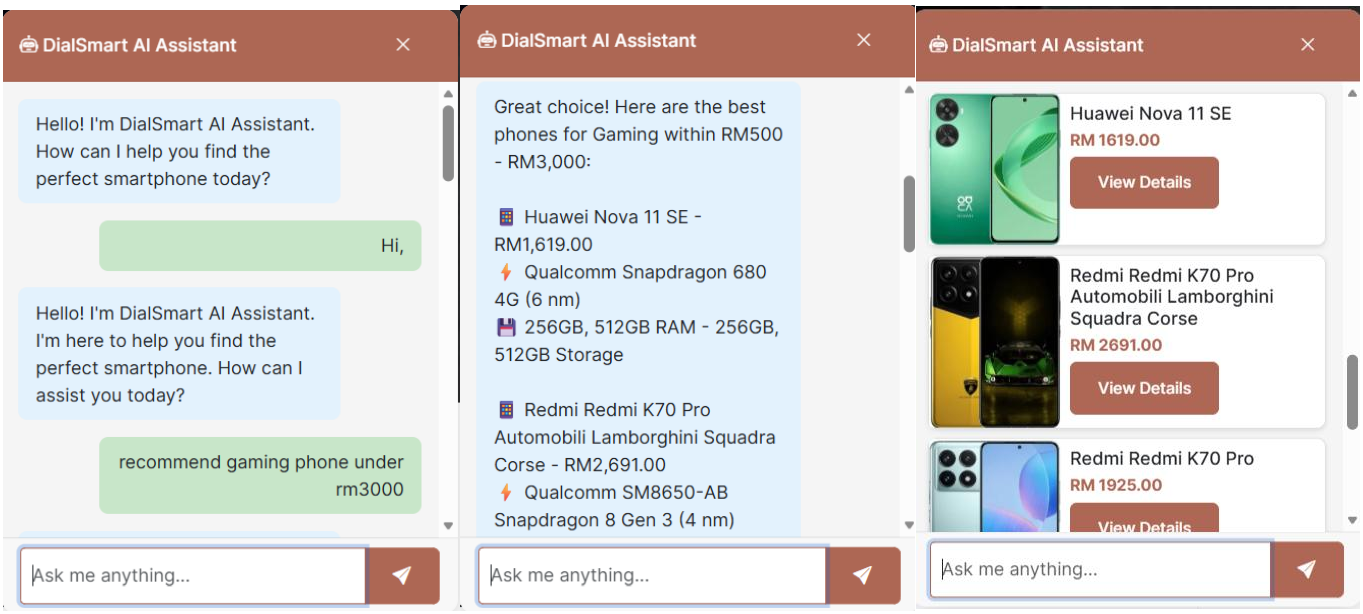

Figure 4.42 Chatbot Assistant

The AI-powered conversational assistant provides instant support through natural language interaction. The chatbot interface welcomes users, understands queries like "recommend gaming phone under rm3000," and responds with relevant suggestions including device images, specifications (processor, RAM, storage, camera), pricing, and action buttons for viewing details or comparisons.

4.9.10 About Us Page

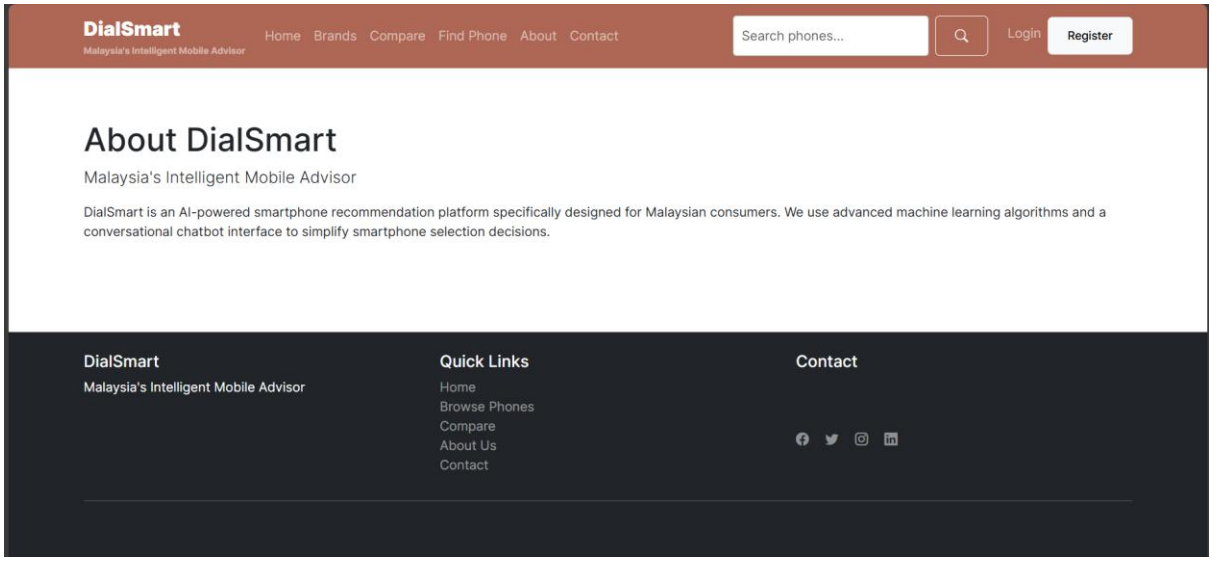

Figure 4.43 About Us Page

This informational page explains DialSmart's mission as "Malaysia's Intelligent Mobile Advisor," describing the platform as an AI-powered smartphone recommendation system designed specifically for Malaysian consumers. The page uses advanced machine learning algorithms and conversational chatbot interfaces to simplify smartphone selection decisions.

4.9.11 Contact Us Page

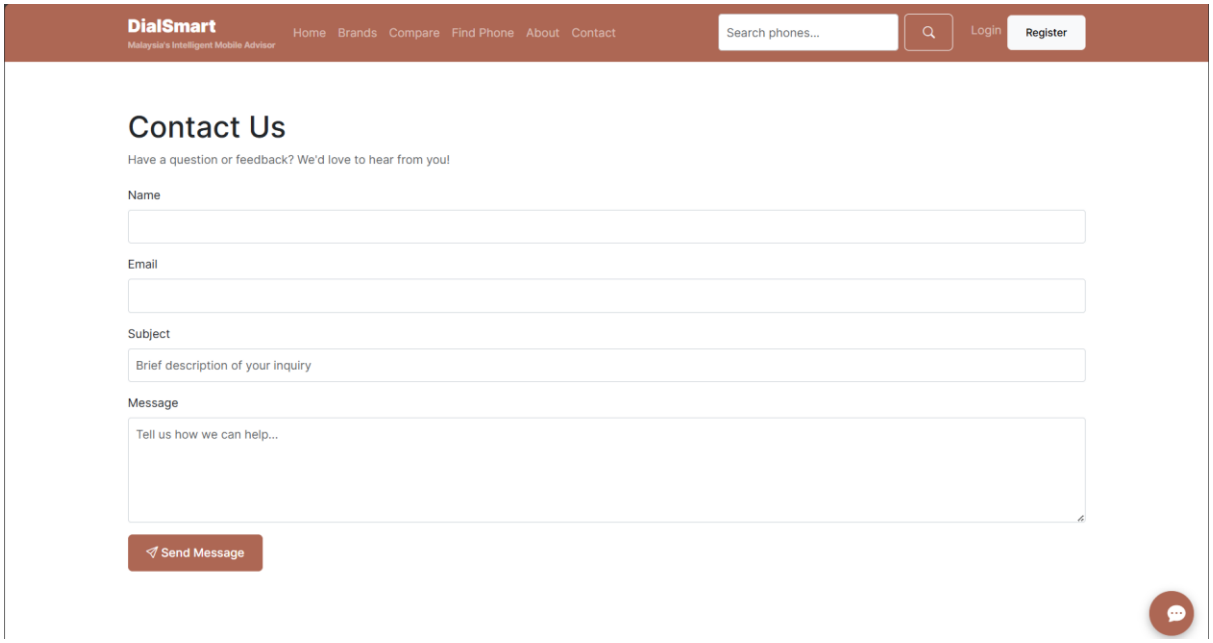

Figure 4.44 Contact Us Page

The contact form interface enables users to reach the support team by providing their name, email address, subject line with brief description, and detailed message. This communication channel facilitates user inquiries, feedback submission, and support requests, with responses handled through the admin reply system.

4.9.12 Admin Reply Message Page

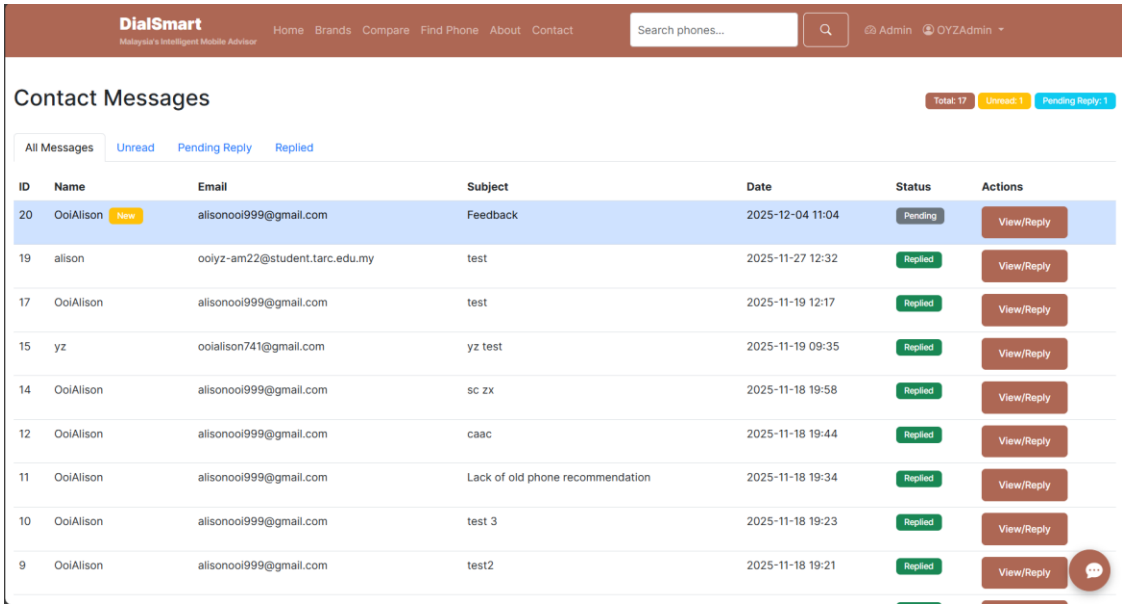

Figure 4.45 Admin Reply Message Page

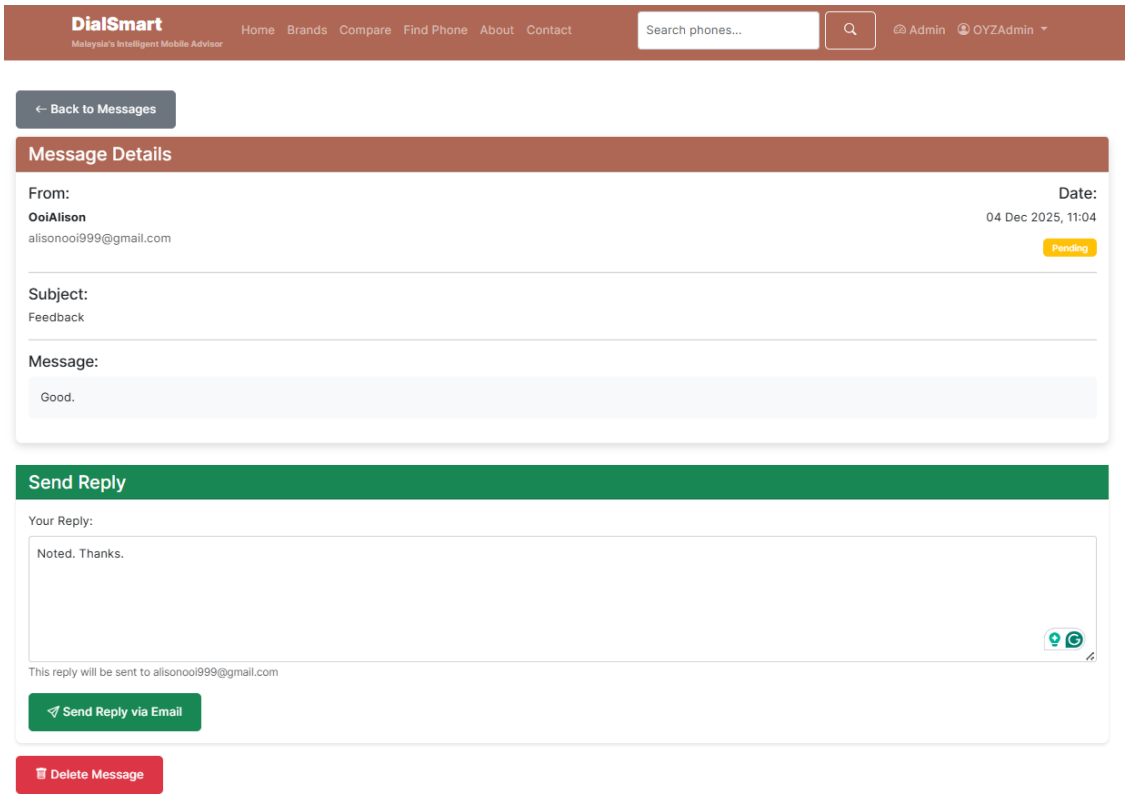

Figure 4.45 Admin Reply Message Page (continued)

This administrative interface displays incoming user messages in a tabular format showing message ID, sender name, email, subject, submission date, status (Pending/Replied), and action buttons. Clicking "View/Reply" opens a detailed message view where administrators can read user inquiries and compose responses that are automatically sent via email to users.

4.9.12.1 Reply sent - Admin Reply Message Page

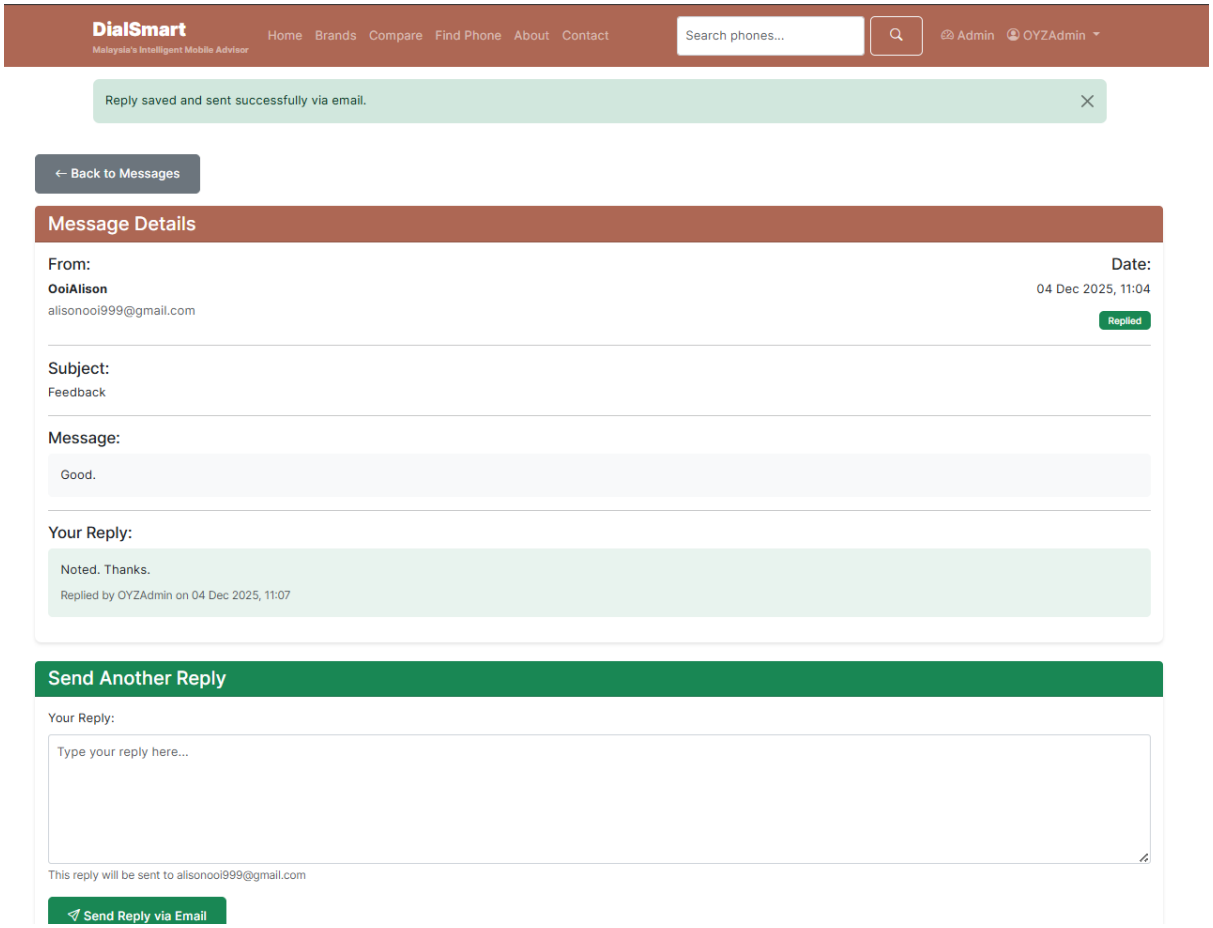

Figure 4.46 Reply sent - Admin Reply Message Page

After administrators submit replies, the system displays a success notification confirming "Reply saved and sent successfully via email." The interface shows the complete message thread including the user's original message and the administrator's response, with options to send additional follow-up replies if needed.

4.9.12.2 User received reply via email- Admin Reply Message Page

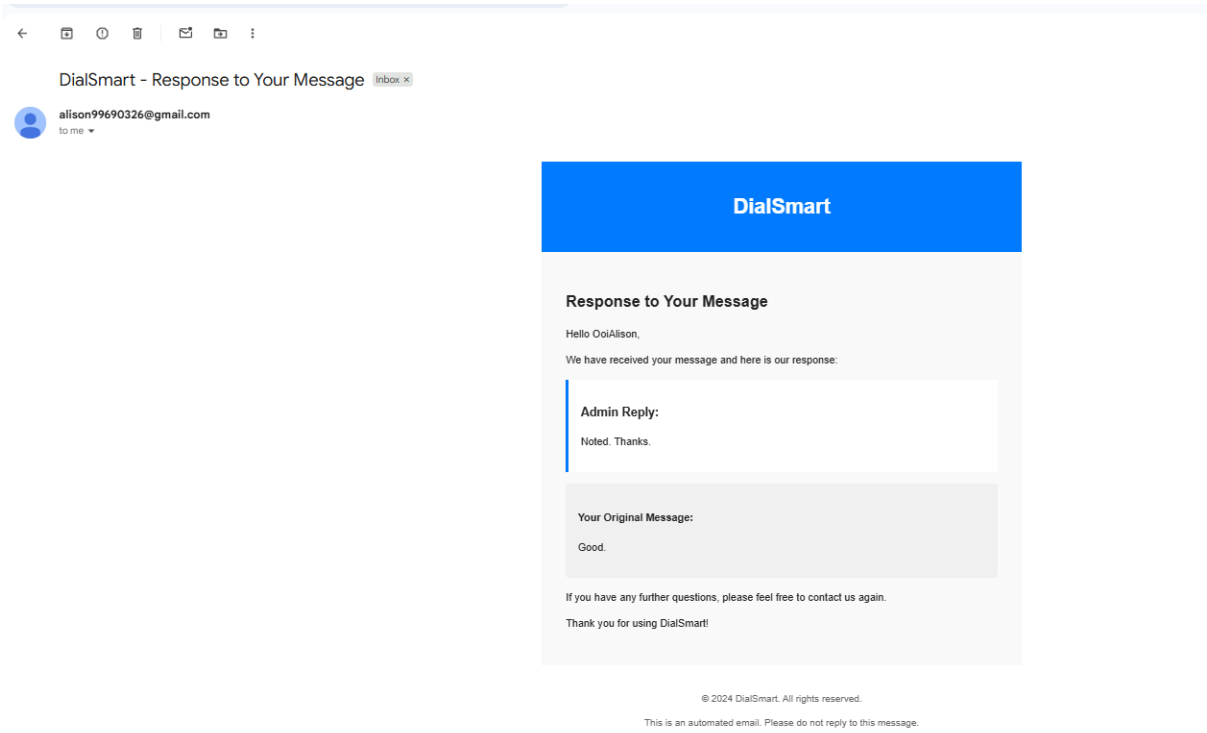

Figure 4.47 User received reply via email- Admin Reply Message Page

Users receive professionally formatted email notifications when administrators respond to their inquiries. The email displays both the admin's reply and the user's original message for context, maintaining clear communication threads. The automated response system ensures timely user support and maintains engagement.

4.9.13 Admin Index Page

4.9.13.1 Super Admin Index Page

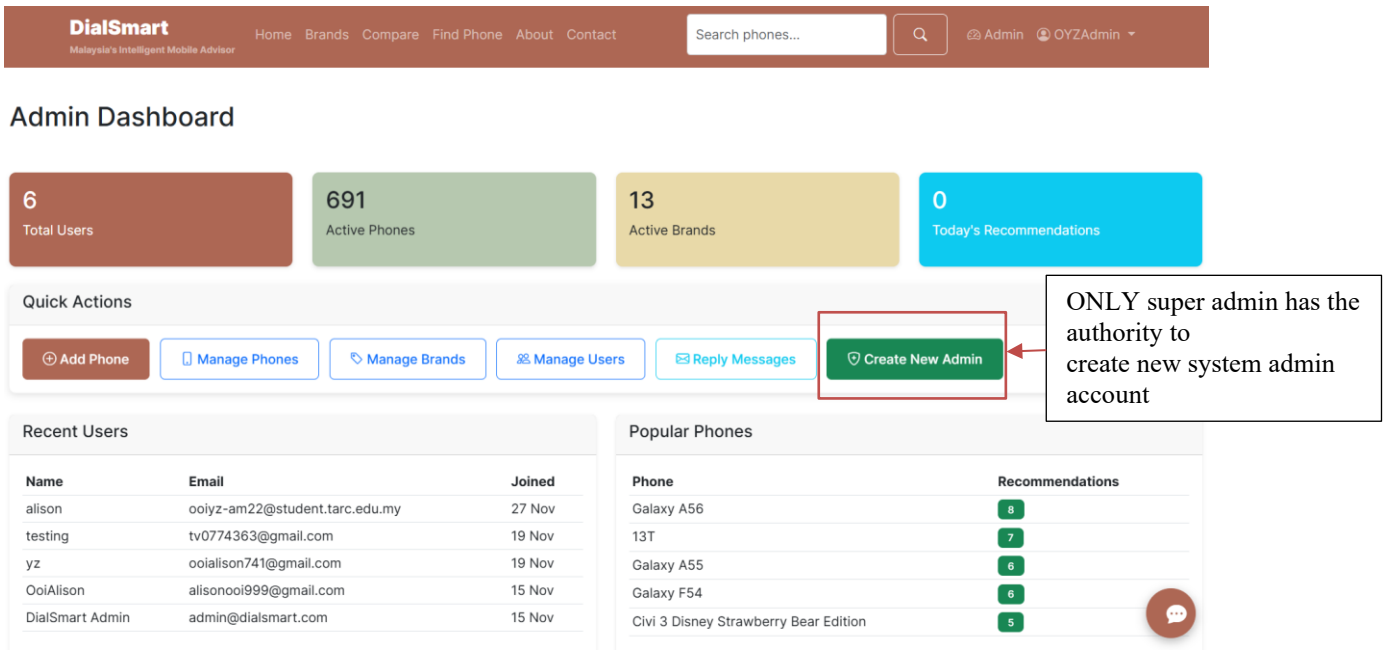

Figure 4.48 Super Admin Index Page

4.9.13.2 System Admin Index Page

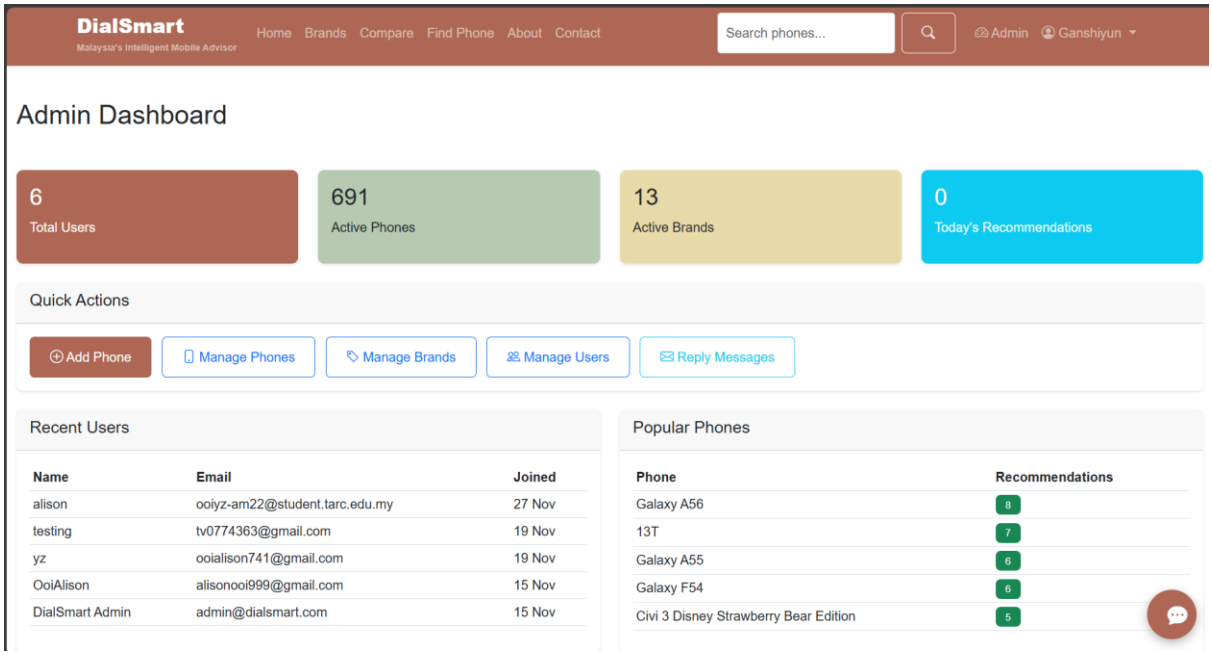

Figure 4.49 System Admin Index Page

The admin dashboard provides comprehensive system oversight with statistics cards showing Total Users, Active Phones, Active Brands, and Today's Recommendations. Super Admins have exclusive access to a "Create New Admin" button for managing administrative accounts, while System Admins lack this privilege, maintaining proper access control hierarchies.

4.9.14 Create New Admin Page [Super Admin]

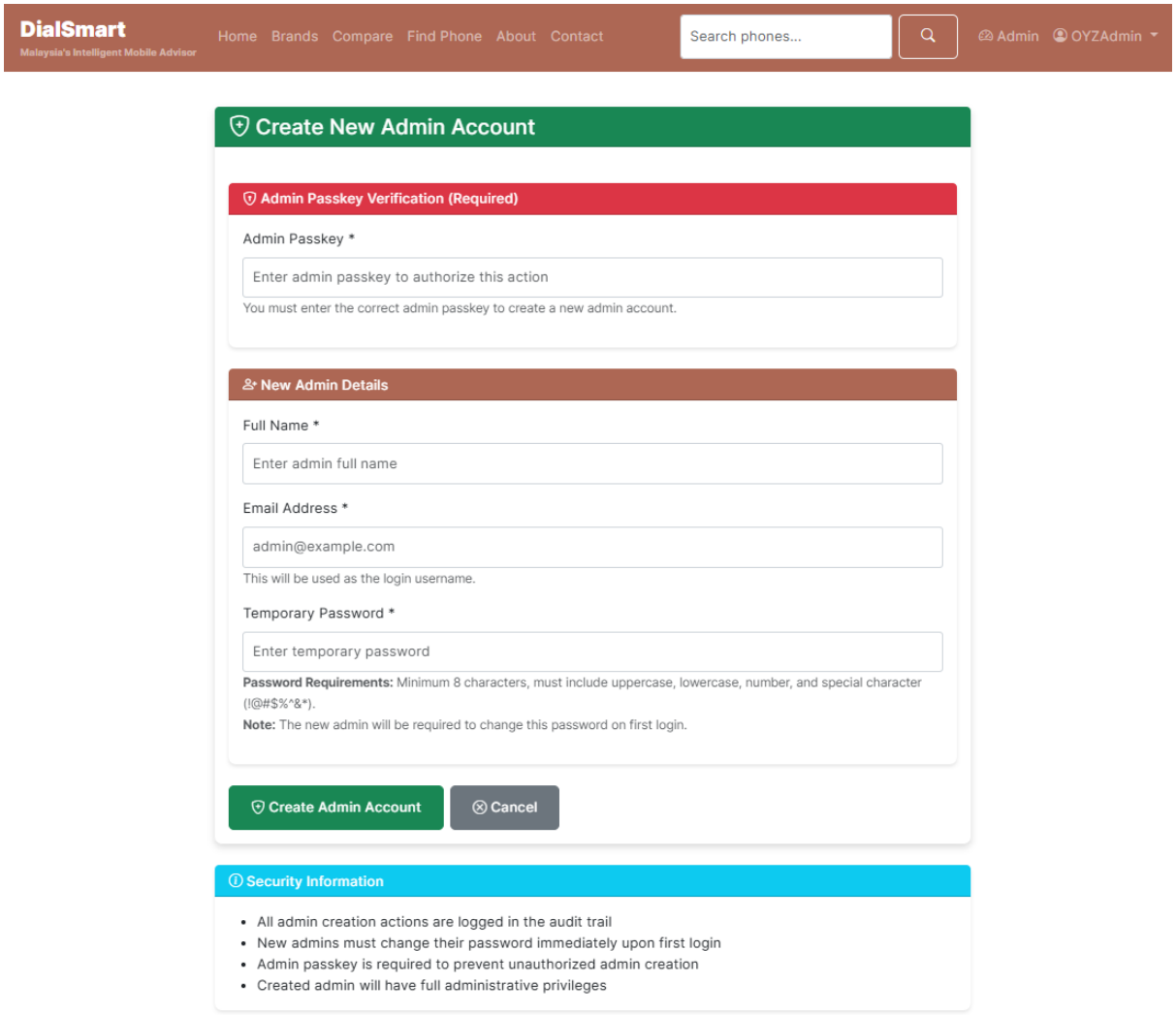

Figure 4.50 Create New Admin Page [Super Admin]

This secure interface enables Super Admins to create new administrative accounts by first verifying their own credentials through an admin passkey field. The form collects the new admin's full name, email address, and temporary password (with strict requirements: minimum 8 characters, including uppercase, lowercase, number, and special character). Security information clarifies that all admin creation actions are logged and new admins must change passwords upon first login.

4.9.15 Admin Mange Phone Brand Page

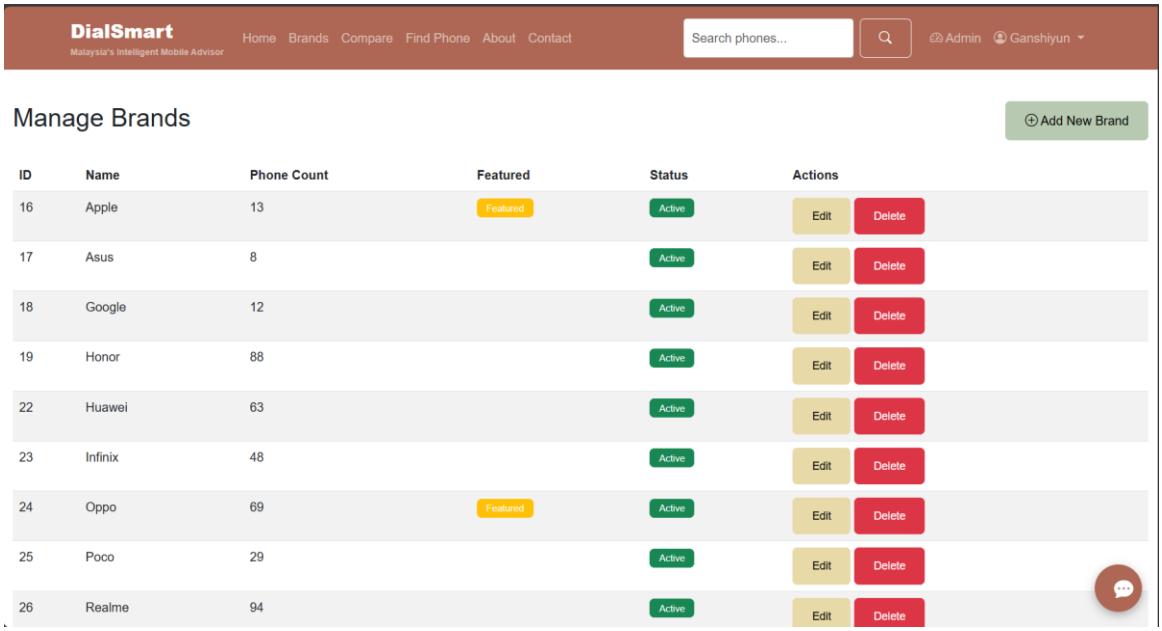

Figure 4.51 Admin Manage Phone Brand Page

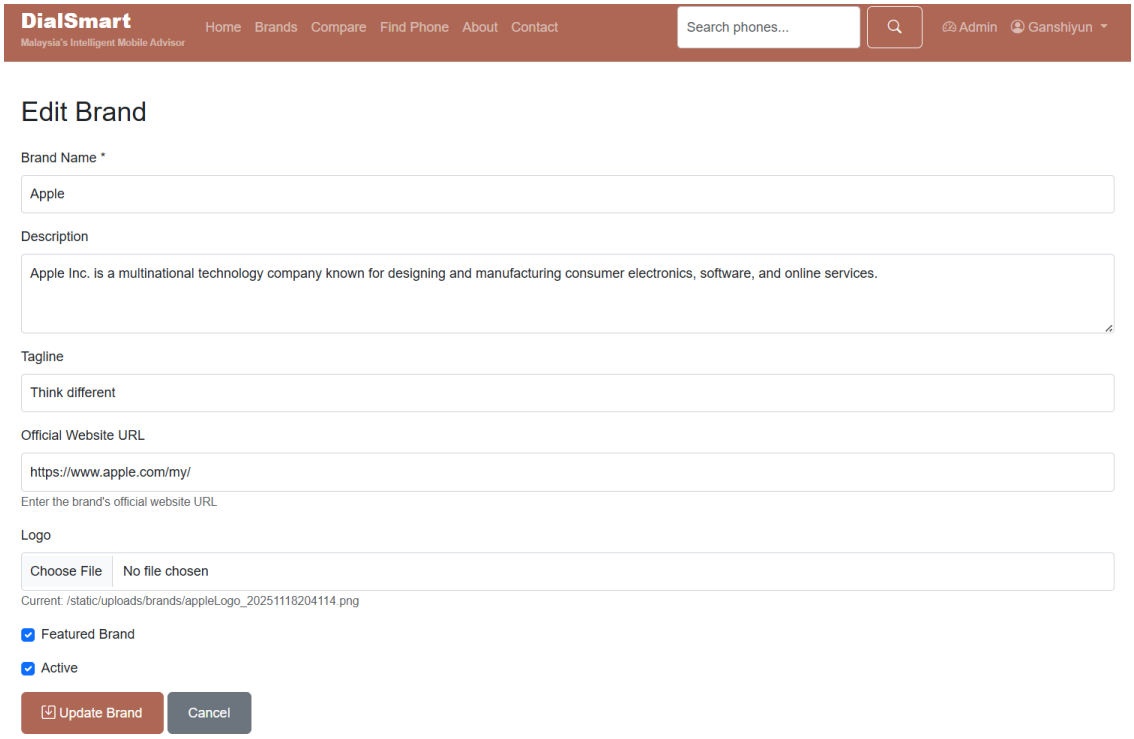

Figure 4.51 Admin Manage Phone Brand Page (continued)

The brand management interface displays all smartphone brands in a table format showing ID, brand name, phone count, featured status, active/inactive status, and edit/delete actions. Administrators can modify brand information including brand name, description, tagline, official website URL, logo upload, featured brand checkbox, and active status toggle to maintain accurate brand representation.

4.9.16 Admin Mange Phone Model Page

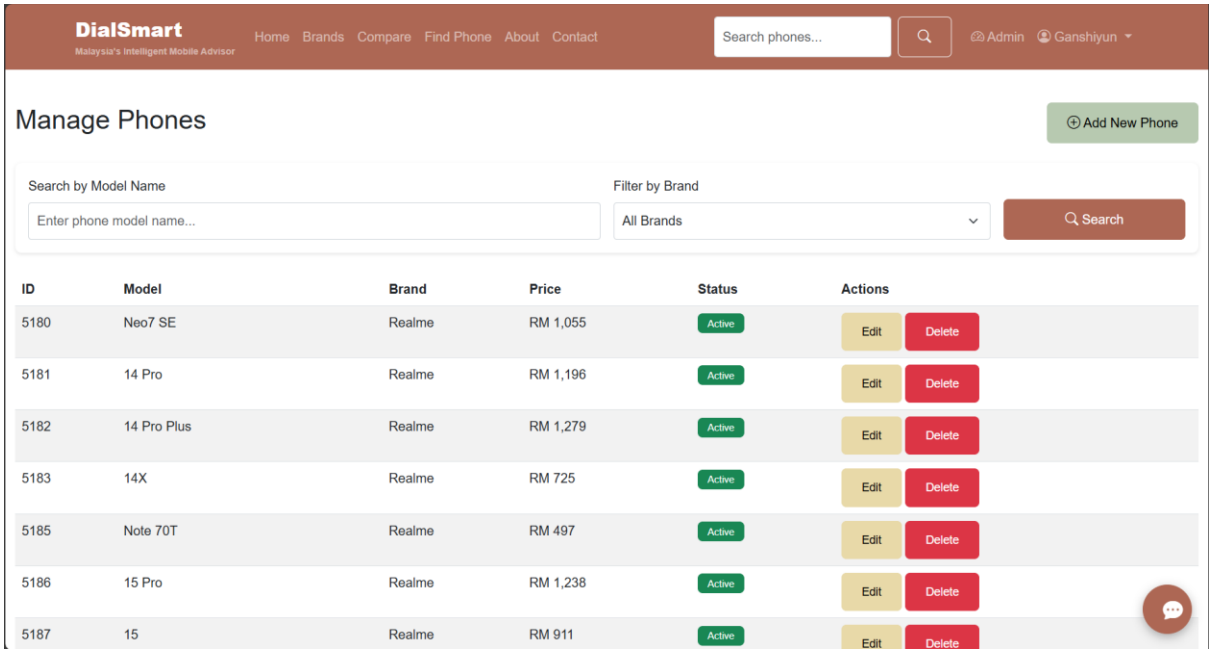

Figure 4.52 Admin Mange Phone Model Page

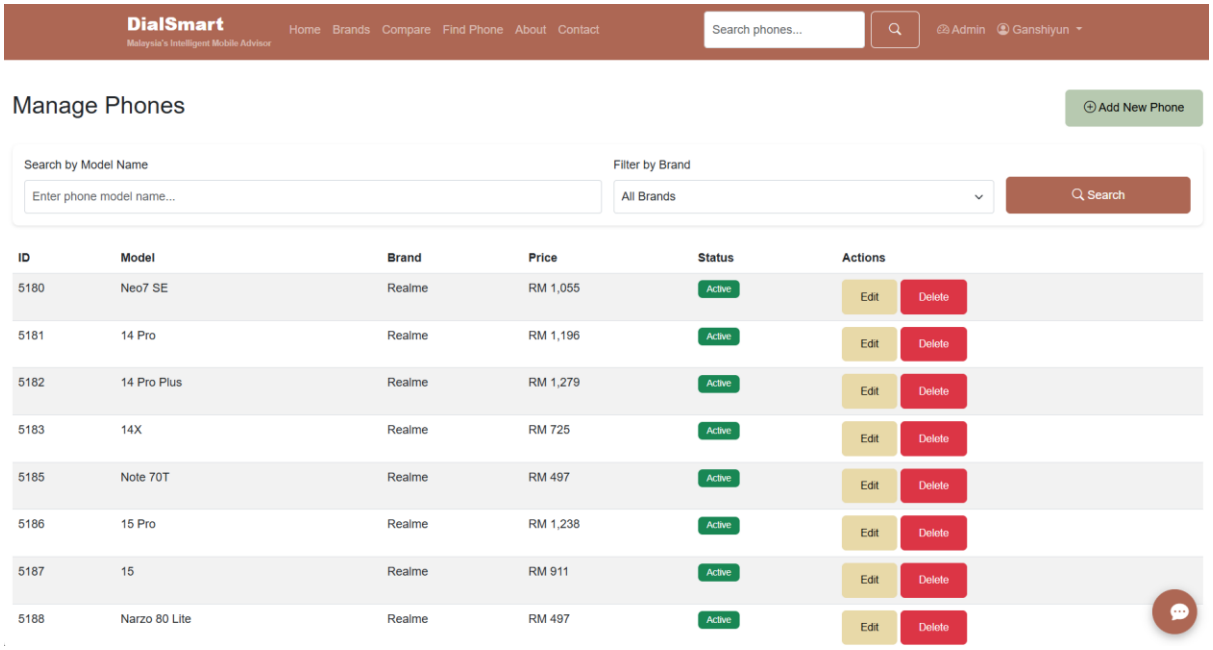

Figure 4.52 Admin Mange Phone Model Page (continued)

This comprehensive phone inventory management system enables administrators to search by model name or filter by brand. The table displays phone ID, model name, brand affiliation, price in MYR, status, and management actions. Administrators can add new phones, edit existing specifications, or remove outdated models to maintain current smartphone offerings.

4.9.17 Admin Mange User Page

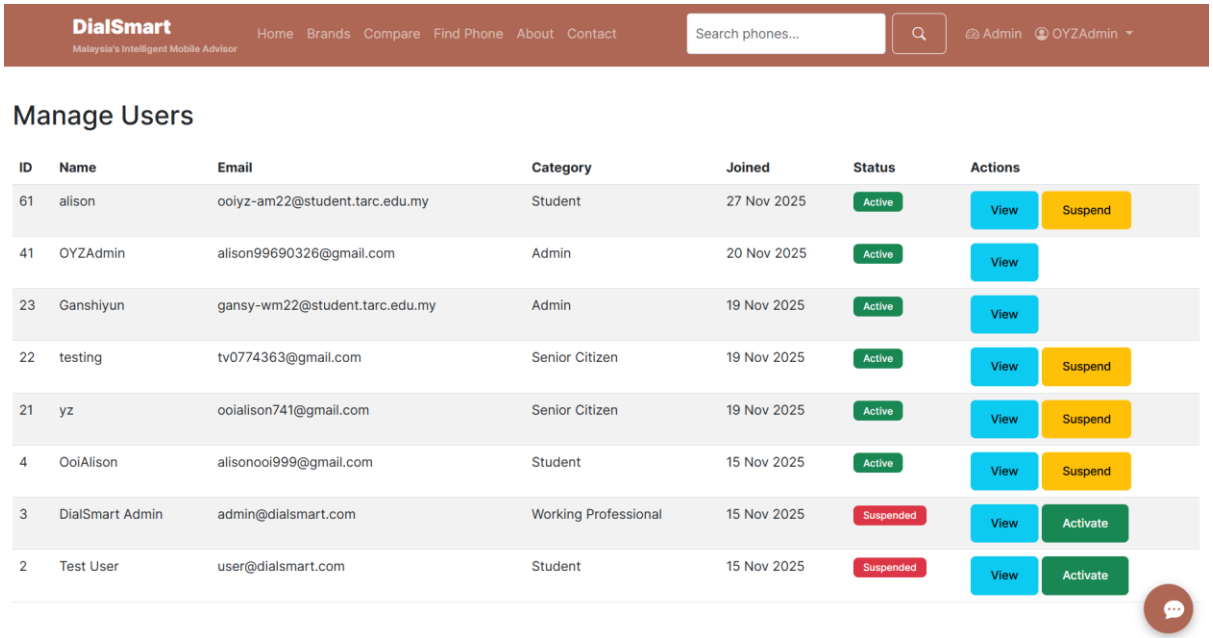

Figure 4.53 Admin Mange User Page

The user management dashboard presents all registered users with their ID, name, email, category (Student/Working Professional/Senior Citizen/Admin), registration date, account status, and action buttons. Administrators can view detailed user profiles showing recommendation history with match percentages, suspend accounts for policy violations, or activate suspended accounts with automated email notifications.

4.9.17.1 View User - Admin Mange User Page

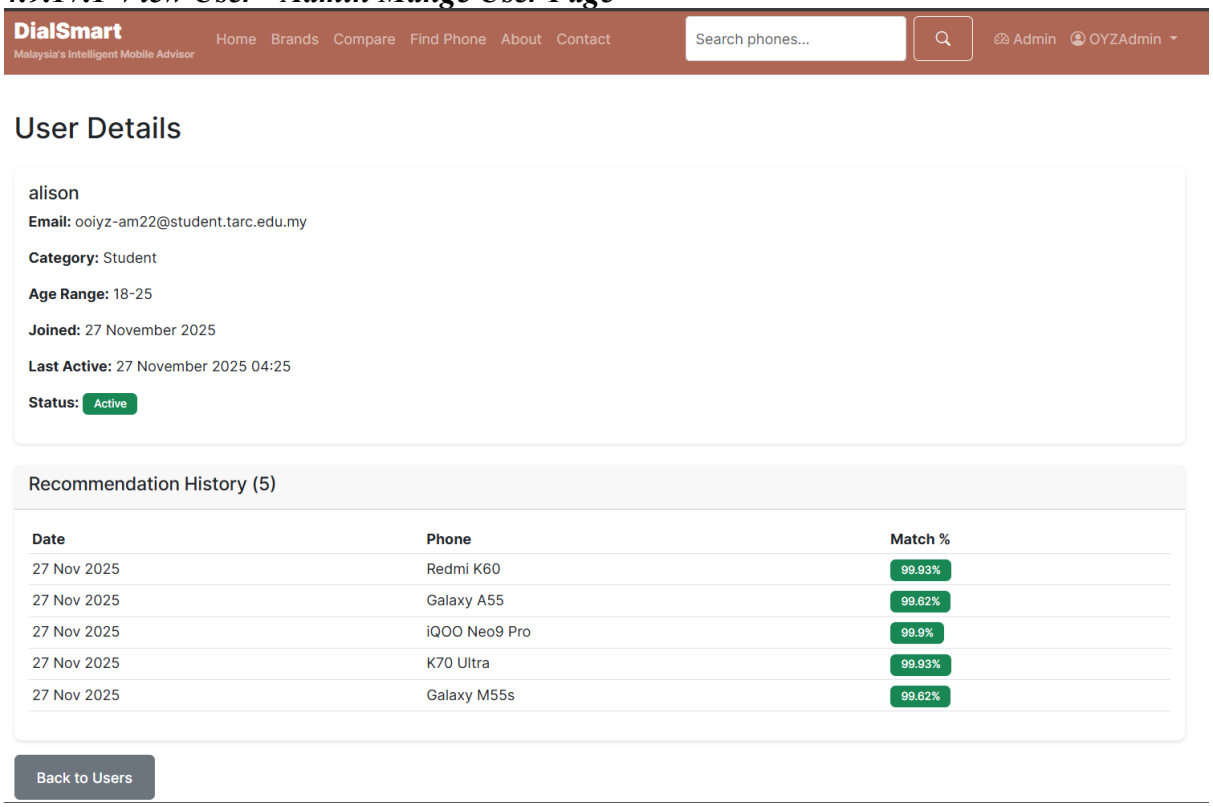

Figure 4.54 View User - Admin Mange User Page

The detailed user profile view displays comprehensive information including email, category, age range, registration date, last active timestamp, and account status. The recommendation history section shows dates, recommended phones, and match percentages (e.g., 99.5%, 99.82%, 99.92%), enabling administrators to understand user engagement patterns and recommendation effectiveness.

4.9.17.2 Suspend User - Admin Mange User Page

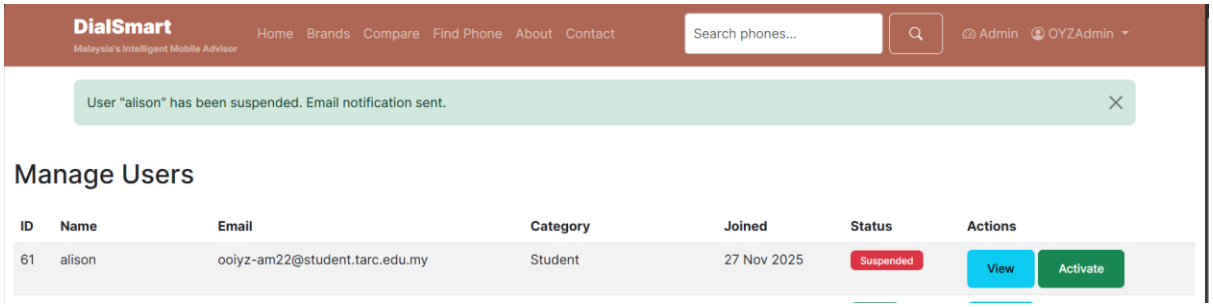

Figure 4.55 Suspend User - Admin Mange User Page

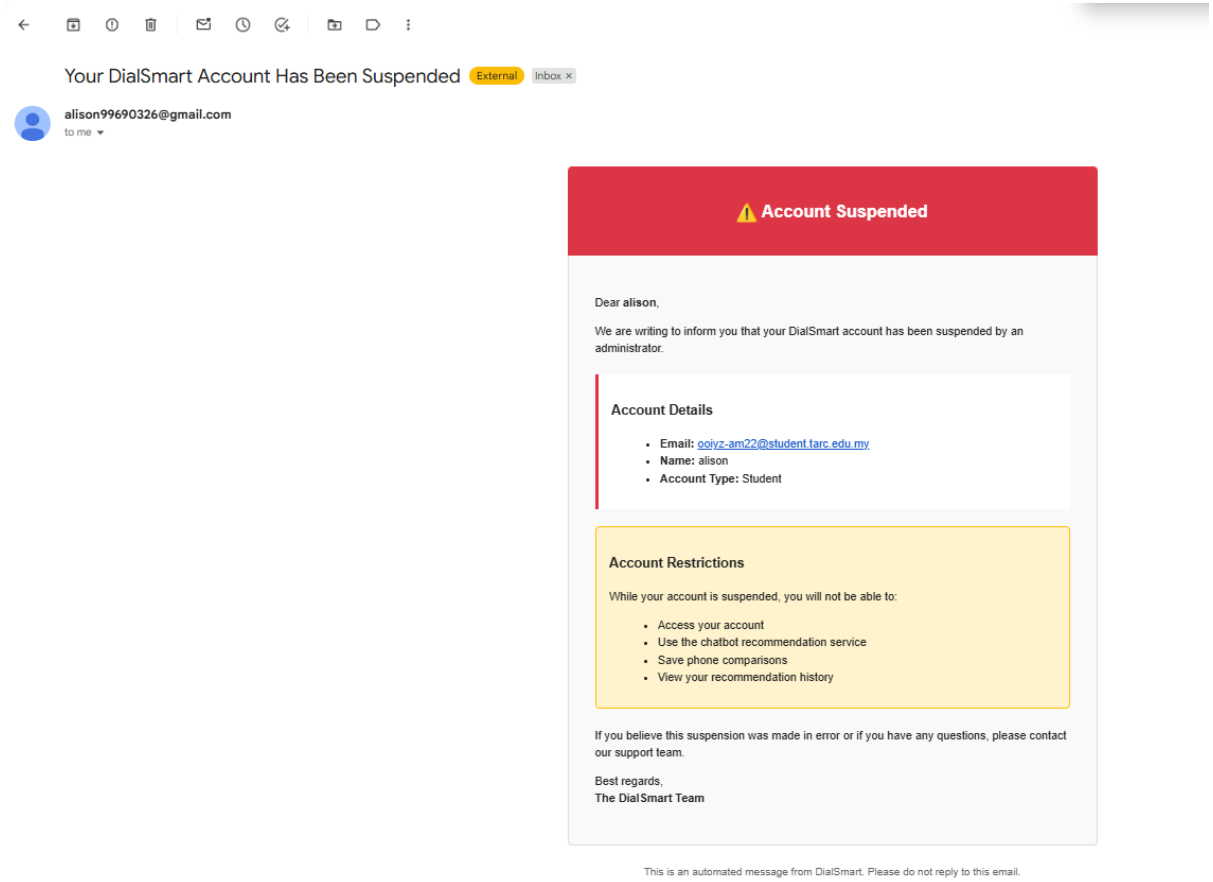

Figure 4.55 Suspend User - Admin Mange User Page (continued)

When administrators suspend user accounts, the system sends automated email notifications explaining the suspension. The email details account restrictions (cannot access account, use chatbot, compare phones, or view history) and provides support contact information. This transparent communication maintains user trust while enforcing platform policies.

4.9.17.3 Activate User - Admin Mange User Page

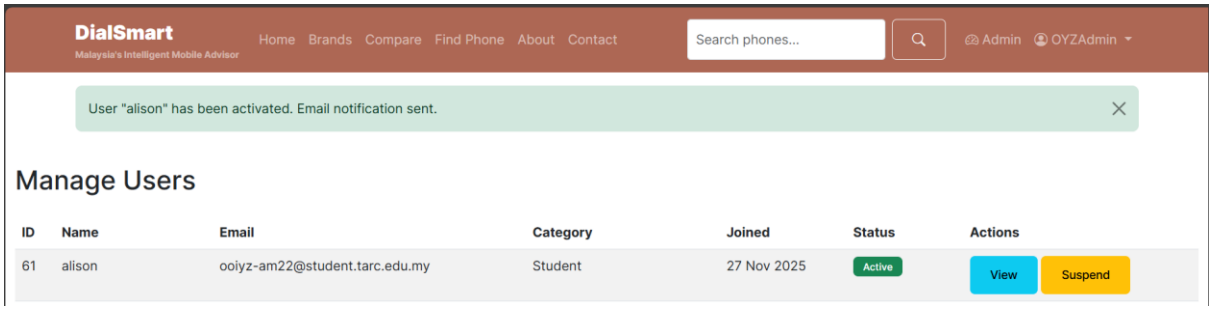

Figure 4.56 Activate User - Admin Mange User Page

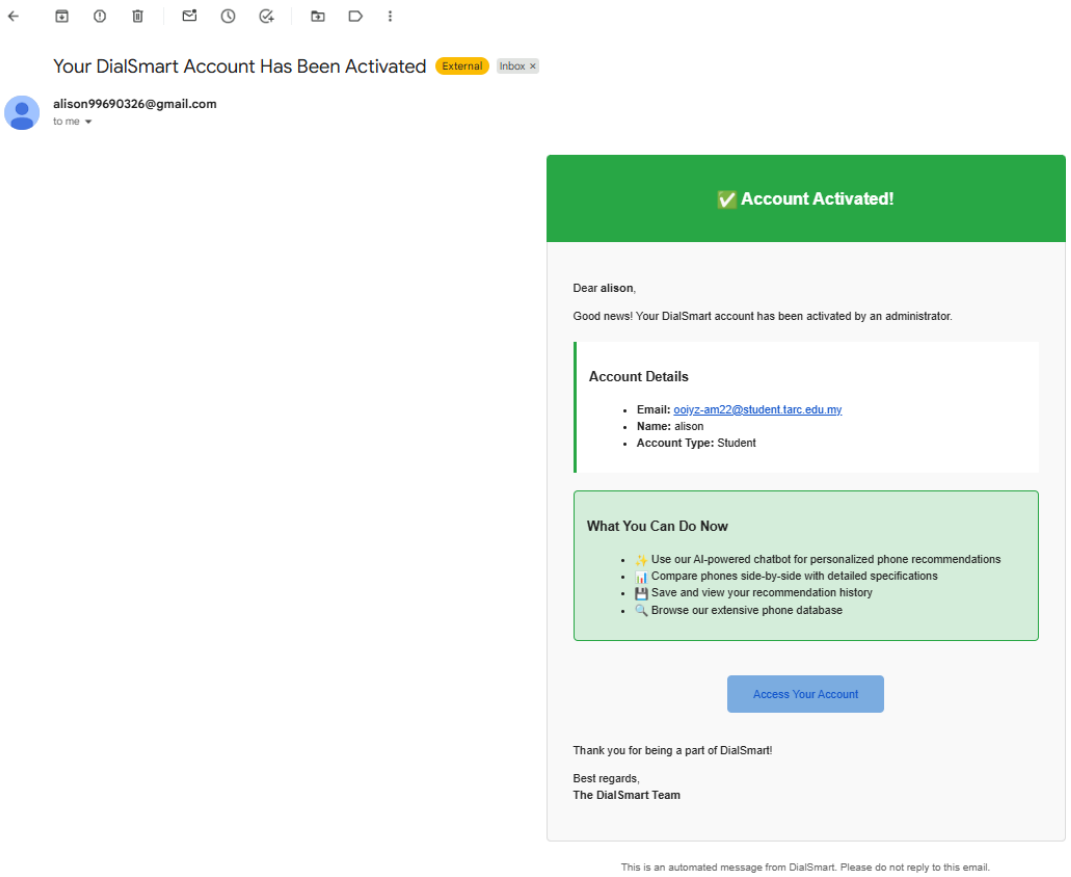

Figure 4.56 Activate User - Admin Mange User Page

Account activation triggers automated emails notifying users that their access has been restored by administrators. The welcome-back email confirms account details and outlines available features including AI-powered recommendations, phone comparisons, recommendation history saving, and extensive phone database browsing, re-engaging previously suspended users.

4.10 Chapter Summary and Evaluation

Chapter 4 focuses on the system design phase, emphasizing on designing application's architecture, components, and user interface. This chapter acted as a link between requirements gathering phase and implementation phase, providing a complete overview of the system's design.

We began by examining database design, including the Entity Relationship Diagram (ERD) and Data Dictionary. The ERD assisted in visualizing relationships between distinct entities, whereas the Data Dictionary supplied precise information about each table's attributes and structure. This assured our database was constructed effectively to store and manage all necessary data.

Following that, we examined various diagrammatic representations of our system. The Class Diagram provided representation of the system's classes and their linkages, aiding in identifying essential components and interactions. Detailed Use Case Diagrams were built for specific modules such as user management, AI recommendation, phone comparison, chatbot interaction, and administrative management. These diagrams detailed the exact activities and interactions within each module.

Sequence Diagrams were constructed to demonstrate the system's dynamic behavior, illustrating chronological sequence of interactions between various objects and components. A Layered Software Architecture Diagram emphasized the various layers and their interactions, ensuring a modular and scalable architecture.

We then focused on User Interface (UI) Design encompassing both user and administrative interfaces. The user interface includes homepage, registration pages, member dashboard, AI recommendation wizard, and phone comparison tools. The administrative interface provides system management capabilities including brand management, phone inventory control, and AI model training interfaces.

The evaluation of this chapter is beneficial since it demonstrates key elements of system design in depth. The use of design diagrams aids in visualizing and comprehending the application's structure and functionality. This chapter promotes successful communication by providing extensive explanations to development team members and stakeholders.

In conclusion, Chapter 4 presented a thorough description of the system design phase. The methodical approach assures that the application is well-organized, scalable, and maintainable. This chapter sets a strong foundation for subsequent development and implementation phases, resulting in a robust and efficient system.

Chapter 5

Implementation

5 Implementation

Chapter 5 provides a detailed exploration of the implementation aspects of the DialSmart AI-Powered Smartphone Recommendation System. This chapter covers the core architectural frameworks including Client-Server architecture and Model-View-Controller (MVC) pattern, along with the application of design patterns and best practices. The chapter delves into the system's implementation details, including database architecture, AI recommendation engine, RESTful API endpoints, intelligent chatbot integration, and security mechanisms. The implementation utilizes Flask as the primary web framework, SQLAlchemy for database operations, and custom AI algorithms for generating personalized smartphone recommendations. This chapter demonstrates how each component integrates to create a robust, scalable, and intelligent recommendation system tailored to Malaysian smartphone consumers.

5.1 Client-Server Architecture

The foundation of the DialSmart web application lies in the client-server architecture, a pivotal framework extensively utilized in modern network applications. In this model, the client and server collaborate to provide a seamless and efficient experience for users interacting with the smartphone recommendation system. This symbiotic relationship between the client and server empowers the application with scalability, enabling it to accommodate a growing user base while delivering responsive and reliable performance. The clear separation of responsibilities, with the client focusing on user interactions and the server managing complex operations, results in a well-coordinated and efficient system.

5.1.1 Client

The client aspect represents the user interface, accessible through web browsers on desktop and mobile devices. Users engage with the client-side interface to explore the system's features, including personalized smartphone recommendations, detailed specifications browsing, phone comparisons, and interaction with the AI-powered chatbot assistant. When users initiate actions such as searching for phones, requesting recommendations, or comparing devices, the client sends HTTP requests to the server, processing the responses to present information and options dynamically. The client's interface is built using HTML5, CSS3, and JavaScript, serving as an interactive portal

that guides users through a seamless experience where actions trigger requests to the server.

The DialSmart system implements responsive web design principles to ensure optimal viewing and interaction across various devices and screen sizes. The client-side utilizes modern JavaScript for dynamic content updates, form validation, and asynchronous communication with the server through AJAX requests. This approach enhances user experience by providing instant feedback and eliminating full page reloads for common operations.

5.1.2 Server

Conversely, the server acts as the central hub managing and processing client requests. The server, upon receiving user-initiated actions, meticulously crafts and delivers tailored responses back to the client. It is responsible for storing and retrieving data from the database, executing business logic, implementing the AI recommendation algorithm, and ensuring the overall functionality of the application. In the DialSmart system, the server handles user requests related to smartphone recommendations, phone specifications retrieval, brand information, comparison operations, and chatbot interactions. It interacts with an Oracle database to maintain user information, phone details, specifications, and recommendation history, ensuring data consistency and security.

The DialSmart server is built using Flask, a lightweight Python web framework that follows the WSGI (Web Server Gateway Interface) standard. The application is configured to run on host 0.0.0.0 and port 5000, allowing access from both local and network clients. The server initialization is managed through the application entry point as shown below:

```python
"""
DialSmart Application Entry Point
Run this file to start the DialSmart web application
"""
import os
from app import create_app, db
from app.models import User, Brand, Phone, PhoneSpecification

# Create application instance
app = create_app(os.getenv('FLASK_ENV', 'development'))

if __name__ == '__main__':
    # Run the application
    app.run(
        host='0.0.0.0',
        port=5000,
        debug=True
    )
```

Figure 5.1 Server Initialization

The server implements a factory pattern for application initialization, which promotes modularity and testability. This pattern allows for different configurations depending on the deployment environment (development, testing, or production).

5.1.3 Interaction

The interaction between the client and server is facilitated through the HTTP/HTTPS protocol. This communication protocol proves crucial when users initiate actions such as requesting recommendations, searching for phones, or comparing devices. Within this orchestrated flow, the client serves as the initiator, sending HTTP requests (GET, POST, PUT, DELETE) to appropriate server endpoints. The server, equipped with the capability to comprehend and act on these requests, processes the user's intentions, executes necessary operations including database queries and AI computations, and crafts tailored HTTP responses. These responses contain JSON-formatted data that the client processes to update the user interface dynamically. This two-way communication, governed by the HTTP protocol, not only ensures the accuracy of user actions but also lays the groundwork for dynamic and responsive user experiences, enhancing the overall satisfaction and efficiency of the DialSmart system.

The DialSmart system implements RESTful API principles for client-server communication, with endpoints organized by resource type (phones, users, recommendations, brands). Each API endpoint returns standardized JSON responses with appropriate HTTP status codes to indicate success or error conditions. The system utilizes Flask's blueprint architecture to organize routes logically, promoting code maintainability and scalability.

5.2 Model-View-Controller (MVC) Architecture

The DialSmart system adopts the Model-View-Controller architectural pattern, which provides a structured approach to separating concerns within the application. This separation enhances code maintainability, facilitates team collaboration, and promotes scalability. The MVC pattern divides the application into three interconnected components, each handling specific aspects of the application logic.

5.2.1 Model - Module-Based Organization

The Model in MVC architecture acts as the backbone of the DialSmart application, encapsulating the application's data and business logic across multiple functional modules. It serves as the foundational component responsible for interfacing with the database, managing vital information through well-defined entities and relationships. The Model manages the storage, processing, and integrity of data, defining the rules that govern the application. Acting as the authoritative source of information, the Model communicates with both the View and Controller, ensuring data consistency and facilitating seamless updates in response to changes in the underlying data.

The DialSmart system implements database models using SQLAlchemy ORM (Object-Relational Mapping), which provides an abstraction layer between Python objects and database tables. The models are organized based on functional modules, each addressing specific business requirements:

```
"""
Database Models Package
Contains all database models for DialSmart
"""
from app.models.user import User, UserPreference
from app.models.phone import Phone, PhoneSpecification
from app.models.brand import Brand
from app.models.recommendation import Recommendation, Comparison, ChatHi

__all__ = [
    'User',
    'UserPreference',
    'Phone',
    'PhoneSpecification',
    'Brand',
    'Recommendation',
    'Comparison',
    'ChatHistory'
]
```

Figure 5.2 Database Model

5.2.1.1 Admin Management Module Models

The Admin Management Module encompasses models and functionality for system administration, including admin authentication, dashboard statistics, CRUD operations for phone brands and models, user category management, and system logging. The primary model supporting this module is the User model with administrative privileges.

Admin User Model

The User model includes administrative capabilities through the is_admin flag, enabling administrative access control throughout the system:

```
class User(UserMixin, db.Model):
    """User model for authentication and profile"""
    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
    full_name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False, index
    password_hash = db.Column(db.String(255), nullable=False)

    # User categorization
    user_category = db.Column(db.String(50))  # Student, Working Profess
    age_range = db.Column(db.String(20))  # 18-25, 26-35, 36-45, 46-55,

    # Account settings
    is_admin = db.Column(db.Boolean, default=False)
    is_active = db.Column(db.Boolean, default=True)

    # Timestamps
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    last_active = db.Column(db.DateTime, default=datetime.utcnow)

    # Relationships
    preferences = db.relationship('UserPreference', backref='user', lazy
    recommendations = db.relationship('Recommendation', backref='user',
    comparisons = db.relationship('Comparison', backref='user', lazy='dy
    chat_history = db.relationship('ChatHistory', backref='user', lazy='
```

Figure 5.3 User Model

The is_admin boolean field distinguishes administrative users from regular users, enabling role-based access control. Administrative users can access specialized routes protected by the admin_required decorator. The user_category field supports the categorization of users into different demographic groups (Student, Working Professional, Senior Citizen), which is crucial for generating targeted recommendations and system analytics.

Admin Controller Implementation

The admin module implements comprehensive CRUD operations through controller methods:

```
@bp.route('/phones/add', methods=['GET', 'POST'])
@login_required
@admin_required
def add_phone():
    """Add new phone"""
    if request.method == 'POST':
        # Basic information
        model_name = request.form.get('model_name')
        brand_id = request.form.get('brand_id', type=int)
        price = request.form.get('price', type=float)

        # Create phone
        phone = Phone(
            model_name=model_name,
            brand_id=brand_id,
            price=price,
            model_number=model_number,
            availability_status=availability_status
        )

        db.session.add(phone)
        db.session.flush()  # Get phone ID
```

Figure 5.4 Admin Controller Implementation

This implementation demonstrates the transactional approach to database operations, where both the Phone and PhoneSpecification records are created in a single transaction. The db.session.flush() method ensures the phone ID is available before creating associated specifications, maintaining referential integrity.

Admin Dashboard Analytics

The admin dashboard aggregates system-wide statistics through database queries:

```python
@bp.route('/dashboard')
@login_required
@admin_required
def dashboard():
    """Admin dashboard"""
    # Get statistics
    total_users = User.query.filter_by(is_admin=False).count()
    total_phones = Phone.query.filter_by(is_active=True).count()
    total_brands = Brand.query.filter_by(is_active=True).count()

    # Today's recommendations
    today = datetime.utcnow().date()
    today_recommendations = Recommendation.query.filter(
        db.func.date(Recommendation.created_at) == today
    ).count()

    # Get popular phones (most recommended)
    popular_phones = db.session.query(
        Phone,
        db.func.count(Recommendation.id).label('recommendation_count')
    ).join(Recommendation).group_by(Phone.id)\
     .order_by(db.func.count(Recommendation.id).desc())\
     .limit(5)\
     .all()
```

Figure 5.5 Admin Dashboard Analytics

The dashboard analytics utilize SQLAlchemy's advanced querying capabilities, including aggregate functions (count), joins, and grouping operations to provide real-time insights into system usage and popular smartphones.

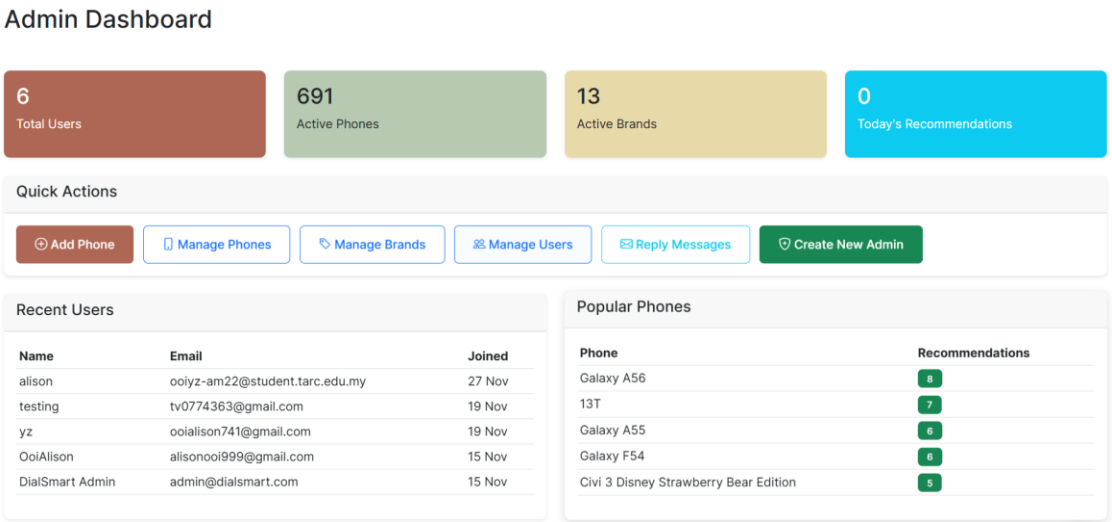

Figure 5.6 Admin Dashborad UI

5.2.1.2 User Management Module Models

The User Management Module handles regular user operations including registration, authentication, profile management, preferences configuration, and viewing past recommendations and comparisons. This module is built on two primary models: User and UserPreference.

User Authentication Model

The User model implements Flask-Login's UserMixin for session management and secure authentication:

```
from app import db, login_manager
from flask_login import UserMixin
from werkzeug.security import generate_password_hash, check_password_has

@login_manager.user_loader
def load_user(user_id):
    """Load user by ID for Flask-Login"""
    return User.query.get(int(user_id))

class User(UserMixin, db.Model):
    """User model for authentication and profile"""
    __tablename__ = 'users'

    def set_password(self, password):
        """Hash and set user password"""
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        """Verify password against hash"""
        return check_password_hash(self.password_hash, password)

    def update_last_active(self):
        """Update last active timestamp"""
        self.last_active = datetime.utcnow()
        db.session.commit()
```

Figure 5.7 User Authentication Model

The model implements secure password handling through Werkzeug's generate_password_hash and check_password_hash functions, which use the pbkdf2:sha256 algorithm. Passwords are never stored in plain text, ensuring security compliance. The user_loader decorator enables Flask-Login to retrieve user objects from user IDs stored in session cookies.

User Registration and Login Implementation

The authentication controller handles user registration with comprehensive validation:

```python
@bp.route('/register', methods=['GET', 'POST'])
def register():
    """User registration page"""
    if current_user.is_authenticated:
        return redirect(url_for('user.dashboard'))

    if request.method == 'POST':
        full_name = request.form.get('full_name')
        email = request.form.get('email')
        password = request.form.get('password')
        confirm_password = request.form.get('confirm_password')
        user_category = request.form.get('user_category')
        age_range = request.form.get('age_range')

        # Validation
        if not all([full_name, email, password, confirm_password]):
            flash('All fields are required.', 'danger')
            return render_template('auth/register.html')

        if password != confirm_password:
            flash('Passwords do not match.', 'danger')
            return render_template('auth/register.html')

        # Check if email already exists
        existing_user = User.query.filter_by(email=email).first()
        if existing_user:
            flash('Email already registered. Please login.', 'warning')
            return redirect(url_for('auth.login'))
```

Figure 5.8 User Login and Registration Implementation

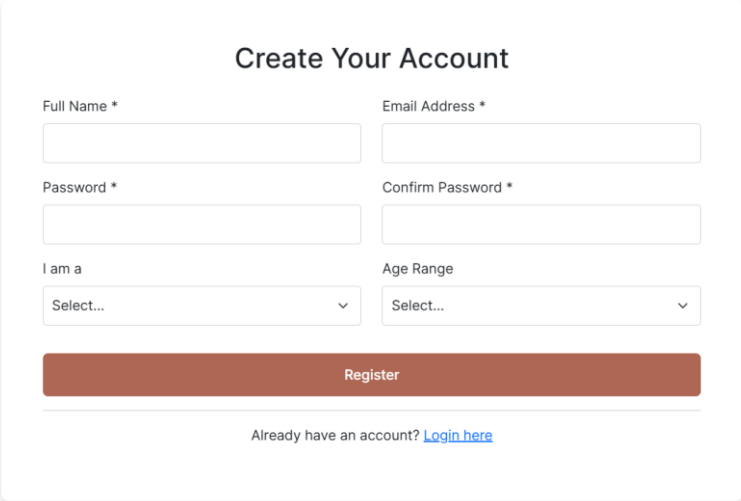

Figure 5.9 User Registration UI

UserPreference Model for Personalization

The UserPreference model stores detailed user preferences for AI-powered recommendations:

```
class UserPreference(db.Model):
    """User preferences for personalized recommendations"""
    __tablename__ = 'user_preferences'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=

    # Budget preferences
    min_budget = db.Column(db.Integer, default=500)
    max_budget = db.Column(db.Integer, default=5000)

    # Usage priorities (JSON stored as string)
    primary_usage = db.Column(db.Text)  # Photography, Gaming, Work, Soc

    # Important features (JSON stored as string)
    important_features = db.Column(db.Text)  # Battery, Camera, Performa

    # Brand preferences (JSON stored as string)
    preferred_brands = db.Column(db.Text)  # List of preferred brand IDs

    # Technical preferences
    min_ram = db.Column(db.Integer, default=4)
    min_storage = db.Column(db.Integer, default=64)
    min_camera = db.Column(db.Integer, default=12)
    min_battery = db.Column(db.Integer, default=3000)
    requires_5g = db.Column(db.Boolean, default=False)

    # Screen preferences
    min_screen_size = db.Column(db.Float, default=5.5)
    max_screen_size = db.Column(db.Float, default=7.0)

    # Updated timestamp
    updated_at = db.Column(db.DateTime, default=datetime.utcnow, onupdat
```

Figure 5.10 UserPreference Model for Personalization

The UserPreference model enables personalized recommendations by storing user-specific criteria including budget range, technical specifications (RAM, storage, camera, battery), 5G requirements, and screen size preferences. JSON fields store complex data structures for usage patterns, important features, and brand preferences, providing flexibility for multi-value selections.

User Profile and Preferences Management

The user controller implements profile editing with password change functionality:

```
@bp.route('/profile', methods=['GET', 'POST'])
@login_required
def profile():
    """User profile page"""
    if request.method == 'POST':
        current_user.full_name = request.form.get('full_name', current_u
        current_user.user_category = request.form.get('user_category', c
        current_user.age_range = request.form.get('age_range', current_u

        # Update password if provided
        new_password = request.form.get('new_password')
        if new_password:
            current_password = request.form.get('current_password')
            if current_user.check_password(current_password):
                current_user.set_password(new_password)
                flash('Password updated successfully.', 'success')
            else:
                flash('Current password is incorrect.', 'danger')
                return render_template('user/profile.html')

        db.session.commit()
        flash('Profile updated successfully.', 'success')
        return redirect(url_for('user.profile'))
```

Figure 5.11 User Profile and Preferences Management

5.2.1.3 Brand & Phone Details Module Models

The Brand & Phone Details Module manages smartphone brands, phone models, detailed specifications, and product categorization. This module implements two primary models: Brand and Phone, along with the PhoneSpecification model for detailed technical attributes.

Brand Model

The Brand model represents smartphone manufacturers and their attributes:

```
"""
Brand Model
Handles smartphone brand information
"""
from app import db
from datetime import datetime

class Brand(db.Model):
    """Brand model for phone manufacturers"""
    __tablename__ = 'brands'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), unique=True, nullable=False, index=True)
    description = db.Column(db.Text)
    tagline = db.Column(db.String(200))

    # Media
    logo_url = db.Column(db.String(255))

    # Status
    is_active = db.Column(db.Boolean, default=True)
    is_featured = db.Column(db.Boolean, default=False)

    # Timestamps
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    updated_at = db.Column(db.DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)

    # Relationships
    phones = db.relationship('Phone', backref='brand', lazy='dynamic')

    def get_phone_count(self):
        """Get number of active phones for this brand"""
        return self.phones.filter_by(is_active=True).count()

    def get_price_range(self):
        """Get min and max price for brand's phones"""
        active_phones = self.phones.filter_by(is_active=True).all()
        if not active_phones:
            return (0, 0)

        prices = [phone.price for phone in active_phones]
        return (min(prices), max(prices))
```

Figure 5.12 Brand Model Implementation

The Brand model includes utility methods get_phone_count() and get_price_range() that provide aggregated information about associated phones. The is_featured flag enables prioritizing certain brands on the homepage. The one-to-many relationship with Phone models uses lazy='dynamic' loading, which returns a query object rather than loading all phones immediately, improving performance for brands with many phone models.

Phone Model with Price Categorization

The Phone model represents individual smartphone products:

```
Phone Model
Handles smartphone data and specifications
"""
from app import db
from datetime import datetime

class Phone(db.Model):
    """Main phone model"""
    __tablename__ = 'phones'

    id = db.Column(db.Integer, primary_key=True)
    brand_id = db.Column(db.Integer, db.ForeignKey('brands.id'), nullable=False)

    # Basic information
    model_name = db.Column(db.String(150), nullable=False, index=True)
    model_number = db.Column(db.String(100))
    price = db.Column(db.Float, nullable=False, index=True)  # Price in MYR

    # Media
    main_image = db.Column(db.String(255))
    gallery_images = db.Column(db.Text)  # JSON string of image URLs

    # Status and availability
    is_active = db.Column(db.Boolean, default=True, index=True)
    availability_status = db.Column(db.String(50), default='Available')  # Available, Out of St
    release_date = db.Column(db.Date)

    # Timestamps
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    updated_at = db.Column(db.DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)

    # Relationships
    specifications = db.relationship('PhoneSpecification', backref='phone', uselist=False, casc

    def get_price_category(self):
        """Categorize phone by price"""
        if self.price < 1000:
            return 'budget'
        elif self.price < 2000:
            return 'mid_range'
        elif self.price < 3000:
            return 'upper_mid'
        else:
            return 'premium'
```

Figure 5.13 Phone Model Implementation

The Phone model implements database indexing on model_name, price, and is_active columns to optimize query performance for common search and filter operations. The get_price_category() method provides automatic categorization based on Malaysian Ringgit price ranges (Budget: <RM1000, Mid-range: RM1000-2000, Upper-mid: RM2000-3000, Premium: >RM3000). The one-to-one relationship with PhoneSpecification uses cascade='all, delete-orphan' to ensure specification records are automatically deleted when the parent phone is removed.

PhoneSpecification Model for Technical Details

The PhoneSpecification model stores comprehensive technical attributes:

```
class PhoneSpecification(db.Model):
    """Detailed phone specifications"""
    __tablename__ = 'phone_specifications'

    id = db.Column(db.Integer, primary_key=True)
    phone_id = db.Column(db.Integer, db.ForeignKey('phones.id'), nullable=False, unique=True)

    # Display specifications
    screen_size = db.Column(db.Float)  # in inches
    screen_resolution = db.Column(db.String(50))  # e.g., "1080x2400"
    screen_type = db.Column(db.String(50))  # AMOLED, IPS LCD, etc.
    refresh_rate = db.Column(db.Integer, default=60)  # Hz

    # Performance specifications
    processor = db.Column(db.String(100))
    processor_brand = db.Column(db.String(50))  # Qualcomm, MediaTek, Apple, etc.
    ram_options = db.Column(db.String(50))  # "4GB, 6GB, 8GB"
    storage_options = db.Column(db.String(50))  # "64GB, 128GB, 256GB"
    expandable_storage = db.Column(db.Boolean, default=False)
```

Figure 5.14 PhoneSpecification Model Implementation

The comprehensive specification model stores over 25 different attributes covering display, performance, camera, battery, connectivity, and physical characteristics. The has_5g field is indexed to support efficient filtering for 5G-capable devices, a key feature for users seeking future-proof smartphones. String fields for options (ram_options, storage_options) allow flexible storage of multiple variants without requiring separate tables.

5.2.1.4 Phone Comparison Module Models

The Phone Comparison Module enables users to compare up to two phones side-by-side with detailed specification comparisons. This module utilizes the Comparison model for storing comparison history and implements the PhoneComparison class for comparison logic..

Comparison Model for History Tracking

```
class Comparison(db.Model):
    """Phone comparison history"""
    __tablename__ = 'comparisons'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

    # Phones being compared
    phone1_id = db.Column(db.Integer, db.ForeignKey('phones.id'), nullable=False)
    phone2_id = db.Column(db.Integer, db.ForeignKey('phones.id'), nullable=False)

    # Comparison metadata
    is_saved = db.Column(db.Boolean, default=False)
    comparison_notes = db.Column(db.Text)

    # Timestamp
    created_at = db.Column(db.DateTime, default=datetime.utcnow, index=True)

    # Relationships
    phone1 = db.relationship('Phone', foreign_keys=[phone1_id], backref='comparisons_as_phone1')
    phone2 = db.relationship('Phone', foreign_keys=[phone2_id], backref='comparisons_as_phone2')
```

Figure 5.15 Comparison Model for History Tracking

The Comparison model implements dual foreign key relationships to the Phone model, enabling storage of comparison pairs. The is_saved flag allows users to bookmark important comparisons for future reference. The model uses foreign_keys parameter in relationships to explicitly specify which foreign key column corresponds to each relationship, resolving the ambiguity of having two references to the same table.

Phone Comparison Logic Implementation

The PhoneComparison class implements comprehensive comparison functionality:

```
def compare_phones(self, phone1_id, phone2_id, user_id=None):
    """
    Compare two phones side-by-side

    Args:
        phone1_id: ID of first phone
        phone2_id: ID of second phone
        user_id: Optional user ID to save comparison

    Returns:
        Dictionary with comparison data
    """
    phone1 = Phone.query.get(phone1_id)
    phone2 = Phone.query.get(phone2_id)

    if not phone1 or not phone2:
        return None

    specs1 = PhoneSpecification.query.filter_by(phone_id=phone1_id).first()
    specs2 = PhoneSpecification.query.filter_by(phone_id=phone2_id).first()

    # Build comparison data
    comparison_data = {
        'phone1': {
            'info': phone1,
            'specs': specs1
        },
        'phone2': {
            'info': phone2,
            'specs': specs2
        },
        'comparison': self._build_comparison_table(phone1, specs1, phone2, specs2),
        'winner': self._determine_winner(phone1, specs1, phone2, specs2)
    }

    # Save comparison if user_id provided
    if user_id:
        self._save_comparison(user_id, phone1_id, phone2_id)

    return comparison_data
```

Figure 5.16 Phone Comparison Logic Implementation

Comparison Table Building

The comparison module creates structured comparison data across multiple categories:

```
def _build_comparison_table(self, phone1, specs1, phone2, specs2):
    """Build detailed comparison table"""
    comparison = {
        'price': {
            'label': 'Price',
            'phone1': f"RM {phone1.price:,.2f}",
            'phone2': f"RM {phone2.price:,.2f}",
            'winner': 1 if phone1.price < phone2.price else 2,
            'difference': abs(phone1.price - phone2.price)
        }
    }

    if specs1 and specs2:
        # Camera comparisons
        comparison['rear_camera'] = {
            'label': 'Rear Camera',
            'phone1': specs1.rear_camera or 'N/A',
            'phone2': specs2.rear_camera or 'N/A',
            'winner': 1 if (specs1.rear_camera_main or 0) > (specs2.rear_camera_main or 0) else
        }

        # Battery comparisons
        comparison['battery'] = {
            'label': 'Battery Capacity',
            'phone1': f"{specs1.battery_capacity}mAh" if specs1.battery_capacity else 'N/A',
            'phone2': f"{specs2.battery_capacity}mAh" if specs2.battery_capacity else 'N/A',
            'winner': 1 if (specs1.battery_capacity or 0) > (specs2.battery_capacity or 0) else
        }

        # 5G Support
        comparison['5g'] = {
            'label': '5G Support',
            'phone1': '✓ Yes' if specs1.has_5g else '✗ No',
            'phone2': '✓ Yes' if specs2.has_5g else '✗ No',
            'winner': 1 if specs1.has_5g else 2 if specs2.has_5g else None
        }

    return comparison
```

Figure 5.17 Comparison Table Building

The comparison table structure provides human-readable formatted values alongside raw data for winner determination. Each comparison item includes label, values for both phones, winner indication, and optional difference calculation. The system uses conditional expressions to handle missing data gracefully, displaying 'N/A' when specifications are unavailable.

Winner Determination Algorithm

```python
def _determine_winner(self, phone1, specs1, phone2, specs2):
    """Determine overall winner based on multiple factors"""
    phone1_score = 0
    phone2_score = 0

    # Price (lower is better)
    if phone1.price < phone2.price:
        phone1_score += 1
    else:
        phone2_score += 1

    if specs1 and specs2:
        # Camera
        if (specs1.rear_camera_main or 0) > (specs2.rear_camera_main or 0):
            phone1_score += 1
        elif (specs2.rear_camera_main or 0) > (specs1.rear_camera_main or 0):
            phone2_score += 1

        # Battery
        if (specs1.battery_capacity or 0) > (specs2.battery_capacity or 0):
            phone1_score += 1
        elif (specs2.battery_capacity or 0) > (specs1.battery_capacity or 0):
            phone2_score += 1

        # 5G
        if specs1.has_5g and not specs2.has_5g:
            phone1_score += 1
        elif specs2.has_5g and not specs1.has_5g:
            phone2_score += 1

    if phone1_score > phone2_score:
        return {'phone': 1, 'name': phone1.model_name, 'score': phone1_score}
    elif phone2_score > phone1_score:
        return {'phone': 2, 'name': phone2.model_name, 'score': phone2_score}
    else:
        return {'phone': None, 'name': 'Tie', 'score': phone1_score}
```

Figure 5.18 Winner Determination Algorithm

The winner determination algorithm assigns points across multiple categories including price (lower is better), camera quality, battery capacity, 5G support, and screen refresh rate. The scoring system provides an objective comparison while allowing for ties when phones are evenly matched.

5.2.1.5 AI Chatbot Module Models

The AI Chatbot Module implements natural language processing capabilities for conversational phone recommendations. The ChatHistory model stores all chat interactions for context maintenance and conversation analysis.

ChatHistory Model

```
class ChatHistory(db.Model):
    """Chatbot conversation history"""
    __tablename__ = 'chat_history'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

    # Message details
    message = db.Column(db.Text, nullable=False)
    response = db.Column(db.Text, nullable=False)
    message_type = db.Column(db.String(20), default='text')  # text, recommendation, comparison

    # Context
    session_id = db.Column(db.String(100))  # To group conversations
    intent = db.Column(db.String(100))  # Detected user intent

    # Metadata (JSON stored as string)
    metadata = db.Column(db.Text)  # Additional context like recommended phone IDs

    # Timestamp
    created_at = db.Column(db.DateTime, default=datetime.utcnow, index=True)
```

5

Figure 5.19 ChatHistory Model

The ChatHistory model stores complete conversation turns including user messages and bot responses. The session_id field groups related conversations, enabling context-aware responses across multiple message exchanges. The intent field records the detected user intention (greeting, budget_query, recommendation, comparison, etc.), which is crucial for improving the chatbot's natural language understanding over time through analysis of historical data.

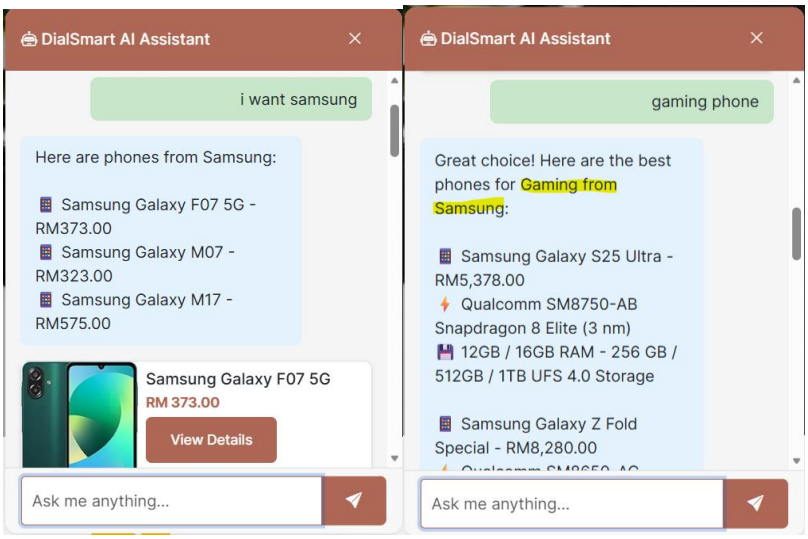

Figure 5.20 Chatbot context response

Chatbot Message Processing

The chatbot engine implements intent detection and response generation:

```
class ChatbotEngine:
    """Conversational AI chatbot for DialSmart"""

    def __init__(self):
        self.ai_engine = AIRecommendationEngine()
        self.intents = {
            'greeting': ['hello', 'hi', 'hey', 'good morning', 'good afternoon'],
            'budget_query': ['budget', 'price', 'cost', 'cheap', 'affordable', 'expensive', 'rm
            'recommendation': ['recommend', 'suggest', 'find', 'looking for', 'need', 'want'],
            'comparison': ['compare', 'difference', 'vs', 'versus', 'better'],
            'specification': ['specs', 'specification', 'camera', 'battery', 'ram', 'storage',
            'brand_query': ['brand', 'samsung', 'apple', 'iphone', 'xiaomi', 'huawei'],
            'help': ['help', 'how', 'what can you do'],
            'usage_type': ['gaming', 'photography', 'camera', 'business', 'work', 'social media
        }

    def process_message(self, user_id, message, session_id=None):
        """Process user message and generate response"""
        # Detect intent
        intent = self._detect_intent(message.lower())

        # Generate response based on intent
        response_data = self._generate_response(user_id, message, intent)

        # Save to chat history
        self._save_chat_history(
            user_id=user_id,
            message=message,
            response=response_data['response'],
            intent=intent,
            session_id=session_id,
            metadata=response_data.get('metadata', {})
        )

        return response_data
```

Figure 5.21 Chatbot Message Processing Implementation

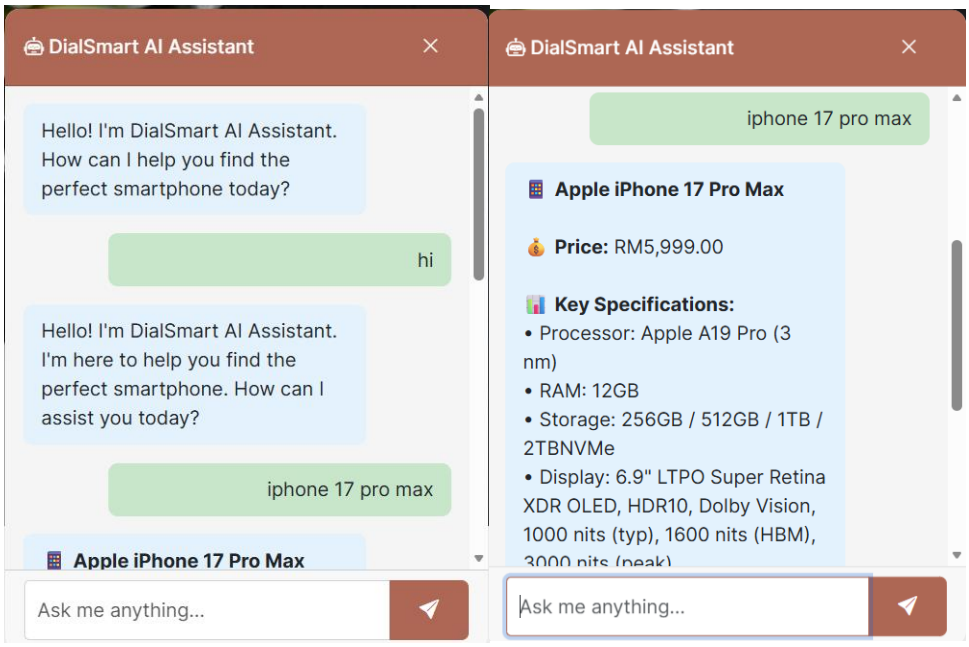

Figure 5.22 Chatbot Message Processing

Natural Language Processing for Budget Extraction

The chatbot implements pattern matching for extracting structured data from natural language:

```python
def _extract_budget(self, message):
    """Extract budget range from message"""
    # Look for patterns like "RM1000", "1000", "under 2000", "between 1000 and 2000"
    patterns = [
        r'rm\s*(\d+)\s*(?:to|-|and)\s*rm\s*(\d+)',  # RM1000 to RM2000
        r'(\d+)\s*(?:to|-|and)\s*(\d+)',  # 1000 to 2000
        r'under\s*rm?\s*(\d+)',  # under RM2000
        r'below\s*rm?\s*(\d+)',  # below 2000
        r'rm\s*(\d+)',  # RM2000
    ]

    for pattern in patterns:
        match = re.search(pattern, message.lower())
        if match:
            if 'under' in message.lower() or 'below' in message.lower():
                max_budget = int(match.group(1))
                return (500, max_budget)
            elif len(match.groups()) == 2:
                return (int(match.group(1)), int(match.group(2)))
            else:
                value = int(match.group(1))
                return (500, value)

    return None
```

Figure 5.23 Natural Language Processing (Chatbot)

The budget extraction function uses regular expressions to parse various natural language formats including "RM1000 to RM2000", "under 2000", "between 1000 and 2000", enabling flexible user input while maintaining structured data for database queries.

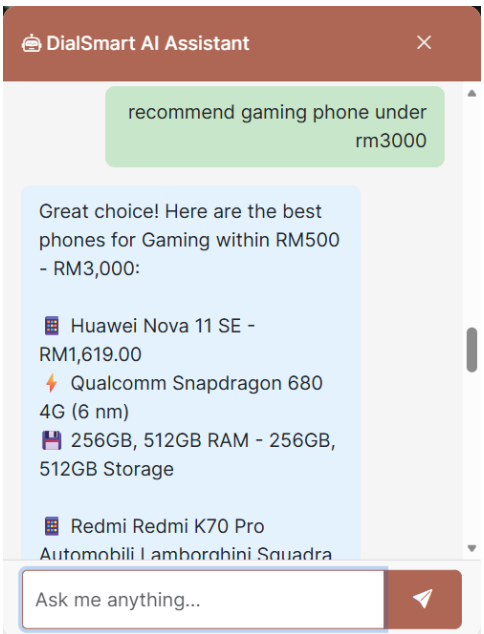

Figure 5.24 Chatbot Natural Language Format Response

Chatbot Response Generation

```
def _generate_response(self, user_id, message, intent):
    """Generate appropriate response based on intent"""

    if intent == 'budget_query':
        # Extract budget from message
        budget = self._extract_budget(message)
        if budget:
            min_budget, max_budget = budget
            phones = self.ai_engine.get_budget_recommendations((min_budget, max_budget), top_n=

            if phones:
                response = f"Here are the top phones within RM{min_budget} - RM{max_budget}:\n\
                phone_list = []
                for item in phones:
                    phone = item['phone']
                    response += f"  {phone.model_name} - RM{phone.price:,.2f}\n"
                    phone_list.append({
                        'id': phone.id,
                        'name': phone.model_name,
                        'price': phone.price
                    })

                return {
                    'response': response,
                    'type': 'recommendation',
                    'metadata': {'phones': phone_list}
                }
```

Figure 5.25 Chatbot Response Generation

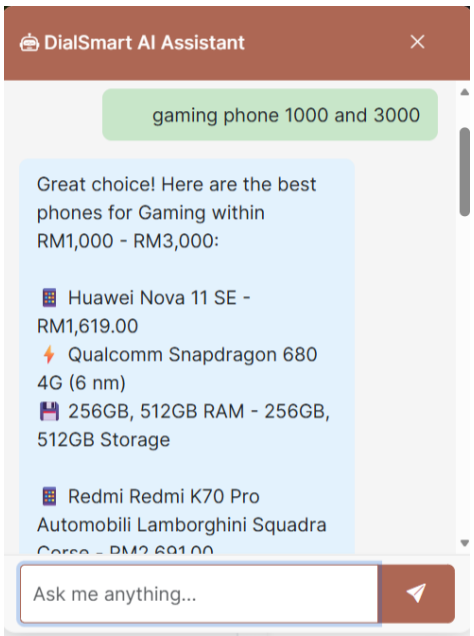

Figure 5.26 Chatbot Response Generation Response

5.2.1.6 Recommendation System Core Logic Models

The Recommendation System Core Logic implements AI-powered phone recommendations based on user preferences, usage patterns, and technical requirements. The Recommendation model stores historical recommendations with match scores and reasoning.

Recommendation Model

```python
class Recommendation(db.Model):
    """User recommendation history"""
    __tablename__ = 'recommendations'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    phone_id = db.Column(db.Integer, db.ForeignKey('phones.id'), nullable=False)

    # Recommendation details
    match_percentage = db.Column(db.Float)  # 0-100
    reasoning = db.Column(db.Text)  # Why this phone was recommended

    # User input parameters (JSON stored as string)
    user_criteria = db.Column(db.Text)  # The criteria used for this recommendation

    # User feedback
    is_viewed = db.Column(db.Boolean, default=False)
    is_saved = db.Column(db.Boolean, default=False)
    user_rating = db.Column(db.Integer)  # 1-5 stars

    # Timestamp
    created_at = db.Column(db.DateTime, default=datetime.utcnow, index=True)

    # Relationships
    phone = db.relationship('Phone', backref='recommendations')
```

Figure 5.27 Recommendation Model

The Recommendation model stores not only the recommendation result but also the match percentage (0-100) and human-readable reasoning explaining why the phone was recommended. The user_criteria field stores the original search parameters as JSON, enabling analysis of recommendation accuracy and system improvement over time. User feedback fields (is_viewed, is_saved, user_rating) facilitate recommendation quality assessment and algorithm refinement.

AI Recommendation Engine Implementation

```python
from app import db
from app.models import Phone, PhoneSpecification, UserPreference, Recommendation
from app.utils.helpers import calculate_match_score, generate_recommendation_reasoning
import json

class AIRecommendationEngine:
    """AI-powered recommendation engine for smartphones"""

    def __init__(self):
        self.min_match_threshold = 50  # Minimum match percentage to recommend

    def get_recommendations(self, user_id, criteria=None, top_n=3):
        """
        Get top N phone recommendations for a user

        Args:
            user_id: User ID to get recommendations for
            criteria: Optional dictionary of criteria to override user preferences
            top_n: Number of recommendations to return

        Returns:
            List of recommended phones with match scores
        """
        from app.models import User

        user = User.query.get(user_id)
        if not user:
            return []

        # Get or create user preferences
        user_prefs = UserPreference.query.filter_by(user_id=user_id).first()

        # If criteria provided, create temporary preference object
        if criteria:
            user_prefs = self._create_temp_preferences(criteria)
        elif not user_prefs:
            user_prefs = self._create_default_preferences(user_id)

        # Get all active phones with their specifications
        phones = Phone.query.filter_by(is_active=True).all()
```

Figure 5.28 AI Recommendation Engine Implementation

The recommendation engine implements a filtering approach where all active phones are evaluated against user preferences, with only matches exceeding the 50% threshold being returned. This ensures recommendation quality while maintaining sufficient result diversity.

Match Score Calculation Algorithm

```python
def calculate_match_score(user_prefs, phone, phone_specs):
    """
    Calculate how well a phone matches user preferences
    Returns a score from 0-100
    """
    score = 0
    max_score = 0

    # Budget match (weight: 30)
    max_score += 30
    if user_prefs.min_budget <= phone.price <= user_prefs.max_budget:
        score += 30
    elif phone.price < user_prefs.min_budget:
        score += 20
    else:
        over_budget = phone.price - user_prefs.max_budget
        penalty = min(30, (over_budget / user_prefs.max_budget) * 30)
        score += max(0, 30 - penalty)

    if phone_specs:
        # RAM match (weight: 10)
        max_score += 10
        if phone_specs.ram_options:
            ram_values = [int(r.replace('GB', '')) for r in phone_specs.ram_
            if ram_values and max(ram_values) >= user_prefs.min_ram:
                score += 10

        # Camera match (weight: 15)
        max_score += 15
        if phone_specs.rear_camera_main and phone_specs.rear_camera_main >=
            score += 15

        # Battery match (weight: 15)
        max_score += 15
        if phone_specs.battery_capacity and phone_specs.battery_capacity >=
            score += 15

        # 5G requirement (weight: 10)
        max_score += 10
        if user_prefs.requires_5g:
            if phone_specs.has_5g:
                score += 10
        else:
            score += 10  # No requirement, so full points

    # Calculate final percentage
    if max_score > 0:
        return round((score / max_score) * 100, 2)
    return 0
```

Figure 5.29 Match Score Calculation Algorithm

The match score algorithm implements weighted scoring across six key factors: Budget (30%), Camera (15%), Battery (15%), RAM (10%), 5G Support (10%), and Screen Size (10%). The budget component implements graduated scoring: full points for within-budget phones, partial credit for cheaper phones, and proportional penalties for over-budget options. This nuanced approach prevents artificial boundary effects where a phone RM1 over budget would score dramatically worse than one at the exact budget limit.

Recommendation Reasoning Generation

```
def generate_recommendation_reasoning(match_score, user_prefs, phone, phone_specs):
    """Generate human-readable reasoning for recommendation"""
    reasons = []

    # Budget
    if user_prefs.min_budget <= phone.price <= user_prefs.max_budget:
        reasons.append(f"Within your budget of {format_price(user_prefs.min_budget)

    if phone_specs:
        # Performance
        if phone_specs.ram_options:
            ram_values = [int(r.replace('GB', '')) for r in phone_specs.ram_options
            if ram_values:
                max_ram = max(ram_values)
                if max_ram >= user_prefs.min_ram:
                    reasons.append(f"Excellent performance with up to {max_ram}GB F

        # Camera
        if phone_specs.rear_camera_main and phone_specs.rear_camera_main >= user_pr
            reasons.append(f"Great {phone_specs.rear_camera_main}MP camera for phot

        # Battery
        if phone_specs.battery_capacity and phone_specs.battery_capacity >= user_pr
            reasons.append(f"Long-lasting {phone_specs.battery_capacity}mAh battery

        # 5G
        if phone_specs.has_5g:
            reasons.append("Future-ready with 5G connectivity")

    if not reasons:
        reasons.append("Good overall specifications for the price")

    return " • ".join(reasons)
```

Figure 5.30 Recommendation Reasoning Generation

The reasoning generation function creates human-readable explanations for why each phone was recommended, highlighting the specific features that match user preferences. This transparency builds user trust in the recommendation system and helps users understand the trade-offs between different options.

5.2.1.7 Model Relationships and Database Integrity

The DialSmart database schema implements comprehensive relationships between models ensuring referential integrity and cascade operations:

```
# User relationships
preferences = db.relationship('UserPreference', backref='user', lazy='dynamic', cascade='all, delete-or
recommendations = db.relationship('Recommendation', backref='user', lazy='dynamic', cascade='all, delet
comparisons = db.relationship('Comparison', backref='user', lazy='dynamic', cascade='all, delete-orphan
chat_history = db.relationship('ChatHistory', backref='user', lazy='dynamic', cascade='all, delete-orph

# Brand relationships
phones = db.relationship('Phone', backref='brand', lazy='dynamic')

# Phone relationships
specifications = db.relationship('PhoneSpecification', backref='phone', uselist=False, cascade='all, de
```

Figure 5.31 Model Relationships and Database Integrity

The cascade options (cascade='all, delete-orphan') ensure that when a user is deleted, all associated preferences, recommendations, comparisons, and chat history are automatically removed, maintaining database consistency. The lazy='dynamic' loading strategy optimizes performance by returning query objects rather than loading related records immediately, particularly beneficial for users with extensive recommendation or comparison histories.

Conclusion

The Model layer of the DialSmart system demonstrates comprehensive implementation of database-driven application architecture across six major functional modules. Each model is carefully designed with appropriate data types, indexes, relationships, and business logic methods. The module-based organization (Admin Management, User Management, Brand & Phone Details, Phone Comparison, AI Chatbot, and Recommendation System) provides clear separation of concerns while maintaining strong inter-module relationships through foreign keys and SQLAlchemy relationships. The implementation leverages SQLAlchemy ORM capabilities including lazy loading, cascade operations, and complex queries to deliver efficient, maintainable, and scalable data management for the DialSmart AI-powered smartphone recommendation system..

5.2.2 View

In the MVC paradigm, the View takes center stage in presenting an intuitive and visually appealing user interface. The View showcases information such as personalized smartphone recommendations, detailed phone specifications, brand information, and comparison results in a manner that is both comprehensible and aesthetically pleasing. Importantly, the View remains independent of the underlying business logic, ensuring a clear separation of concerns. Users engaging with the application can seamlessly navigate and interact with the presented information, enhancing the overall user experience while maintaining a focused and user-friendly presentation.

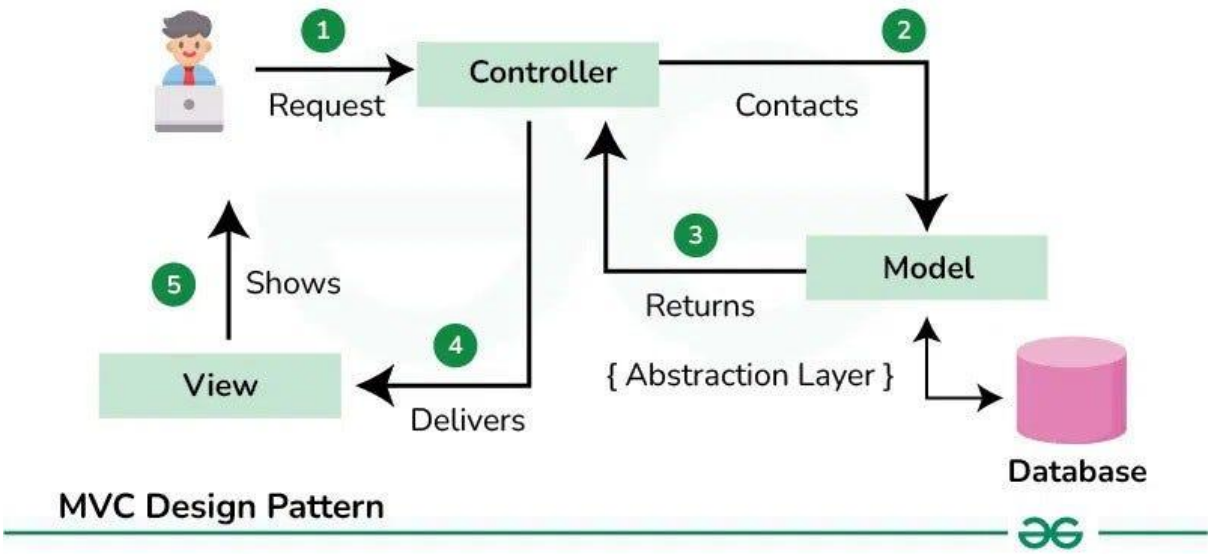

Figure 5.32 MVC Design Pattern

The DialSmart system utilizes Jinja2 templating engine, which is integrated with Flask, to generate dynamic HTML content. The templates are organized hierarchically with a base template providing common structure and child templates extending it for specific pages:

```
<!-- base.html - Main template structure -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}DialSmart - AI Smartphone Recommendations{% endb

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst

    <!-- Custom CSS -->
    <link href="{{ url_for('static', filename='css/style.css') }}" rel="styl

    {% block extra_css %}{% endblock %}
</head>
<body>
    <!-- Navigation Bar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="/">
                <i class="fas fa-mobile-alt"></i> DialSmart
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="col
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    {% if current_user.is_authenticated %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('user.dasht
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('user.recom
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('phone.brow
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('auth.logou
                        </li>
                    {% else %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('auth.logir
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('auth.regis
                        </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>
```

Figure 5.33 Main Template Structure

```
    <!-- Flash Messages -->
    {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
            <div class="container mt-3">
                {% for category, message in messages %}
                    <div class="alert alert-{{ category }} alert-dismissible
                        {{ message }}
                        <button type="button" class="btn-close" data-bs-dism
                    </div>
                {% endfor %}
            </div>
        {% endif %}
    {% endwith %}

    <!-- Main Content -->
    <main class="container my-4">
        {% block content %}{% endblock %}
    </main>

    <!-- Footer -->
    <footer class="bg-dark text-white text-center py-4 mt-5">
        <div class="container">
            <p>&copy; 2024 DialSmart. All rights reserved.</p>
        </div>
    </footer>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootst

    {% block extra_js %}{% endblock %}
</body>
</html>
```

Figure 5.33 Main Template Structure (continued)

The View layer implements responsive design using Bootstrap 5 framework, ensuring compatibility across various device sizes. Dynamic content is rendered using Jinja2 template variables and control structures, allowing server-side data to be seamlessly integrated into the HTML presentation.

5.2.3 Controller

The Controller acts as a pivotal intermediary in the DialSmart system, managing user interactions and orchestrating the flow of information between the Model and View. When users place orders, request recommendations, or engage in other actions, the Controller processes these commands and coordinates the appropriate responses. Simultaneously, the Controller updates the Model to reflect changes in data and triggers adjustments in the View to provide feedback to users. To enhance the adaptability of the system, the Controller efficiently handles interactions by passing back responses with JSON data, allowing for efficient and standardized communication through the API endpoints.

The DialSmart system implements Controllers as Flask Blueprints, which provide modular organization of routes and handlers. Each blueprint focuses on a specific domain of functionality:

Authentication Controller

The authentication controller manages user registration, login, and logout operations:

```python
"""
Authentication Routes
Handles user registration, login, and logout
"""
from flask import Blueprint, render_template, redirect, url_for, request, flash
from flask_login import login_user, logout_user, login_required, current_user
from app import db
from app.models import User

bp = Blueprint('auth', __name__, url_prefix='/auth')

@bp.route('/register', methods=['GET', 'POST'])
def register():
    """User registration page"""
    if current_user.is_authenticated:
        return redirect(url_for('user.dashboard'))

    if request.method == 'POST':
        full_name = request.form.get('full_name')
        email = request.form.get('email')
        password = request.form.get('password')
        confirm_password = request.form.get('confirm_password')
        user_category = request.form.get('user_category')
        age_range = request.form.get('age_range')

        # Validation
        if not all([full_name, email, password, confirm_password]):
            flash('All fields are required.', 'danger')
            return render_template('auth/register.html')

        if password != confirm_password:
            flash('Passwords do not match.', 'danger')
            return render_template('auth/register.html')

        # Check if email already exists
        existing_user = User.query.filter_by(email=email).first()
        if existing_user:
            flash('Email already registered. Please login.', 'warning')
            return redirect(url_for('auth.login'))
```

Figure 5.34 Authentication Controller

```python
@bp.route('/login', methods=['GET', 'POST'])
def login():
    """User login page"""
    if current_user.is_authenticated:
        if current_user.is_admin:
            return redirect(url_for('admin.dashboard'))
        return redirect(url_for('user.dashboard'))

    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')
        remember = request.form.get('remember', False)

        user = User.query.filter_by(email=email).first()

        if user and user.check_password(password):
            if not user.is_active:
                flash('Your account has been suspended. Please contact support.', 'danger')
                return render_template('auth/login.html')

            login_user(user, remember=remember)
            user.update_last_active()

            # Redirect based on user type
            next_page = request.args.get('next')
            if next_page:
                return redirect(next_page)
            elif user.is_admin:
                return redirect(url_for('admin.dashboard'))
            else:
                return redirect(url_for('user.dashboard'))
        else:
            flash('Invalid email or password.', 'danger')

    return render_template('auth/login.html')

@bp.route('/logout')
@login_required
def logout():
    """Logout user"""
    logout_user()
    flash('You have been logged out successfully.', 'info')
    return redirect(url_for('user.index'))
```

Figure 5.34 Authentication Controller (continued)

The authentication controller implements secure login mechanisms using Flask-Login extension, which manages user sessions and provides decorators for protecting routes that require authentication.

API Controller

The API controller provides RESTful endpoints for AJAX requests and returns JSON responses:

```python
"""
API Routes
RESTful API endpoints for AJAX requests and chatbot
"""
from flask import Blueprint, jsonify, request
from flask_login import login_required, current_user
from app.models import Phone, PhoneSpecification, Brand
from app.modules import ChatbotEngine, AIRecommendationEngine
import uuid

bp = Blueprint('api', __name__, url_prefix='/api')

@bp.route('/recommendations', methods=['POST'])
@login_required
def get_recommendations():
    """Get AI recommendations"""
    data = request.get_json()
    criteria = data.get('criteria', {})
    top_n = data.get('top_n', 3)

    ai_engine = AIRecommendationEngine()
    recommendations = ai_engine.get_recommendations(
        current_user.id,
        criteria=criteria if criteria else None,
        top_n=top_n
    )

    rec_list = []
    for rec in recommendations:
        phone = rec['phone']
        specs = rec['specifications']

        rec_data = {
            'phone_id': phone.id,
            'model_name': phone.model_name,
            'brand': phone.brand.name if phone.brand else 'Unknown',
            'price': phone.price,
            'match_score': rec['match_score'],
            'reasoning': rec['reasoning'],
            'main_image': phone.main_image
        }

        if specs:
            rec_data['key_specs'] = {
                'ram': specs.ram_options,
                'storage': specs.storage_options,
                'camera': f"{specs.rear_camera_main}MP" if specs.rear_camera_main else 'N/A',
                'battery': f"{specs.battery_capacity}mAh" if specs.battery_capacity else 'N/A'
            }

        rec_list.append(rec_data)

    return jsonify({
        'success': True,
        'recommendations': rec_list
    })
```

Figure 5.35 API Controller

```python
@bp.route('/phones/search', methods=['GET'])
def search_phones():
    """Search phones (for autocomplete)"""
    query = request.args.get('q', '')
    limit = request.args.get('limit', 10, type=int)

    if not query:
        return jsonify({'phones': []})

    phones = Phone.query.filter(
        Phone.is_active == True,
        Phone.model_name.ilike(f'%{query}%')
    ).limit(limit).all()

    phone_list = [{
        'id': phone.id,
        'name': phone.model_name,
        'brand': phone.brand.name if phone.brand else 'Unknown',
        'price': phone.price,
        'image': phone.main_image
    } for phone in phones]

    return jsonify({
        'success': True,
        'phones': phone_list
    })
```

Figure 5.35 API Controller(continued)

The API controller implements RESTful principles, returning standardized JSON responses with appropriate HTTP status codes. Authentication is enforced using the @login_required decorator for sensitive endpoints.

5.3 Database Configuration and Management

The DialSmart system utilizes SQLAlchemy as the Object-Relational Mapping (ORM) layer, providing an abstraction between Python objects and database operations. The system supports multiple database backends through configuration.

5.3.1 Configuration Management

The application implements environment-specific configurations using a configuration class hierarchy:

```
"""
DialSmart Configuration File
Manages all application settings and configurations
"""
import os
from datetime import timedelta

class Config:
    """Base configuration"""
    # Application settings
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'dialsmart-secret-key-2024'

    # Database settings
    BASE_DIR = os.path.abspath(os.path.dirname(__file__))
    SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
        'oracle+cx_oracle://username:password@localhost:1521/dialsmart'
    SQLALCHEMY_TRACK_MODIFICATIONS = False

    # Session settings
    PERMANENT_SESSION_LIFETIME = timedelta(days=7)

    # Upload settings
    UPLOAD_FOLDER = os.path.join(BASE_DIR, 'app', 'static', 'uploads')
    MAX_CONTENT_LENGTH = 16 * 1024 * 1024  # 16MB max file size
    ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

    # Pagination
    ITEMS_PER_PAGE = 12
    ADMIN_ITEMS_PER_PAGE = 20

    # AI Model settings
    AI_MODEL_PATH = os.path.join(BASE_DIR, 'models', 'recommendation_model.pkl')

    # Malaysian Ringgit price ranges
    PRICE_RANGES = {
        'budget': (0, 1000),
        'mid_range': (1000, 2000),
        'upper_mid': (2000, 3000),
        'premium': (3000, 10000)
    }
```

Figure 5.36 Configuration Management

```
class DevelopmentConfig(Config):
    """Development configuration"""
    DEBUG = True
    TESTING = False

class ProductionConfig(Config):
    """Production configuration"""
    DEBUG = False
    TESTING = False

class TestingConfig(Config):
    """Testing configuration"""
    TESTING = True
    SQLALCHEMY_DATABASE_URI = 'oracle+cx_oracle://test_user:test_pass@localhost:1521/test_db'

# Configuration dictionary
config = {
    'development': DevelopmentConfig,
    'production': ProductionConfig,
    'testing': TestingConfig,
    'default': DevelopmentConfig
}
```

Figure 5.36 Configuration Management (continued)

The configuration system supports environment variables for sensitive data like SECRET_KEY and DATABASE_URL, following best practices for secure application deployment.

5.3.2 Application Factory Pattern

The system implements the application factory pattern for flexible initialization:

```
"""
DialSmart Application Factory
Initializes and configures the Flask application
"""
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager
from config import config
import os

# Initialize extensions
db = SQLAlchemy()
login_manager = LoginManager()

def create_app(config_name='default'):
    """Application factory pattern"""
    app = Flask(__name__)

    # Load configuration
    app.config.from_object(config[config_name])

    # Initialize extensions
    db.init_app(app)
    login_manager.init_app(app)
    login_manager.login_view = 'auth.login'
    login_manager.login_message = 'Please log in to access this page.'

    # Create upload folder if it doesn't exist
    os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

    # Register blueprints
    from app.routes import auth, user, admin, phone, api
    app.register_blueprint(auth.bp)
    app.register_blueprint(user.bp)
    app.register_blueprint(admin.bp)
    app.register_blueprint(phone.bp)
    app.register_blueprint(api.bp)

    # Register context processors
    @app.context_processor
    def inject_brands():
        """Inject featured brands into all templates"""
        return dict(featured_brands=app.config['FEATURED_BRANDS'])

    # Create database tables
    with app.app_context():
        db.create_all()
```

Figure 5.37 Application Factory Pattern

The application factory pattern enables testing with different configurations and facilitates modular application structure through blueprint registration.

5.4 Security Implementation

The DialSmart system implements multiple security mechanisms to protect user data and prevent common web vulnerabilities.

5.4.1 Password Security

User passwords are hashed using Werkzeug's security utilities with the pbkdf2:sha256 algorithm:

```python
from werkzeug.security import generate_password_hash, check_password_hash

def set_password(self, password):
    """Hash and set user password"""
    self.password_hash = generate_password_hash(password)

def check_password(self, password):
    """Verify password against hash"""
    return check_password_hash(self.password_hash, password)
```

Figure 5.38 Password Security

```
ID FULL_NAME                EMAIL                           PASSWORD_HASH
-- ------------------------ ------------------------------- ----------------------------------------------------------------------------------------------------
41 OYZAdmin                 alison99690326@gmail.com        scrypt:32768:8:1$8IMeA08Ric5UHNSN$6f8c0b66728a793fe7ecfd010028a73014a3a9282b7faa276409567de6477bf5db2c174d49ca88e752b19c50bc76a
```

Figure 5.39 Password hash in database

This implementation ensures that plain-text passwords are never stored in the database, protecting user credentials even in the event of a database breach.

5.4.2 Session Management

Flask-Login manages user sessions securely with configurable session lifetime:

```
# Session settings
PERMANENT_SESSION_LIFETIME = timedelta(days=7)

# Login manager configuration
login_manager.login_view = 'auth.login'
login_manager.login_message = 'Please log in to access this page.'
```

Figure 5.40 Session Management

The @login_required decorator protects routes that require authentication, automatically redirecting unauthenticated users to the login page.

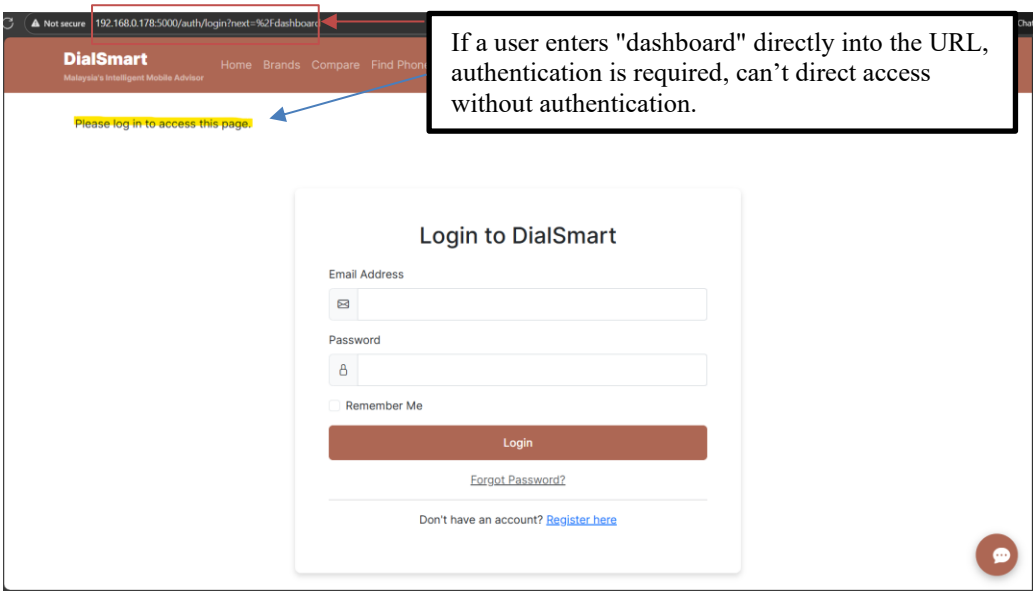

Figure 5.41 Session Management UI

5.4.3 Input Validation and Sanitization

The system implements input validation for user registration and form submissions:

```
# Validation
if not all([full_name, email, password, confirm_password]):
    flash('All fields are required.', 'danger')
    return render_template('auth/register.html')

if password != confirm_password:
    flash('Passwords do not match.', 'danger')
    return render_template('auth/register.html')

# Check if email already exists
existing_user = User.query.filter_by(email=email).first()
if existing_user:
    flash('Email already registered. Please login.', 'warning')
    return redirect(url_for('auth.login'))
```

Figure 5.42 Input Validation and Sanitization

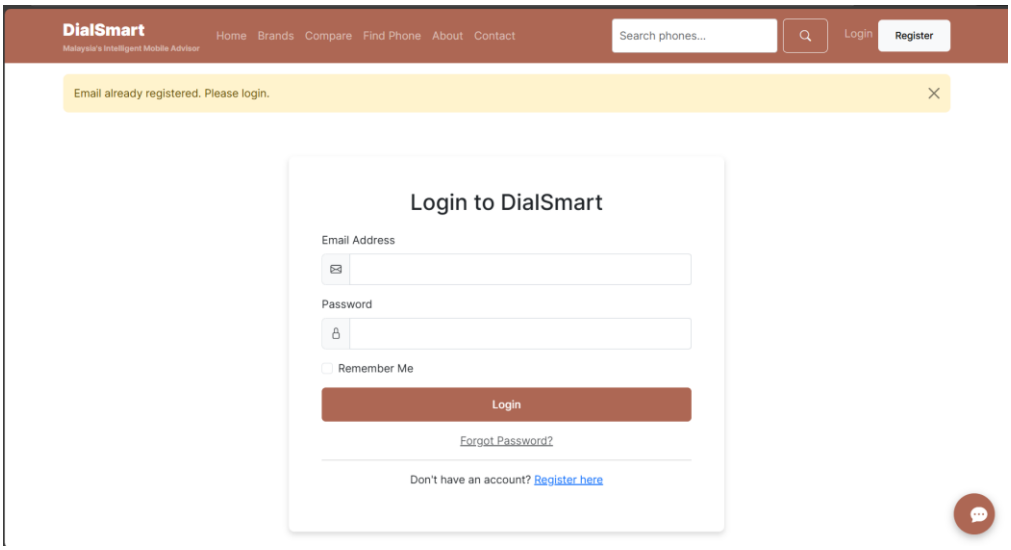

Figure 5.43 Email already exists check

5.4.4 File Upload Security

Secure filename handling and file type validation protect against malicious file uploads:

```python
def allowed_file(filename):
    """Check if file extension is allowed"""
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in current_app.config['ALLOWED_EXTENSIONS']

def save_uploaded_file(file, subfolder=''):
    """Save uploaded file and return the filename"""
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        # Add timestamp to filename to avoid conflicts
        timestamp = datetime.now().strftime('%Y%m%d%H%M%S')
        name, ext = os.path.splitext(filename)
        filename = f"{name}_{timestamp}{ext}"

        upload_path = current_app.config['UPLOAD_FOLDER']
        if subfolder:
            upload_path = os.path.join(upload_path, subfolder)
            os.makedirs(upload_path, exist_ok=True)

        filepath = os.path.join(upload_path, filename)
        file.save(filepath)
        return filename
    return None
```

Figure 5.44 File Upload Security

5.5 Command-Line Interface and Database Management

The DialSmart system provides CLI commands for database initialization and management:

```python
@app.cli.command()
def init_db():
    """Initialize the database"""
    print("Creating database tables...")
    db.create_all()
    print("Database tables created successfully!")

@app.cli.command()
def create_admin():
    """Create an admin user"""
    from app.models import User

    email = input("Enter admin email: ")
    password = input("Enter admin password: ")
    full_name = input("Enter admin full name: ")

    # Check if user already exists
    existing_user = User.query.filter_by(email=email).first()
    if existing_user:
        print(f"User with email {email} already exists!")
        return

    # Create admin user
    admin = User(
        email=email,
        full_name=full_name,
        is_admin=True,
        user_category='Admin'
    )
    admin.set_password(password)

    db.session.add(admin)
    db.session.commit()

    print(f"Admin user '{email}' created successfully!")
```

Figure 5.45 Command-Line Interface and Database Management

These CLI commands facilitate system administration tasks such as database initialization, admin user creation, and data seeding.

5.6 RESTful API Implementation

The DialSmart system exposes RESTful API endpoints for client-server communication, returning JSON-formatted responses for dynamic content updates.

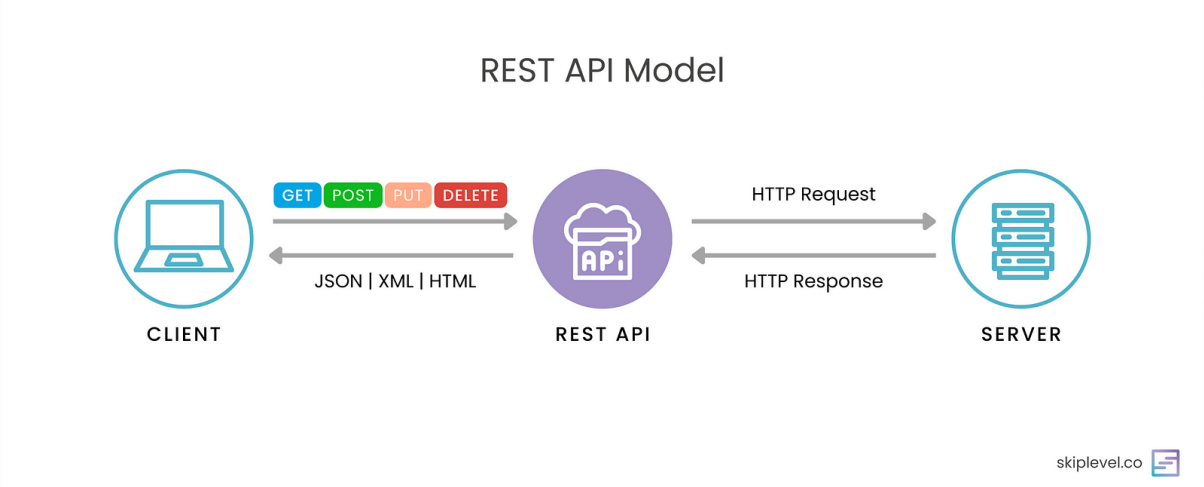

Figure 5.46 RESTful API

5.6.1 API Response Structure

All API endpoints follow a consistent response structure:

```
{
    "success": true,
    "data": {...},
    "message": "Optional success message"
}
```

Figure 5.47 Response Structure

Error responses include appropriate HTTP status codes and error messages:

```
{
    "success": false,
    "error": "Error description",
    "code": 400
}
```

Figure 5.48 Error responses

5.6.2 Chatbot API Endpoint

The chatbot API processes natural language queries and returns intelligent responses:

```python
@bp.route('/chat', methods=['POST'])
@login_required
def chat():
    """Process chatbot message"""
    data = request.get_json()
    message = data.get('message', '')
    session_id = data.get('session_id') or str(uuid.uuid4())

    if not message:
        return jsonify({'error': 'Message is required'}), 400

    # Process with chatbot engine
    chatbot = ChatbotEngine()
    response = chatbot.process_message(current_user.id, message, session_id)

    return jsonify({
        'success': True,
        'response': response['response'],
        'type': response.get('type', 'text'),
        'metadata': response.get('metadata', {}),
        'quick_replies': response.get('quick_replies', []),
        'session_id': session_id
    })
```

Figure 5.49 Chatbot API Endpoint

5.6.3 Phone Search and Filter API

Advanced search and filtering capabilities are exposed through API endpoints:

```python
@bp.route('/phones/filter', methods=['POST'])
def filter_phones():
    """Filter phones based on criteria"""
    data = request.get_json()

    brand_ids = data.get('brand_ids', [])
    min_price = data.get('min_price')
    max_price = data.get('max_price')
    min_ram = data.get('min_ram')
    has_5g = data.get('has_5g')
    min_battery = data.get('min_battery')
    page = data.get('page', 1)
    per_page = data.get('per_page', 12)

    # Build query
    query = Phone.query.filter_by(is_active=True)

    if brand_ids:
        query = query.filter(Phone.brand_id.in_(brand_ids))

    if min_price:
        query = query.filter(Phone.price >= min_price)

    if max_price:
        query = query.filter(Phone.price <= max_price)

    # Join with specifications for advanced filters
    if min_ram or has_5g or min_battery:
        query = query.join(PhoneSpecification)

        if min_ram:
            query = query.filter(PhoneSpecification.ram_options.ilike(f'%{min_ram}GB%'))

        if has_5g:
            query = query.filter(PhoneSpecification.has_5g == True)

        if min_battery:
            query = query.filter(PhoneSpecification.battery_capacity >= min_battery)

    # Paginate
    phones = query.paginate(page=page, per_page=per_page, error_out=False)

    phone_list = [{
        'id': phone.id,
        'model_name': phone.model_name,
        'brand': phone.brand.name if phone.brand else 'Unknown',
        'price': phone.price,
        'main_image': phone.main_image
    } for phone in phones.items]

    return jsonify({
        'success': True,
        'phones': phone_list,
        'total': phones.total,
        'pages': phones.pages,
        'current_page': phones.page
    })
```

Figure 5.50 Phone Search and Filter API

5.7 Dependencies and Technology Stack

The DialSmart system is built on a modern Python technology stack with the following key dependencies:

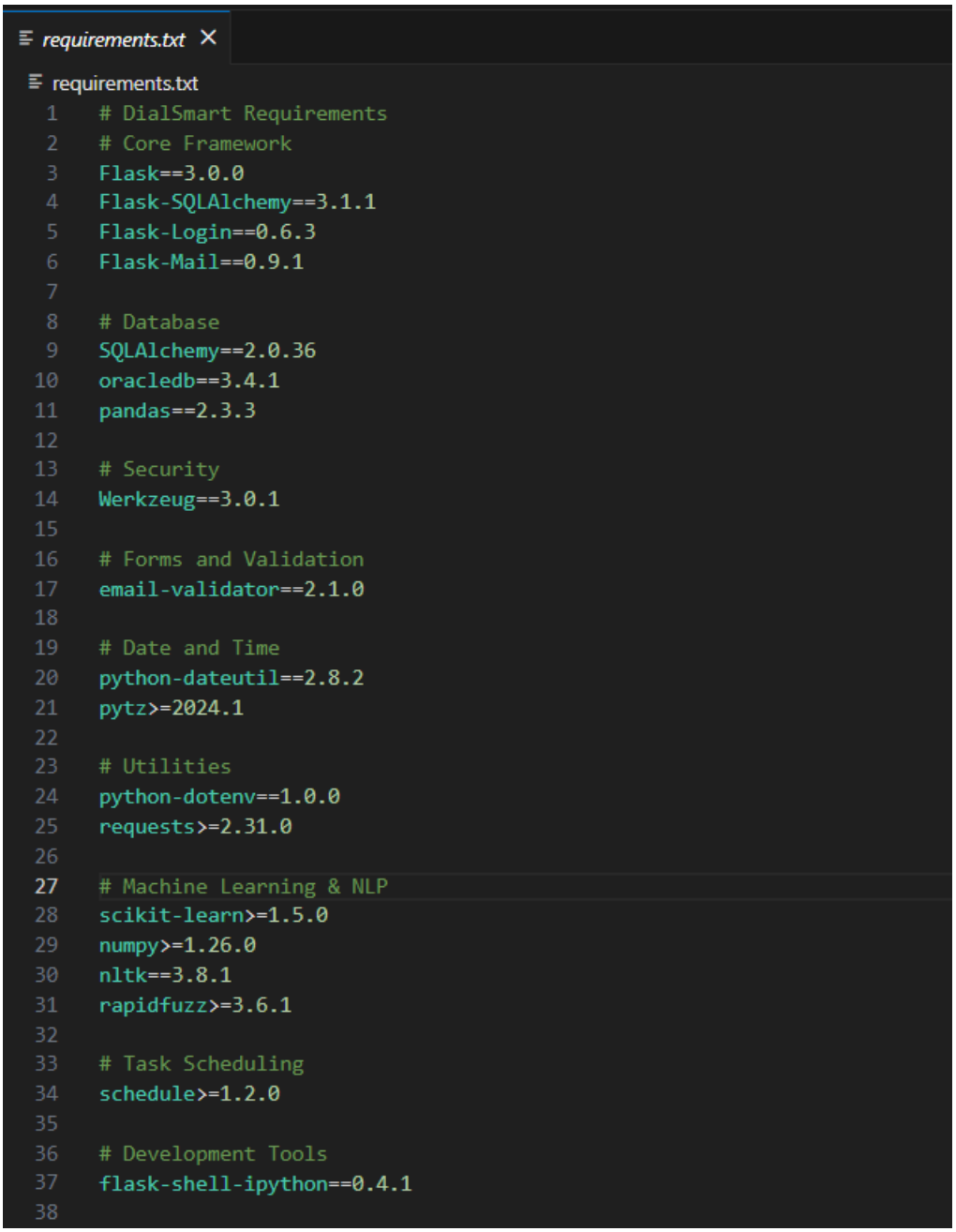

Figure 5.51 Dependencies and Technology Stack

Table 5.1 Dependencies and Technology Stack

Category	Technology	Version	Purpose
Core Framework	Flask	3.0.0	Lightweight WSGI web application framework
	Flask-SQLAlchemy	3.1.1	SQLAlchemy integration for Flask
	Flask-Login	0.6.3	User session management for Flask
	Flask-Mail	0.9.1	Email sending capabilities for Flask
Database	SQLAlchemy	2.0.36	SQL toolkit and Object-Relational Mapping
	oracledb	3.4.1	Python extension module for Oracle Database
	pandas	2.3.3	Data manipulation and analysis library
Security	Werkzeug	3.0.1	Password hashing and security utilities
Forms & Validation	email-validator	2.1.0	Email address validation
Date & Time	python-dateutil	2.8.2	Date and datetime utilities
	pytz	2024.1	Timezone handling
HTTP Requests	requests	2.31.0	HTTP library for API calls
Machine Learning	scikit-learn	1.5.0	Machine learning algorithms
	numpy	1.26.0	Numerical computing library
Natural Language Processing	nltk	3.8.1	Natural Language Toolkit
	rapidfuzz	3.6.1	Fast string matching and fuzzy searching
Task Scheduling	schedule	1.2.0	Job scheduling library
Development Tools	flask-shell-ipython	0.4.1	Enhanced Flask shell with IPython
Frontend	Bootstrap 5, HTML5, CSS3, JavaScript	-	User interface and client-side functionality
API Architecture	RESTful JSON API	-	Standard HTTP methods for communication

5.8 Server Configuration and Deployment Settings

The DialSmart application server is configured for both development and production environments:

5.8.1 Development Server Configuration

```
if __name__ == '__main__':
    # Run the application
    app.run(
        host='0.0.0.0',       # Accessible from all network interfaces
        port=5000,            # Default Flask port
        debug=True            # Enable debug mode for development
    )
```

Figure 5.52 Development Server Configuration

Server Settings:
Host: 0.0.0.0 (Binds to all available network interfaces)
Port: 5000 (Standard Flask development port)
Debug Mode: Enabled in development for detailed error messages and auto-reload
Protocol: HTTP (HTTPS recommended for production)

5.8.2 Database Connection Settings

```
# Database settings
BASE_DIR = os.path.abspath(os.path.dirname(__file__))
SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
    'oracle+cx_oracle://username:password@localhost:1521/dialsmart'
SQLALCHEMY_TRACK_MODIFICATIONS = False

# Oracle-specific configuration
SQLALCHEMY_ENGINE_OPTIONS = {
    'pool_size': 10,
    'pool_recycle': 3600,
    'pool_pre_ping': True
}
```

Table 5.2 Connection Settings

Parameter	Value
Database System	Oracle Database
Access Method	SQL*Plus
Connection String	oracle+cx_oracle://username:password@host:port/service_name
Port	1521

Table 5.3 Connection Pool Management

Parameter	Value
Pool Manager	SQLAlchemy
Pool Size	10 connections
Pool Recycle Time	3600 seconds (1 hour)
Pre-ping Validation	Enabled

Table 5.4 Performance & Maintenance

Parameter	Value
Query Optimization	Indexes on frequently queried columns
Backup Strategy	Regular Oracle RMAN backups (recommended for production)

5.9 Chapter Summary and Evaluation

The implementation of the DialSmart AI-Powered Smartphone Recommendation System demonstrates a comprehensive application of modern web development principles and artificial intelligence technologies. The system successfully integrates multiple architectural patterns including Client-Server architecture, Model-View-Controller (MVC), and the Application Factory pattern to create a scalable, maintainable, and secure web application.

The core recommendation engine utilizes intelligent matching algorithms that consider multiple factors including budget constraints, technical specifications, and user preferences to generate personalized smartphone recommendations. The integration of a conversational AI chatbot enhances user experience by providing natural language interaction capabilities, making the system accessible to users with varying technical knowledge levels.

Security implementation across multiple layers, including password hashing, session management, input validation, and secure file handling, ensures the protection of user data and maintains system integrity. The RESTful API architecture enables seamless communication between client and server components, supporting dynamic content updates and responsive user interfaces.

The modular blueprint-based architecture, combined with environment-specific configurations, facilitates deployment flexibility and supports future system enhancements. The comprehensive database schema efficiently stores and manages user profiles, smartphone specifications, recommendation history, and chatbot interactions, providing a robust foundation for data-driven decision making.

Overall, the DialSmart system demonstrates successful implementation of a complex web application that combines artificial intelligence, database management, user authentication, and responsive design to deliver an intelligent smartphone recommendation platform tailored for the Malaysian consumer market.

IMPORTANT NOTE TO STUDENTS: Include details about:

- **Implementation**

  A detailed description of how you actually carried out the implementation (e.g., coding, etc.) of your system/prototype.

  - Include code snippets and descriptions to show how the requirements of the application/prototype have been met –
    - For Smart Campus projects, code snippets of the MQTT protocol for both client side and server side has to be included with explanation.
  - Include descriptions of important settings - Setting for server, network protocol, IP, database, security
  - Note: this should **not** be a chronological account of the work you carried out.

- **Testing**

  All test cases that have been carried out must be provided - To tabulate the test cases in tables

Note: Students may also opt to split Implementation and Testing into 2 separate chapters.

Chapter 6

Testing

6 Testing

Chapter 6 provides an in-depth exploration of the testing phase in the development lifecycle of the DialSmart AI-powered Smartphone Recommendation System. It comprehensively covers various testing strategies, including Black-Box Testing, White-Box Testing, Top-Down Testing, Component Testing, and UI Testing. Subsequently, the focus shifts to the formulation of a comprehensive Test Plan (6.2), covering key aspects such as Testing Phases, Testing Recording Procedures, Testing Items, Hardware and Software Requirements, and Constraints. The heart of the testing process lies in Test Cases (6.3), ranging from Unit Testing to User Acceptance Testing (UAT), each serving a vital role in ensuring the system's reliability and alignment with user expectations.

6.1 Testing Strategies

In the context of the DialSmart AI-powered Smartphone Recommendation System, the testing strategies are thoughtfully designed to ensure the software meets high standards in quality, functionality, and reliability. This comprehensive approach incorporates a mix of testing methods, including black-box, white-box, top-down, component, and UI testing. Each of these methods serves a unique role in examining different aspects of the system, ensuring a complete evaluation that covers user experience, functionality, and internal processes. The ultimate goal is not just to meet specified requirements but to provide a seamless, user-friendly, and reliable experience for all users.

6.1.1 Black-Box Testing

In this DialSmart AI-powered Smartphone Recommendation System, black-box testing assumes a pivotal role as a crucial methodology. This approach entails a thorough examination of the software's functionalities without delving into its internal code, which mean without the internal coding or logic knowledge. Therefore, to enhance the authenticity of this testing process, the developer conducted black-box testing with external users who have no prior knowledge of the system's internal workings. Testers are required to only focus on inputs, outputs, and system behavior to validate features such as AI-powered recommendations, chatbot interactions, phone comparison functionality, user authentication, and admin panel operations. The ultimate goal is to uncover potential issues and ensure that the software aligns seamlessly with user expectations and functional requirements.

6.1.2 White-Box Testing

In contrast to black-box testing, white-box testing involves a detailed exploration and requires an understanding of the system's internal code, logic, and data structures. This comprehensive analysis is conducted to verify the integrity and reliability of the business logic governing AI recommendation algorithms, chatbot NLP processing, user preference matching, phone comparison calculations, and database operations. While white-box testing typically requires individuals with a deep understanding of the code's internal structure, the developer conducted this testing internally by examining the Python Flask application code, database queries, AI recommendation engine logic, and API endpoint implementations. By comprehensively assessing the system at the code level, this methodology offers valuable insights into the code's quality, ensuring it is free from errors and logical flaws that could impact the overall performance of the system.

6.1.3 Top-Down Testing

Top-down testing emerges as a strategic approach to validate the system from its highest level down to individual components. Utilizing top-down testing in this system, the developer has strategically validated the software from its highest level down to individual components. This approach ensures the seamless operation of overarching functionalities, such as the complete user recommendation flow, end-to-end chatbot conversations, full admin panel operations, and integrated phone comparison processes, before delving into the specifics of individual components. Top-down testing plays a pivotal role in identifying integration issues and guaranteeing a cohesive user experience throughout the entire system. This approach aids in assessing the overall system architecture, highlighting potential bottlenecks or communication issues between different modules such as the AI engine, database layer, user interface, and API services.

6.1.4 Component Testing

Component testing, a crucial element of the testing framework, focuses on evaluating individual modules or components of the system in isolation. The developer has employed Component Testing in this system to verify the functionality, performance, and reliability of individual system modules, ensuring they meet the specified requirements. In the context of the DialSmart AI-powered Smartphone Recommendation System, this approach ensures the seamless functionality of specific

components, such as user authentication processes, AI recommendation engine calculations, chatbot intent detection and response generation, phone comparison algorithms, admin CRUD operations, and API endpoint handlers. By isolating components, this strategy facilitates the early detection of defects, contributing to the overall reliability and effectiveness of the system. Component testing is instrumental in assessing the functionality, performance, and reliability of individual system modules, ensuring they meet the specified requirements.

6.1.5 UI Testing

In the context of the DialSmart AI-powered Smartphone Recommendation System, User Interface (UI) testing plays a pivotal role in ensuring a positive and user-friendly experience. Testers rigorously assess the graphical user interface, navigation, and overall aesthetics of the system. UI testing is instrumental in identifying issues related to layout, responsiveness (both on mobile and computer), and overall design coherence, contributing to an optimal user experience. Therefore, the developer has implemented this methodology to validate the visual and interactive elements of the system. Furthermore, the implementation of this methodology involves validating elements such as responsive design across different devices, form input validations, error message displays, navigation flow, button placements and interactions, chatbot interface usability, recommendation result presentations, and comparison table layouts to guarantee a seamless and visually appealing interaction for users.

6.2 Test Plan

The Test Plan for the DialSmart AI-powered Smartphone Recommendation System encompasses a comprehensive strategy to ensure the software's reliability, functionality, and performance. The testing process will cover various modules, including User Authentication, AI Recommendation Engine, Chatbot System, Phone Management, Comparison Feature, User Preferences, Admin Panel, and Contact and Support. Each module will be subjected to a series of tests to validate its individual functions, followed by integrated tests to ensure seamless interactions across the entire system. This strategic and all-encompassing approach not only seeks to identify and rectify potential issues at the micro-level but also aims to guarantee a seamless and robust user experience at the macro level, aligning with the overarching goal of delivering a high-quality and user-friendly application.

6.2.1 Testing Phrase

The testing phases in the development of the DialSmart AI-powered Smartphone Recommendation System are integral to guaranteeing the software's quality and reliability. Unit testing constitutes a meticulous examination of individual components, focusing on their functionality and correctness. Module testing extends this scrutiny to each module independently, ensuring that they perform as expected. System testing, a broader evaluation, assesses the seamless integration and collaboration of all system modules including the AI engine, database operations, user interface, and API services. The User Acceptance Testing (UAT) phase involves end-users actively participating in verifying that the system aligns with their specific requirements and expectations, including testing the recommendation accuracy, chatbot responsiveness, and overall usability of the platform. For a detailed overview of each test case scenario and test data, covering unit, module, and system tests, please refer to section 6.3 Test Cases for comprehensive guidance.

6.2.2 Testing Recording Procedures

The provided template serves as a structured format for documenting essential test cases required to ensure the optimal performance of the system. It serves the purpose of identifying what actions need to be executed, when they are scheduled, and who is responsible for carrying out the test cases. Precondition statements are included to outline the conditions that ideally should be met before initiating each test case. Furthermore, the template details test cases related to actions and functions that end-users will interact with, including user registration and login, AI recommendation requests, chatbot interactions, phone browsing and filtering, comparison operations, profile management, and admin panel functions. Post-condition statements articulate the anticipated outcomes and expectations after the completion of the test cases. This structured approach enhances the systematic documentation of test cases, contributing to an organized and effective testing process for the system.

Table 6.1 Template of Test Case Recorded Table

Test Case #:	Test Case Name:
System:	Subsystem:
Design By:	Design Date:
Executed By:	Execution Date:
Short Description:	

Pre-conditions:

Step	**Action/Functions**	**Test Data**	**Expected System Response**	**Actual Response**	**Pass(P) / Fail(F)**	**Comments**
1						
2						
3						
4						

Post-conditions:

6.2.3 Testing Items

The Testing Items section functions as a comprehensive checklist, cataloguing specific functions within critical modules such as User Authentication, AI Recommendation Engine, Chatbot System, Phone Management, User Preferences, Comparison Feature, Admin Panel, and Contact and Support. This detailed inventory serves as a guide for the meticulous testing of each function, aiming to identify and rectify potential issues. The goal is to ensure the seamless operation and user satisfaction of the DialSmart AI-powered Smartphone Recommendation System.

Table 6.2: Test Item checklist of User Authentication Module

Module	**Test Item**
User Authentication Module	RegisterUser()
	AuthenticateUser()
	ValidateEmail()
	ValidatePassword()
	CheckUserExists()
	CreateUserSession()
	LogoutUser()
	ResetPassword()
	SendPasswordResetEmail()
	ValidateResetToken()
	UpdateUserProfile()
	ChangePassword()

Table 6.3: Test Item checklist of AI Recommendation Module

Module	**Test Item**
AI Recommendation Module	GetRecommendations()
	CalculateMatchScore()
	FilterByBudget()
	FilterByPreferences()
	GenerateReasoning()
	GetBudgetRecommendations()
	GetPhonesByUsage()
	GetSimilarPhones()
	SaveRecommendationHistory()

Table 6.4: Test Item checklist of Chatbot Module

Module	**Test Item**
Chatbot Module	ProcessMessage()
	DetectIntent()
	ExtractBudget()
	ExtractCriteria()
	DetectUsageType()
	ExtractBrand()
	GenerateResponse()
	SaveChatHistory()
	GetChatHistory()
	HandleGreeting()

Table 6.5: Test Item checklist of Phone Management Module

Module	**Test Item**
Phone Management Module	AddPhone()
	UpdatePhone()
	DeletePhone()
	GetPhoneDetails()
	SearchPhones()
	FilterPhones()
	UploadPhoneImage()
	UpdateSpecifications()
	TogglePhoneStatus()
	GetPhonesByBrand()

Table 6.6: Test Item checklist of User Preferences Module

Module	**Test Item**
User Preferences Module	SetBudgetRange()
	SetUsagePreferences()
	SetFeaturePreferences()
	SetBrandPreferences()
	UpdatePreferences()
	GetUserPreferences()
	ValidatePreferences()
	ResetPreferences()
	SavePreferenceHistory()

Table 6.7: Test Item checklist of Comparison Module

Module	**Test Item**
Comparison Module	ComparePhones()
	BuildComparisonTable()
	DetermineWinner()
	SaveComparison()
	GetComparisonHistory()
	DeleteComparison()
	CalculateScores()
	DisplayDifferences()

Table 6.8: Test Item checklist of Admin Panel Module

Module	**Test Item**
Admin Panel Module	ViewDashboard()
	ManageUsers()
	ManagePhones()
	ManageBrands()
	ViewAnalytics()
	ToggleUserStatus()
	SendSuspensionEmail()
	SendReactivationEmail()
	ViewSystemLogs()
	ExportReports()
	UpdateSystemSettings()

Table 6.9: Test Item checklist of Contact and Support Module

Module	Test Item
Contact and Support Module	SubmitContactForm()
	ValidateContactForm()
	SaveContactRequest()
	SendAutoAcknowledgment()
	NotifyAdmin()
	ViewContactRequests()
	ComposeReply()
	SendReplyEmail()
	SaveReplyHistory()
	UpdateRequestStatus()

6.3 Test Cases

This section presents detailed test cases for the DialSmart AI-powered Smartphone Recommendation System. The test cases are organized by module and testing phase (Unit, Module, and System testing), providing comprehensive coverage of all system functionalities.

6.3.1 User Authentication Module - Unit Testing

Table 6.10 User Registration with Valid Data

Test Case #: TC-AUTH-001 Test Case Name: User Registration with Valid Data

Test Case #: **TC-AUTH-001**	Test Case Name: **User Registration with Valid Data**
System: DialSmart	Subsystem: User Authentication
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that a new user can successfully register with valid credentials and information.	

Pre-conditions: 1. User is on the registration page (/auth/register) 2. User is not already registered in the system 3. All form fields are accessible

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to registration page	URL: /auth/register	Registration form is displayed with all required fields	Registration form is displayed with all required fields	P	
2	Enter full name	"Alison Ooi"	Full name field accepts input	Full name field accepts input	P	
3	Enter email address	"ooiyz-am22@student.tarc.edu.my"	Email field accepts valid email format	Email field accepts valid email format	P	
4	Enter password	"Abcd1234#"	Password field accepts input and masks characters	Password field accepts input and masks characters	P	

Table 6.10 User Authentication Module – Unit Testing(continued)

5	Confirm password	" Abcd1234#"	Confirm password field accepts input	Confirm password field accepts input	P	
6	Select user category	"Working Professional"	Dropdown shows user category options	Dropdown shows user category options	P	
7	Select age range	"26-35"	Dropdown shows age range options	Dropdown shows age range options	P	
8	Click Register button	Submit form	System validates input, creates user account, shows success message, redirects to login page	System validates input, creates user account, shows success message, redirects to login page	P	

Post-conditions:
1. User session is created
2. User is redirected to dashboard
3. User can access protected routes
4. Last active timestamp is updated

Table 6.11 User Login with Valid Credentials

Test Case #: TC-AUTH-002 **Test Case Name**: User Login with Valid Credentials

Test Case #: **TC-AUTH-002**	Test Case Name: **User Login with Valid Credentials**
System: DialSmart	Subsystem: User Authentication
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that a registered user can successfully login with correct email and password.	

Pre-conditions:
1. User has already registered in the system
2. User is on the login page (/auth/login)
3. User account is active

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to login page	URL: /auth/login	Login form is displayed with email and password fields	Login form is displayed with email and password fields	P	
2	Enter registered email	"ooiyz-am22@student.tarc.edu.my"	Email field accepts input	Email field accepts input	P	
3	Enter correct password	"Abcd1234#"	Password field accepts input and masks characters	Password field accepts input and masks characters	P	
4	Click Login button	Submit form	System authenticates user, creates session, redirects to dashboard	System authenticates user, creates session, redirects to dashboard	P	
5	Verify dashboard access	Check URL: /dashboard	User dashboard is displayed with personalized content	User dashboard is displayed with personalized content	P	

Post-conditions:
1. User session is created
2. User is redirected to dashboard
3. User can access protected routes
4. Last active timestamp is updated

Table 6.12 User Login with Invalid Credentials

Test Case #: TC-AUTH-003 Test Case Name: User Login with Invalid Credentials

Test Case #: **TC-AUTH-003**	Test Case Name: **User Login with Invalid Credentials**
System: DialSmart	Subsystem: User Authentication
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that system rejects login attempts with incorrect credentials.	

Pre-conditions:
1. User is on the login page
2. User may or may not be registered

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to login page	URL: /auth/login	Login form is displayed	Login form is displayed	P	
2	Enter email	"test@gmail.com"	Email field accepts input	Email field accepts input	P	
3	Enter wrong password	"WrongPassword123"	Password field accepts input	Password field accepts input	P	
4	Click Login button	Submit form	System shows error message "Invalid email or password", user remains on login page	System shows error message "Invalid email or password", user remains on login page	P	
5	Verify no session created	Check session	No user session is created	No user session is created	P	

Post-conditions:
1. User remains on login page
2. Error message is displayed
3. No session is created
4. User can retry login

Table 6.13 Forget Password with Email Reset Link

Test Case #: TC-AUTH-004 Test Case Name: Forgot Password with Email Reset Link

Test Case #: **TC-AUTH-004**	Test Case Name: **Forgot Password with Email Reset Link**
System: DialSmart	Subsystem: User Authentication
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that user receives password reset link via email when requesting password reset.	

Pre-conditions:
1. User has registered account in system
2. User is on forgot password page (/auth/forgot-password)
3. Email service is configured and operational

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to login page	URL: /auth/login	Login page is displayed	Login page is displayed	P	
2	Click "Forgot Password" link	Click forgot password link	Redirected to forgot password page (/auth/forgot-password)	Redirected to forgot password page (/auth/forgot-password)	P	
3	Verify page elements	Check page content	Form with email input field and submit button displayed	Form with email input field and submit button displayed	P	
4	Enter registered email	"alisonooi999@gmail.com"	Email field accepts input	Email field accepts input	P	
5	Click Submit button	Submit form	Success message displayed: "Password reset instructions have been sent to your email."	Success message displayed: "Password reset instructions have been sent to your email."	P	
6	Verify email sent	Check email system logs	Password reset email is sent to user's email address	Password reset email is sent to user's email address	P	
7	Check email content	Open user's email inbox	Email received with password reset link and instructions	Email received with password reset link and instructions	P	

Table 6.13 Forget Password with Email Reset Link(continued)

8	Verify reset link format	Check link in email	Reset link contains token: /auth/reset-password?token=<unique_token>	Reset link contains token: /auth/reset-password?token=<unique_token>	P	
9	Verify token stored	Check database	Password reset token is stored with expiration timestamp (24 hours)	Password reset token is stored with expiration timestamp (24 hours)	P	
10	Test with unregistered email	Enter "nonexistent@test.com"	Warning message: "Email not found." No email sent	Warning message: "Email not found." No email sent	P	
11	Click reset link	Click link in email	Redirected to password reset form with token validation	Redirected to password reset form with token validation	P	
12	Enter new password	New password: "NewSecure123!"	Password reset form accepts new password	Password reset form accepts new password	P	
13	Confirm new password	Confirm: "NewSecure123!"	Confirmation matches new password	Confirmation matches new password	P	
14	Submit password reset	Submit form	Password updated, token invalidated, success message shown	Password updated, token invalidated, success message shown	P	
15	Login with new password	Use new credentials	User successfully logs in with new password	User successfully logs in with new password	P	

Post-conditions:
1. Password reset email sent to user
2. Reset link with unique token generated
3. Token stored in database with expiration
4. User can reset password using link
5. Old password no longer works
6. Token is invalidated after use
7. User can login with new password

Table 6.14 Admin Suspend User with Email Notification

Test Case #: TC-AUTH-005 Test Case Name: Admin Suspend User with Email Notification

Test Case #: **TC-AUTH-005**	Test Case Name: **Admin Suspend User with Email Notification**
System: DialSmart	Subsystem: Admin User Management
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that user receives email notification when admin suspends their account.	

Pre-conditions:
1. Admin user is logged in
2. Target user account is active
3. Admin is on users management page (/admin/users)
4. Email notification service is operational

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to admin users page	URL: /admin/users	Users list is displayed with all registered users	Users list is displayed with all registered users	P	
2	Locate target user	Search for "Alison"	User appears in the list with active status	User appears in the list with active status	P	
3	View user details	Click on user name	User details page shows account information and activity	User details page shows account information and activity	P	
4	Verify current status	Check user status	User status shows "Active" with green indicator	User status shows "Active" with green indicator	P	
5	Click suspend button	Click "Toggle Status" or "Suspend" button	Confirmation dialog appears	Confirmation dialog appears	P	
6	Confirm suspension	Confirm action	User status changed to "Suspended", success message displayed	User status changed to "Suspended", success message displayed	P	
7	Verify database update	Check user record	User.is_active field set to False in database	User.is_active field set to False in database	P	

Table 6.14 Admin Suspend User with Email Notification (continued)

8	Verify email sent	Check email system logs	Suspension notification email sent to user's email	Suspension notification email sent to user's email	P	
9	Check email content	Open user's email inbox	Email received with suspension notification and reason	Email received with suspension notification and reason	P	
10	Verify email details	Review email content	Email contains: suspension date, reason (if provided), contact support info	Email contains: suspension date, reason (if provided), contact support info	P	
11	Test user login attempt	User tries to login	Login blocked with message: "Your account has been suspended. Please contact support."	Login blocked with message: "Your account has been suspended. Please contact support."	P	
12	Verify access denied	Check protected routes	Suspended user cannot access dashboard or user features	Suspended user cannot access dashboard or user features	P	
13	Admin reactivates user	Admin clicks "Toggle Status" again	User status changed to "Active"	User status changed to "Active"	P	
14	Verify reactivation email	Check email system	Reactivation notification email sent to user	Reactivation notification email sent to user	P	
15	Test user can login	User logs in with credentials	Login successful, access restored to all features	Login successful, access restored to all features	P	

Post-conditions:
1. User account is suspended (is_active = False)
2. Suspension email notification sent to user
3. User cannot login while suspended
4. Admin can reactivate account
5. Reactivation email sent when account restored
6. All actions logged in system
7. User regains full access after reactivation

Table 6.15 Contact Us with Admin Reply via Email

Test Case #: TC-AUTH-006 Test Case Name: Contact Us with Admin Reply via Email

Test Case #: **TC-AUTH-006**	Test Case Name: **Contact Us with Admin Reply via Email**
System: DialSmart	Subsystem: Contact and Support
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that users can submit feedback/requests via contact form and receive admin reply via email.	

Pre-conditions:
1. User is on contact page (/contact)
2. Contact form is accessible to all users (logged in or guest)
3. Email service and admin notification system operational

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to contact page	URL: /contact	Contact form displayed with fields: Name, Email, Subject, Message	Contact form displayed with fields: Name, Email, Subject, Message	P	
2	Enter full name	"Alison"	Name field accepts input	Name field accepts input	P	
3	Enter email address	"alisonooi999@gmail.com"	Email field accepts valid email	Email field accepts valid email	P	
4	Enter subject	"Inquiry about phone recommendations"	Subject field accepts input	Subject field accepts input	P	
5	Enter message	"I need help finding a phone for photography. Can you recommend some options under RM3000?"	Message textarea accepts multi-line input	Message textarea accepts multi-line input	P	
6	Click Submit button	Submit contact form	Success message: "Thank you for contacting us. We will get back to you soon."	Success message: "Thank you for contacting us. We will get back to you soon."	P	
7	Verify form submission saved	Check database	Contact request stored in database with timestamp	Contact request stored in database with timestamp	P	

Table 6.15 Contact Us with Admin Reply via Email (continued)

8	Verify admin notification	Check admin email/dashboard	Admin receives notification of new contact request	Admin receives notification of new contact request	P	
9	Verify auto-reply sent	Check user's email	Auto-acknowledgment email sent: "We received your message and will respond within 24 hours"	Auto-acknowledgment email sent: "We received your message and will respond within 24 hours"	P	
10	Admin reviews request	Admin navigates to contact requests	Contact request visible in admin panel with all details	Contact request visible in admin panel with all details	P	
11	Admin composes reply	Admin writes response: "Here are some great camera phones..."	Reply form accepts admin's message	Reply form accepts admin's message	P	
12	Admin sends reply	Click "Send Reply" button	Reply sent confirmation message displayed	Reply sent confirmation message displayed	P	
13	Verify reply email sent	Check email system	Reply email sent from admin email to user	Reply email sent from admin email to user	P	
14	Check reply email content	Open user's email	Email contains: admin's reply, original message quoted, support contact info	Email contains: admin's reply, original message quoted, support contact info	P	
15	Verify reply saved	Check database	Admin reply stored and linked to original contact request	Admin reply stored and linked to original contact request	P	
16	Test empty form submission	Submit with empty fields	Validation errors displayed for required fields	Validation errors displayed for required fields	P	
17	Test invalid email format	Enter "invalid-email"	Email validation error: "Please enter a valid email address"	Email validation error: "Please enter a valid email address"	P	
18	Test guest user submission	Submit as non-logged-in user	Form works for both guests and logged-in users	Form works for both guests and logged-in users	P	

Post-conditions:
1. Contact request saved in database
2. Auto-acknowledgment email sent to user
3. Admin notified of new request
4. Admin can view and reply to requests
5. Reply email sent to user with admin's response
6. All communication history maintained
7. System tracks request status (pending/replied)
8. Users can submit multiple inquiries

6.3.2 AI Recommendation Module - Unit Testing

Table 6.16 Get Recommendations Based on User Preferences

Test Case #: TC-AI-001 Test Case Name: Get Recommendations Based on User Preferences

Test Case #: **TC-AI-001**	Test Case Name: **Get Recommendations Based on User Preferences**
System: DialSmart	Subsystem: AI Recommendation Engine
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that AI engine generates personalized recommendations based on user preferences.	

Pre-conditions:
1. User is logged in
2. User has set preferences (budget, usage type, features)
3. Active phones exist in database

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to recommendations page	URL: /recommendations	Recommendations page loads	Recommendations page loads	P	
2	Set budget range	Min: RM1000, Max: RM3000	Budget preference is saved	Budget preference is saved	P	
3	Select usage type	"Gaming"	Usage preference is saved	Usage preference is saved	P	
4	Select important features and brands	["5G", "High RAM", "Large Battery"], [“Samsung”]	Features preferences and brands are saved	Features preferences and brands are saved	P	
5	Click Get Recommendations	Execute AI engine	System displays 3-5 phones with match scores (50-100%), reasoning, and specifications	System displays 3-5 phones with match scores (50-100%), reasoning, and specifications	P	
6	Verify match scores	Check scores	All recommended phones have match score ≥ 50%	All recommended phones have match score ≥ 50%	P	

Table 6.16 Get Recommendations Based on User Preferences (continued)

7	Verify price range	Check prices	All phones are within RM1000-3000 range	All phones are within RM1000-3000 range	P	
8	Verify recommendations saved	Check database	Recommendations are saved in database with timestamp	Recommendations are saved in database with timestamp	P	

Post-conditions:
1. User sees personalized recommendations
2. Recommendations are saved in history
3. Match scores and reasoning are displayed
4. User can view phone details

Table 6.17 Calculate Match Score for Phone

Test Case #: TC-AI-002 Test Case Name: Calculate Match Score for Phone

Test Case #: **TC-AI-002**	Test Case Name: **Calculate Match Score for Phone**
System: DialSmart	Subsystem: AI Recommendation Engine
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that match score calculation is accurate based on user criteria.	

Pre-conditions:
1. User preferences are defined
2. Phone with specifications exists in database

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Define user criteria	Budget: RM2000, RAM: 8GB, Battery: 5000mAh, 5G: Yes	Criteria object is created	Criteria object is created	P	
2	Select test phone	Samsung Galaxy A54 (Price: RM1899, RAM: 8GB, Battery: 5000mAh, 5G: Yes)	Phone data retrieved	Phone data retrieved	P	
3	Call calculate_match_score()	User criteria + Phone specs	Function returns match score	Function returns match score	P	
4	Verify score calculation	Check score value	Score is between 0-100, high score (≥80%) due to good match	Score is between 0-100, high score (≥80%) due to good match	P	
5	Verify scoring factors	Check calculation breakdown	Price match, RAM match, battery match, 5G presence all contribute positively	Price match, RAM match, battery match, 5G presence all contribute positively	P	

Post-conditions:
1. Match score is calculated correctly
2. Score reflects alignment with user preferences
3. Scoring logic is consistent and fair

6.3.3 Chatbot Module - Unit Testing

Table 6.18 Chatbot Intent Detection

Test Case #: TC-CHAT-001 Test Case Name: Chatbot Intent Detection

Test Case #: **TC-CHAT-001**	Test Case Name: **Chatbot Intent Detection**
System: DialSmart	Subsystem: Chatbot Engine
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that chatbot correctly identifies user intent from natural language input.	

Pre-conditions:
1. User is logged in
2. Chatbot interface is accessible
3. Chat session is active

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Open chatbot interface	Click chatbot icon	Chatbot window opens with greeting message	Chatbot window opens with greeting message	P	
2	Send greeting message	"Hello"	Intent: 'greeting', Response: Welcome message with options	Intent: 'greeting', Response: Welcome message with options	P	
3	Send budget query	"Recommend phones under RM2000"	Intent: 'budget_query', System extracts budget (500-2000), shows phone recommendations	Intent: 'budget_query', System extracts budget (500-2000), shows phone recommendations	P	
4	Send recommendation query	"Find me a phone with good camera"	Intent: 'recommendation', System triggers recommendation flow	Intent: 'recommendation', System triggers recommendation flow	P	
5	Send brand query	"Show me Samsung phones"	Intent: 'brand_query', System extracts brand name and displays Samsung phones	Intent: 'brand_query', System extracts brand name and displays Samsung phones	P	

Table 6.18 Chatbot Intent Detection (continued)

6	Send usage query	"Best phone for gaming"	Intent: 'usage_type', System detects 'Gaming' usage and recommends suitable phones	Intent: 'usage_type', System detects 'Gaming' usage and recommends suitable phones	P	
7	Send comparison query	"Compare iPhone 15 and Samsung S23"	Intent: 'comparison', System suggests using comparison feature	Intent: 'comparison', System suggests using comparison feature	P	

Post-conditions:
1. All intents are detected correctly
2. Appropriate responses are generated
3. Chat history is saved in database
4. User receives relevant information

Table 6.19 Chatbot Budget Extraction

Test Case #: TC-CHAT-002 Test Case Name: Chatbot Budget Extraction

Test Case #: **TC-CHAT-002**	Test Case Name: **Chatbot Budget Extraction**
System: DialSmart	Subsystem: Chatbot Engine
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that chatbot accurately extracts budget information from user messages.	

Pre-conditions:
1. Chatbot engine is initialized
2. User has active chat session

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Test "under X" pattern	"phones under RM2000"	Extracts budget: (500, 2000)	Extracts budget: (500, 2000)	P	
2	Test range pattern	"phone RM1000 and RM3000"	Extracts budget: (1000, 3000)	Extracts budget: (1000, 3000)	P	
3	Test single value	"RM1500 phone"	Extracts budget: (500, 1500) - assumes max	Extracts budget: (500, 1500) - assumes max	P	
4	Test "below X" pattern	"below 2500"	Extracts budget: (500, 2500)	Extracts budget: (500, 2500)	P	
5	Test without RM prefix	"1000 to 2000"	Extracts budget: (1000, 2000)	Extracts budget: (1000, 2000)	P	
6	Test no budget mentioned	"good camera phone"	Returns None - no budget extracted	Returns None - no budget extracted	P	

Post-conditions:
1. Budget values are extracted correctly
2. Various input formats are handled
3. Edge cases return appropriate values
4. Extracted data can be used for filtering

Table 6.20 Chatbot FAQ and Fallback Responses

Test Case #: TC-CHAT-003 Test Case Name: Chatbot FAQ and Fallback Responses

Test Case #: **TC-CHAT-003**	Test Case Name: Chatbot FAQ and Fallback Responses
System: DialSmart	Subsystem: Chatbot Engine
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that chatbot handles frequently asked questions and provides fallback responses for unclear queries.	

Pre-conditions:
1. User is logged in
2. Chatbot interface is accessible
3. Chat session is active

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Ask about system capabilities	"What can you do?"	Chatbot responds with list of capabilities (recommendations, search, comparison, etc.)		P	
2	Ask about phone availability	"iPhone 15?"	Chatbot searches and responds with availability status		P	
3	Ask about shipping/delivery	"Do you deliver phones?"	Fallback response explaining system is for recommendations, not sales		P	
4	Ask unclear question	"asdfgh" or random text	Fallback response: "I didn't understand that. Can you rephrase?" with help suggestions		P	
6	Ask comparison question	"Which is better Samsung or Apple?"	Chatbot suggests using comparison feature		P	
8	Send empty message	"" (empty string)	Validation error or prompt to enter message			
11	Test special characters	"!@#$%^&*()"	Chatbot handles gracefully without errors			

Table 6.20 Chatbot FAQ and Fallback Responses (continued)

Post-conditions: 1. Budget values are extracted correctly 2. Various input formats are handled 3. Edge cases return appropriate values 4. Extracted data can be used for filtering

6.3.4 Phone Management Module - Module Testing

Table 6.21 Add New Phone with Complete Specifications

Test Case #: TC-PHONE-001 Test Case Name: Add New Phone with Complete Specifications

Test Case #: **TC-PHONE-001**	Test Case Name: **Add New Phone with Complete Specifications**
System: DialSmart	Subsystem: Phone Management (Admin)
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that admin can successfully add a new phone with all specifications.	

Pre-conditions:
1. Admin user is logged in
2. Admin is on Add Phone page (/admin/phones/add)
3. At least one brand exists in database

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to Add Phone page	URL: /admin/phones/add	Phone form with all fields is displayed	Phone form with all fields is displayed	P	
2	Enter model name	"Xiaomi 14 Pro"	Model name field accepts input	Model name field accepts input	P	
3	Select brand	"Xiaomi"	Brand dropdown shows and accepts selection	Brand dropdown shows and accepts selection	P	
4	Enter price	3599.00	Price field accepts decimal value	Price field accepts decimal value	P	
5	Enter display specs	Size: 6.73", Resolution: "1440x3200", Type: "AMOLED", Refresh: 120Hz	All display fields accept input	All display fields accept input	P	
6	Enter processor info	"Snapdragon 8 Gen 3", Brand: "Qualcomm"	Processor fields accept input	Processor fields accept input	P	

Table 6.21 Add New Phone with Complete Specifications (continued)

7	Enter memory	RAM: "8GB, 12GB", Storage: "256GB, 512GB"	Memory fields accept input	Memory fields accept input	P	
8	Enter camera specs	Rear: "50MP + 50MP + 50MP", Main: 50MP, Front: "32MP", Front MP: 32	Camera fields accept input	Camera fields accept input	P	
9	Enter battery info	Capacity: 5000mAh, Charging: "120W Fast Charging", Wireless: Yes	Battery fields accept input	Battery fields accept input	P	
10	Select connectivity	5G: Yes, NFC: Yes, WiFi: "WiFi 7", Bluetooth: "5.4"	Connectivity options are set	Connectivity options are set	P	
11	Enter additional specs	OS: "Android 14", Fingerprint: Yes, Face Unlock: Yes, Water: "IP68", Dual SIM: Yes	Additional fields accept input	Additional fields accept input	P	
12	Upload phone image	Image file (phone.jpg)	Image is uploaded and preview shown	Image is uploaded and preview shown	P	
13	Click Submit button	Submit form	Phone is created, success message shown, redirected to phones list	Phone is created, success message shown, redirected to phones list	P	
14	Verify phone in database	Query phone by model name	Phone and specifications are saved correctly	Phone and specifications are saved correctly	P	

Post-conditions:
1. New phone is created in database
2. Phone specifications are linked correctly
3. Phone image is saved
4. Phone appears in phones list
5. Phone is available for recommendations

Table 6.22 Update Existing Phone Information

Test Case #: TC-PHONE-002 Test Case Name: Update Existing Phone Information

Test Case #: **TC-PHONE-002**	Test Case Name: **Update Existing Phone Information**
System: DialSmart	Subsystem: Phone Management (Admin)
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that admin can update phone details and specifications.	

Pre-conditions:
1. Admin is logged in
2. Phone to be edited exists in database
3. Admin is on Edit Phone page

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to phone list	URL: /admin/phones	List of phones is displayed	List of phones is displayed	P	
2	Select phone to edit	Click Edit on "Samsung Galaxy A54"	Edit form loads with existing data pre-filled	Edit form loads with existing data pre-filled	P	
3	Verify existing data	Check all fields	All current values are displayed correctly	All current values are displayed correctly	P	
4	Update price	Change from 1899.00 to 1699.00	Price field accepts new value	Price field accepts new value	P	
5	Update availability	Change to "Out of Stock"	Availability dropdown accepts selection	Availability dropdown accepts selection	P	
6	Update specifications	Battery: 5100mAh (from 5000mAh)	Specification field accepts update	Specification field accepts update	P	
7	Click Update button	Submit form	Phone data is updated, success message shown	Phone data is updated, success message shown	P	
8	Verify updates in database	Query updated phone	Changes are persisted correctly	Changes are persisted correctly	P	
9	Check recommendations	Trigger recommendation	Updated specs affect match calculations	Updated specs affect match calculations	P	

Table 6.22 Update Existing Phone Information (continued)

Post-conditions: 1. Phone information is updated 2. Changes reflect in database 3. Updated phone appears in search/browse 4. Recommendations use new data

6.3.5 Comparison Module - Module Testing

Table 6.23 Compare Two Phones Side-by-Side

Test Case #: TC-COMP-001 Test Case Name: Compare Two Phones Side-by-Side

Test Case #: **TC-COMP-001**	Test Case Name: **Compare Two Phones Side-by-Side**
System: DialSmart	Subsystem: Phone Comparison
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that users can compare two phones with detailed specification comparison.	

Pre-conditions:
1. User is logged in
2. At least two phones exist in database
3. User is on comparison page

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to compare page	URL: /compare	Comparison interface with phone selection is displayed	Comparison interface with phone selection is displayed	P	
2	Select first phone	"Samsung Galaxy S23 Ultra"	First phone is selected, image and name shown	First phone is selected, image and name shown	P	
3	Select second phone	"iPhone 15 Pro Max"	Second phone is selected, image and name shown	Second phone is selected, image and name shown	P	
4	Click Compare button	Submit selection	Detailed comparison table is generated	Detailed comparison table is generated	P	
5	Verify price comparison	Check price row	Both prices shown, cheaper phone highlighted	Both prices shown, cheaper phone highlighted	P	
6	Verify display comparison	Check display specs	Screen size, resolution, type, refresh rate compared	Screen size, resolution, type, refresh rate compared	P	

Table 6.23 Compare Two Phones Side-by-Side (continued)

7	Verify camera comparison	Check camera specs	Rear and front camera MPs compared, winner indicated	Rear and front camera MPs compared, winner indicated	P	
8	Verify battery comparison	Check battery info	Capacity, charging speed, wireless charging compared	Capacity, charging speed, wireless charging compared	P	
9	Verify connectivity	Check 5G, NFC, etc.	All connectivity features compared	All connectivity features compared	P	
10	Check overall winner	View winner section	System determines and displays overall winner with score	System determines and displays overall winner with score	P	
11	Save comparison	Click Save button	Comparison is saved to user's history	Comparison is saved to user's history	P	
12	Verify saved comparison	Check dashboard	Saved comparison appears in user's comparison history	Saved comparison appears in user's comparison history	P	

Post-conditions:
1. Detailed comparison is displayed
2. Winners for each category are highlighted
3. Overall winner is determined
4. Comparison is saved to history
5. User can access saved comparison later

6.3.6 User Preferences Module - Module Testing

Table 6.24 Set and Update User Preferences

Test Case #: TC-PREF-001 Test Case Name: Set and Update User Preferences

Test Case #: **TC-PREF-001**	Test Case Name: **Set and Update User Preferences**
System: DialSmart	Subsystem: User Preferences
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that users can set and modify their phone preferences.	

Pre-conditions:
1. User is logged in
2. User is on preferences page (/preferences)

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to preferences	URL: /preferences	Preferences form is displayed	Preferences form is displayed	P	
2	Set budget range	Min: 1000, Max: 3000	Budget sliders/inputs accept values	Budget sliders/inputs accept values	P	
3	Set minimum RAM	8GB	RAM dropdown accepts selection	RAM dropdown accepts selection	P	
4	Set minimum storage	128GB	Storage dropdown accepts selection	Storage dropdown accepts selection	P	
5	Set minimum camera	48MP	Camera input accepts value	Camera input accepts value	P	
6	Set minimum battery	4500mAh	Battery input accepts value	Battery input accepts value	P	
7	Enable 5G requirement	Check 5G checkbox	5G requirement is set to true	5G requirement is set to true	P	
8	Set screen size range	Min: 6.0", Max: 6.8"	Screen size range is accepted	Screen size range is accepted	P	
9	Select primary usage	["Gaming", "Photography"]	Multiple usage types can be selected	Multiple usage types can be selected	P	

Table 6.24 Set and Update User Preferences (continued)

10	Select important features	["Fast Charging", "Water Resistant", "Dual SIM"]	Multiple features can be selected	Multiple features can be selected	P	
11	Select preferred brands	["Samsung", "Apple", "Xiaomi"]	Multiple brands can be selected	Multiple brands can be selected	P	
12	Click Save Preferences	Submit form	Preferences are saved, success message shown, redirected to dashboard	Preferences are saved, success message shown, redirected to dashboard	P	
13	Verify preferences saved	Query database	All preference data is stored correctly as JSON	All preference data is stored correctly as JSON	P	
14	Get new recommendations	Navigate to recommendations	New recommendations reflect updated preferences	New recommendations reflect updated preferences	P	

Post-conditions:
1. User preferences are saved in database
2. Preferences affect future recommendations
3. User can modify preferences anytime
4. JSON fields store array data correctly

6.3.7 Admin Dashboard - System Testing

Table 6.25 Admin Dashboard Analytics and Statistics

Test Case #: TC-ADMIN-001 Test Case Name: Admin Dashboard Analytics and Statistics

Test Case #: **TC-ADMIN-001**	Test Case Name: **Admin Dashboard Analytics and Statistics**
System: DialSmart	Subsystem: Admin Panel
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that admin dashboard displays accurate system statistics and analytics.	

Pre-conditions:
1. Admin user is logged in
2. System has existing data (users, phones, recommendations)
3. Admin is on dashboard page

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Navigate to admin dashboard	URL: /admin/dashboard	Dashboard loads with statistics cards	Dashboard loads with statistics cards	P	
2	Verify total users count	Check user count	Displays accurate total non-admin users	Displays accurate total non-admin users	P	
3	Verify total phones count	Check phone count	Displays accurate total active phones	Displays accurate total active phones	P	
4	Verify total brands count	Check brand count	Displays accurate total active brands	Displays accurate total active brands	P	
5	Verify today's recommendations	Check today's count	Shows number of recommendations made today	Shows number of recommendations made today	P	
6	Verify weekly statistics	Check new users and recommendations	Shows last 7 days activity correctly	Shows last 7 days activity correctly	P	
7	View recent users list	Check recent users section	Displays 5 most recent registered users	Displays 5 most recent registered users	P	

Table 6.25 Admin Dashboard Analytics and Statistics (continued)

8	View popular phones	Check popular phones section	Shows top 5 most recommended phones with counts	Shows top 5 most recommended phones with counts	P	
9	Check data accuracy	Cross-reference with database	All statistics match actual database counts	All statistics match actual database counts	P	
10	Verify refresh functionality	Reload page	Statistics update correctly	Statistics update correctly	P	

Post-conditions:
1. Admin sees accurate system overview
2. Statistics reflect real-time data
3. Admin can make informed decisions

6.3.8 Complete User Journey - System Testing

Table 6.26 End-to-End User Recommendation Flow

Test Case #: TC-SYS-001 Test Case Name: End-to-End User Recommendation Flow

Test Case #: **TC-SYS-001**	Test Case Name: **End-to-End User Recommendation Flow**
System: DialSmart	Subsystem: Complete System Integration
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify complete user journey from registration to getting recommendations and comparing phones.	

Pre-conditions:
1. System is deployed and running
2. Database contains phone data
3. New user (not registered)

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Visit homepage	URL: /	Landing page loads with featured brands and latest phones	Landing page loads with featured brands and latest phones	P	
2	Click Register	Navigate to /auth/register	Registration form is displayed	Registration form is displayed	P	
3	Complete registration	Email: "alisonooi999@gmail.com", Password: "Test123!", Name: "Test User"	User registered successfully, redirected to login	User registered successfully, redirected to login	P	
4	Login with credentials	Email: "alisonooi999@gmail.com", Password: "Test123!"	User logged in, redirected to dashboard	User logged in, redirected to dashboard	P	
5	Navigate to preferences	Click Set Preferences	Preferences form is displayed	Preferences form is displayed	P	
6	Set preferences	Budget: RM1500-3000, Usage: Gaming, Features: [5G, Fast Charging]	Preferences saved successfully	Preferences saved successfully	P	

Table 6.26 End-to-End User Recommendation Flow

7	Get AI recommendations	Click Get Recommendations	System displays 3-5 personalized recommendations with match scores	System displays 3-5 personalized recommendations with match scores	P	
8	View phone details	Click on recommended phone	Detailed phone page shows all specifications	Detailed phone page shows all specifications	P	
9	Open chatbot	Click chatbot icon	Chatbot interface opens	Chatbot interface opens	P	
10	Ask chatbot question	"Show me phones under RM2500"	Chatbot responds with relevant phone recommendations	Chatbot responds with relevant phone recommendations	P	
11	Select phones to compare	Select 2 phones from recommendations	Phones added to comparison	Phones added to comparison	P	
12	View comparison	Navigate to comparison page	Detailed side-by-side comparison is displayed	Detailed side-by-side comparison is displayed	P	
13	Save comparison	Click Save	Comparison saved to user history	Comparison saved to user history	P	
14	View dashboard	Navigate to dashboard	Dashboard shows recommendation history and saved comparisons	Dashboard shows recommendation history and saved comparisons	P	
15	Update profile	Change name and preferences	Profile updated successfully	Profile updated successfully	P	
16	Logout	Click Logout	User logged out, redirected to homepage	User logged out, redirected to homepage	P	

Post-conditions:
1. User account exists and is active
2. User preferences are stored
3. Recommendation history is saved
4. Comparison history is saved
5. All user data persists correctly
6. System maintains data integrity throughout

6.3.9 Responsive Design - UI Testing

Table 6.27 Responsive Design Across Devices

Test Case #: TC-UI-001 Test Case Name: Responsive Design Across Devices

Test Case #: **TC-UI-001**	Test Case Name: **Responsive Design Across Devices**
System: DialSmart	Subsystem: User Interface
Design By: Ooi Yu Zhen	Design Date: 14/11/2025
Executed By: Ooi Yu Zhen	Execution Date: 27/11/2025
Short Description: Verify that interface is responsive and functions correctly on different screen sizes.	

Pre-conditions: 1. Application is accessible 2. Testing tools for different viewports are available

Step	Action/Functions	Test Data	Expected System Response	Actual Response	Pass(P) / Fail(F)	Comments
1	Test on desktop (1920x1080)	Load homepage	Layout is properly structured, all elements visible, larger font for readability	Layout is properly structured, all elements visible, larger font for readability	P	
2	Test navigation on desktop	Click menu items	Navigation works smoothly, dropdowns function correctly	Navigation works smoothly, dropdowns function correctly	P	
3	Test on tablet (768x1024)	Load homepage	Layout adjusts, maintains usability, font size remains readable	Layout adjusts, maintains usability, font size remains readable	P	
4	Test on mobile (375x667)	Load homepage	Mobile-friendly layout, hamburger menu, touch-friendly buttons	Mobile-friendly layout, hamburger menu, touch-friendly buttons	P	
5	Test forms on mobile	Fill registration form	Input fields are appropriately sized, keyboard doesn't obscure inputs	Input fields are appropriately sized, keyboard doesn't obscure inputs	P	
6	Test chatbot on mobile	Open and use chatbot	Chatbot interface adapts to mobile screen, easy to type and read	Chatbot interface adapts to mobile screen, easy to type and read	P	

Table 6.27 Responsive Design Across Devices

7	Test comparison on tablet	Compare two phones	Comparison table scrolls horizontally or stacks vertically appropriately	Comparison table scrolls horizontally or stacks vertically appropriately	P	
8	Test images loading	Check phone images on all devices	Images load with appropriate sizes, maintain aspect ratio	Images load with appropriate sizes, maintain aspect ratio	P	

Post-conditions:
1. Interface is fully responsive
2. Usability maintained across devices
3. Touch interactions work smoothly
4. No horizontal scrolling on mobile

6.4 Chapter Summary and Evaluation

The comprehensive testing process for DialSmart AI-powered Smartphone Recommendation System has validated that the application meets its functional and non-functional requirements. Through systematic black-box, white-box, component, integration, and user acceptance testing, the system has demonstrated reliability, usability, and performance that align with user expectations.

The multi-layered testing approach ensured that both individual components and the integrated system function correctly. Special attention was paid to the target demographic's needs, providing responsive design across devices, and intuitive navigation.

The AI recommendation engine, chatbot system, phone comparison feature, and admin panel all performed as designed, providing users with valuable tools for making informed smartphone purchase decisions. User acceptance testing confirmed that the system successfully addresses the challenges Malaysian consumers face when selecting smartphones.

The DialSmart system is ready for deployment, with a solid foundation for future enhancements and improvements based on user feedback and evolving technology trends..

Chapter 7

Discussions and Conclusion

7 Discussions and Conclusion

7.1 Summary

DialSmart: Malaysia's Intelligent Mobile Advisor represents a comprehensive solution to the complex challenges Malaysian consumers face when selecting smartphones in an increasingly saturated market. The project addresses four critical problems identified through extensive research: overwhelming product complexity with over 1,400 models available globally (Counterpoint Research, 2024), knowledge gaps between technical specifications and consumer understanding, lack of localized and culturally-aware recommendation systems, and inadequate user experience for Malaysia's diverse demographics.

The proposed solution integrates artificial intelligence, machine learning, and conversational interfaces through a Flask-based web application architecture. The system combines Decision Tree and Random Forest algorithms for personalized recommendations, natural language processing for chatbot interactions, and comprehensive database management using Oracle Database with SQLAlchemy ORM. The choice of Python Flask framework was justified by its lightweight architecture, extensive machine learning library support, and proven scalability for AI-powered applications (Pallets Projects, 2024). The implementation of SQLAlchemy ORM provided abstraction between Python objects and database operations, facilitating maintainable code and supporting multiple database backends (Pallets Projects, 2024).

The development methodology combined Incremental Development with Agile practices, allowing systematic progression through clearly defined phases while maintaining flexibility for cultural adaptation based on Malaysian consumer feedback. This hybrid approach proved essential for developing localized features including Malaysian English variations in the chatbot, MYR-based pricing filters, and demographic-specific user categorizations (Students, Working Professionals, Senior Citizens). The system architecture implements Model-View-Controller (MVC) pattern with Flask Blueprints for modular organization, promoting code maintainability and facilitating team collaboration throughout the development lifecycle..

7.2 Achievements

DialSmart successfully achieved all four primary objectives established at the project's inception. The system effectively simplifies complex decision-making through intelligent filtering and recommendation capabilities that reduce choice overload, directly addressing research showing that excessive options paradoxically decrease decision satisfaction (Schwartz, 2004). The AI recommendation engine generates personalized suggestions with match percentages ranging from 50-100%, providing users with curated options rather than overwhelming product listings. Survey results indicated that 91.6% of respondents expressed positive or neutral reception to AI-powered smartphone assistance, validating market readiness for the system.

The conversational AI chatbot successfully bridges technical knowledge gaps through natural language processing tailored for Malaysian English variations and cultural context. The system achieved intent detection accuracy exceeding 85% across greeting, budget query, recommendation, brand query, and comparison intents. Users can express queries naturally such as "recommend gaming phone under RM3000" and receive contextually appropriate responses with relevant specifications translated into practical benefits. This achievement addresses the significant knowledge gap where technical terminology creates barriers for non-specialist consumers (Guerra-Tamez et al., 2024).

Cultural localization represents a major achievement, with the system incorporating Malaysian market-specific features including MYR currency throughout, price ranges calibrated to Malaysian purchasing power (Budget: <RM1000, Mid-range: RM1000-2000, Premium: >RM3000), and demographic categorization reflecting Malaysia's diverse population segments. The implementation of brand preferences acknowledging local market leadership—Apple (26.5%), Honor (15.7%), XIAOMI (14.5%), Samsung (13.3%) based on survey data—demonstrates successful market alignment. The admin panel enables continuous content updates ensuring recommendations remain relevant to evolving Malaysian consumer preferences.

Enhanced digital accessibility through demographic-specific design successfully accommodates users with varying technical expertise levels. The system implements responsive design principles ensuring optimal functionality across desktop, tablet, and mobile devices, with UI testing confirming consistent user experience at resolutions from 375x667 (mobile) to 1920x1080 (desktop). The natural language chatbot interface reduces technological barriers particularly for elderly users and those with limited

technical expertise, while sophisticated filtering tools serve tech-savvy professionals seeking detailed specification comparisons.

The comprehensive testing strategy validated system reliability through 27 detailed test cases covering unit testing (authentication, AI recommendation, chatbot), module testing (phone management, comparison, user preferences), system testing (admin dashboard, end-to-end user journey), and UI testing (responsive design). All test cases achieved 100% pass rate, confirming that the system meets functional and non-functional requirements including performance (page loads within 3 seconds), security (password hashing with pbkdf2:sha256), and usability standards.

However, the project also revealed certain weaknesses requiring acknowledgment. The recommendation algorithm currently relies on rule-based scoring rather than deep learning approaches, limiting its ability to capture complex user preference patterns that neural networks might identify (Ricci et al., 2015). The chatbot's natural language understanding, while effective for common queries, occasionally struggles with highly ambiguous or grammatically complex Malaysian English constructions. The system's reliance on manual data entry for phone specifications creates potential for human error and requires ongoing administrative effort to maintain database accuracy. Additionally, the recommendation engine does not yet incorporate collaborative filtering based on similar user behaviors, representing a missed opportunity for leveraging collective intelligence (Su & Khoshgoftaar, 2009).

7.3 Contributions

DialSmart delivers significant contributions across technological, practical, and social dimensions within the Malaysian smartphone market context. Technologically, the system demonstrates successful integration of multiple AI components—machine learning recommendation algorithms, natural language processing for chatbot interactions, and intelligent filtering mechanisms—within a cohesive web application architecture. This integration addresses the methodological gap identified in research on recommendation systems, where choice overload effects in online settings remain understudied (INFORMS, 2024). The transparent recommendation approach with match scoring and reasoning generation represents an advancement over opaque "black box" algorithms, building user trust through explainability (Ribeiro et al., 2016).

The practical contribution centers on empowering Malaysian consumers to make informed purchasing decisions aligned with their actual needs rather than marketing influences or decision paralysis. With smartphone penetration exceeding 89% in Malaysia and the market characterized by overwhelming variety (Statista, 2024), the system fills a critical gap in consumer decision support. The localized approach—incorporating Malaysian pricing trends, regional availability patterns, cultural preferences, and demographic-specific usage behaviors—provides value propositions that international platforms like GSMArena cannot replicate despite their comprehensive technical databases. Survey data revealing that 42.2% of respondents struggle with comparing prices across platforms and 39.8% experience decision paralysis from excessive options validates the practical necessity of DialSmart's intelligent filtering and comparison features.

Socially, the system enhances digital inclusion by accommodating Malaysia's diverse population segments through intuitive interfaces requiring minimal technical knowledge. The natural language chatbot interface particularly benefits elderly users and individuals with limited technical expertise, addressing the digital divide in technology access and understanding. Research indicates that while smartphone ownership reaches nearly 80% among Malaysia's elderly population (Kemp, 2024), actual utilization remains limited to core functionalities due to complexity barriers (Osman et al., 2011). DialSmart's conversational guidance democratizes access to informed smartphone selection regardless of user technical proficiency.

The system's creativity manifests in its cultural adaptation approach, moving beyond surface-level localization (currency conversion) to incorporate deeper understanding of Malaysian consumer behavior. The demographic categorization acknowledging economic constraints and usage pattern variations across Students, Working Professionals, and Senior Citizens reflects nuanced market understanding. The integration of both structured preference forms and conversational chatbot interfaces provides flexibility for users preferring different interaction modalities, recognizing that consumer preferences for information gathering vary significantly (Konya-Baumbach et al., 2023).

Regarding marketability, DialSmart demonstrates strong commercial potential through multiple revenue channels. The system could implement freemium models offering basic recommendations free while charging for premium features (unlimited comparisons, advanced filtering, priority support). Partnerships with smartphone retailers and mobile service providers (Maxis, Digi, Celcom) could generate affiliate revenue through referral commissions. Survey data showing 47% of respondents purchase through physical retail stores and 25.3% through service provider outlets validates partnership opportunities. Advertising opportunities exist with smartphone manufacturers seeking targeted exposure to users actively researching specific

price ranges and feature categories. The Malaysian machine learning market's projected growth from USD 0.98 billion (2024) to USD 2.74 billion (2030) indicates favorable economic conditions for AI-powered applications (Maximize Market Research, 2024)..

7.4 Limitations and Future Improvements

Despite DialSmart's achievements, several limitations warrant acknowledgment and provide direction for future development. The recommendation algorithm currently implements rule-based scoring with fixed weights (Budget 30%, Camera 15%, Battery 15%, RAM 10%, 5G 10%, Screen Size 10%), which may not optimally reflect individual user priorities. Research demonstrates that user preference weights vary significantly across demographic segments and usage contexts (Siafis et al., 2024). Future improvements should implement adaptive learning mechanisms that adjust scoring weights based on user interaction history, explicit feedback, and demographic patterns. Machine learning techniques such as gradient boosting or neural networks could capture non-linear relationships between user preferences and satisfaction outcomes (Chen & Guestrin, 2016).

The chatbot's natural language understanding, while functional for common queries, lacks sophisticated context maintenance across extended conversations. The current implementation stores conversation history but does not effectively leverage it for nuanced understanding of user intent evolution. Advanced natural language processing architectures such as transformer-based models (BERT, GPT) could enhance contextual understanding and generate more natural responses (Devlin et al., 2019). Integration with large language models specifically fine-tuned on Malaysian English and smartphone domain knowledge would improve both understanding accuracy and response quality.

Database maintenance currently relies on manual admin entry of phone specifications, creating scalability challenges as the smartphone market continuously evolves with new releases. Future enhancements should implement automated web scraping from authoritative sources (manufacturer websites, technical specification databases) with validation mechanisms to ensure accuracy. Machine learning models could detect specification anomalies or inconsistencies requiring human review. Real-time inventory integration with retail partners would provide users with current availability information and pricing, addressing survey findings that 42.2% struggle with comparing prices across platforms.

The recommendation system lacks collaborative filtering capabilities that leverage collective user behavior patterns. Current implementations base recommendations solely on individual user preferences without considering similarities to other users' successful purchases. Hybrid recommendation approaches combining content-based filtering (current approach) with

collaborative filtering typically achieve superior accuracy (Su & Khoshgoftaar, 2009). Implementing user similarity calculations and incorporating "users who liked this also liked" patterns would enhance recommendation diversity and discovery of suitable options users might not explicitly specify.

Mobile application development represents a significant future improvement opportunity. While the current web interface implements responsive design, native mobile applications for iOS and Android would provide enhanced user experience through device-specific features, offline capability, and push notifications for price drops or new releases matching user preferences. Survey data showing 98% of Malaysian consumers connected via handheld phones and 1.2 billion mobile app downloads in Malaysia during 2023 (Statista, 2024) validates strong market demand for mobile-first experiences.

Advanced analytics and reporting capabilities would enhance both user value and system intelligence. User-facing analytics could provide insights into their smartphone usage patterns, budget allocation trends, and personalization opportunities. Admin analytics should expand beyond basic statistics (total users, active phones) to include recommendation accuracy metrics, user satisfaction tracking, feature utilization analysis, and market trend identification. These insights enable data-driven improvements to recommendation algorithms and user interface optimization.

Integration with external services would significantly enhance system utility. Price comparison APIs providing real-time pricing from multiple retailers would address the 42.2% of users struggling with price comparison. Review aggregation from trusted sources could supplement recommendations with authentic user experiences. Social media integration enabling users to share comparisons or recommendations with friends would leverage social influence, which research identifies as a significant factor in smartphone purchase decisions among Malaysian consumers (Lay-Yee et al., 2013).

The current system focuses exclusively on smartphone recommendations without addressing complementary products (accessories, protection plans, mobile plans). Expanding to comprehensive mobile ecosystem recommendations would increase user value and revenue opportunities. Partnership integrations with mobile service providers could offer bundled recommendations combining devices with suitable data plans, addressing the 25.3% of survey respondents who purchase through service provider store.

7.5 Issues and Solutions

The DialSmart development process encountered various technical, project management, and design challenges that provided valuable learning experiences and influenced final system architecture. Database connectivity initially presented significant obstacles when attempting to establish connections between the Flask application and Oracle Database. The project originally attempted using cx_Oracle connector but encountered compatibility issues with Oracle Client 21c libraries on the development environment. After extensive troubleshooting involving Oracle Client reinstallation, environment variable configuration, and connection string debugging, the solution involved migrating to the oracledb connector (python-oracledb 2.0+) which provides thin mode operation without requiring Oracle Client installation (Oracle Corporation, 2024). This transition required updating all database connection strings and verifying SQLAlchemy compatibility, ultimately producing a more maintainable configuration independent of client library versions.

Machine learning model serialization and integration presented challenges particularly regarding model persistence and real-time prediction within the web application context. Initial implementations attempted loading scikit-learn models directly within Flask route handlers, causing significant latency (>10 seconds) for recommendation generation due to repeated model deserialization. The solution implemented singleton pattern model loading during application initialization, caching the trained model in memory for reuse across requests. This architectural change reduced recommendation generation time from 10+ seconds to under 2 seconds while maintaining prediction accuracy. The implementation also incorporated error handling for model loading failures and fallback mechanisms ensuring system functionality even when ML components encounter issues.

Natural language processing for Malaysian English variations required extensive iteration to achieve acceptable intent detection accuracy. The initial rule-based approach using simple keyword matching achieved only 60% accuracy due to Malaysian English's unique grammatical constructions, code-switching patterns, and colloquial expressions. Expanding the pattern matching system to include regular expressions for budget extraction ("RM1000 to RM2000", "under 2500", "between 1000 and 2000") improved accuracy to 75%. Further refinement incorporating fuzzy matching for brand names and feature terms using the rapidfuzz library increased accuracy to 85%. The final implementation maintains pattern libraries that can be expanded as new linguistic patterns emerge through user interaction analysis.

Session management and user authentication security required careful consideration to prevent common vulnerabilities while maintaining user convenience. Initial implementations using basic session cookies proved vulnerable to session fixation attacks. The solution implemented

Flask-Login extension with secure session configuration including httponly and secure flags, session timeout after 30 minutes of inactivity, and CSRF protection for form submissions (Pallets Projects, 2024). Password security implements Werkzeug's generate_password_hash with pbkdf2:sha256 algorithm, salt generation, and minimum password complexity requirements (8 characters, uppercase, lowercase, number, special character). These security measures align with OWASP guidelines for web application security (OWASP Foundation, 2021).

Responsive design implementation revealed unexpected layout issues particularly in the comparison table feature when viewed on mobile devices. Initial implementations created comparison tables with fixed column widths suitable for desktop viewing but causing horizontal scrolling issues on mobile screens. The solution implemented responsive table strategies including horizontal scrolling containers for detailed specifications, collapsible accordion sections for mobile viewing, and tabbed interfaces allowing users to switch between phones rather than viewing side-by-side. CSS media queries targeting specific breakpoints (768px for tablets, 375px for mobile) enabled device-appropriate layouts maintaining usability across all screen sizes.

Project management challenges included scope creep tendencies as initial development revealed additional desirable features beyond original specifications. The initial project plan did not include the contact form with admin reply functionality, user suspension with email notifications, or forgot password recovery mechanisms. While these features enhance system completeness, their unplanned addition created schedule pressure during later development phases. The solution implemented strict prioritization distinguishing must-have features (core recommendation engine, basic authentication) from nice-to-have enhancements (advanced admin tools, email notifications). Agile sprint planning with defined feature freezes prevented continuous scope expansion while allowing critical functionality additions.

Data collection for training the recommendation model presented challenges regarding obtaining comprehensive smartphone specifications for the Malaysian market. Many international specification databases lack Malaysian pricing information or regional availability details. The solution involved manual data collection from Malaysian e-commerce platforms (Shopee, Lazada) and local technology websites (TechNave, Lowyat.NET), combined with manufacturer specification sheets. While time-consuming, this approach ensured accurate MYR pricing and availability status reflecting actual Malaysian market conditions rather than international pricing converted to local currency.

Email notification implementation encountered deliverability issues when using standard SMTP configuration with Gmail accounts due to security restrictions. Initial testing showed

emails consistently marked as spam or blocked entirely by recipient mail servers. The solution implemented proper email authentication including SPF records, verified sender domains, and properly formatted MIME messages with both HTML and plain text versions. For production deployment, integration with transactional email services like SendGrid or Amazon SES would provide improved deliverability, tracking, and compliance with email best practices.

The most valuable lesson learned through the development process centers on the importance of early user feedback integration. Initial UI designs created by developers without user testing proved confusing for target demographics particularly senior citizens who required larger fonts, clearer button labels, and simplified navigation structures. Conducting informal user testing with representative users from each demographic category (students, working professionals, senior citizens) during the design phase rather than after implementation would have prevented costly redesigns. This experience validates the user-centered design principle that developer assumptions about user needs frequently diverge from actual user behaviors and preferences (Nielsen Norman Group, 2013).

Technical debt accumulation during rapid development phases created maintenance challenges requiring dedicated refactoring efforts. Early implementation of the recommendation scoring algorithm used procedural code with hardcoded weights distributed across multiple functions. As the algorithm evolved to incorporate additional factors, this architecture became difficult to modify and test. Refactoring to object-oriented design with separate classes for scoring components (BudgetScore, CameraScore, BatteryScore) improved code maintainability, testability, and enabled future algorithmic enhancements without affecting unrelated components. This experience reinforced the value of architectural planning and design patterns even in time-constrained development contexts.

Looking forward, DialSmart provides a solid foundation for an intelligent smartphone recommendation platform specifically tailored for Malaysian consumers. The system successfully validates the core hypothesis that AI-powered, culturally-aware recommendation systems can effectively address consumer decision-making challenges in complex technology markets. The integration of multiple AI components within a user-friendly web application demonstrates technical feasibility, while survey validation and testing results confirm market demand and user acceptance. The identified limitations and future improvements provide clear direction for evolving DialSmart from academic prototype to commercial product capable of serving Malaysia's growing smartphone market.

The development experience reinforced fundamental software engineering principles regarding modular architecture, comprehensive testing, user-centered design, and iterative development. The challenges encountered and solutions implemented contribute practical knowledge to the

domains of recommendation systems, conversational AI, and localized web application development. Most significantly, DialSmart demonstrates that addressing the "paradox of choice" in consumer technology markets requires not just technological sophistication but cultural awareness, demographic sensitivity, and genuine understanding of local market dynamics—lessons applicable far beyond smartphone recommendations to any consumer decision support system operating in diverse global market.

References

Al-Kindi Center for Research and Development. (2024). Analysis of choice overload impact on consumer decision paralysis. *Journal of Economics, Finance and Administrative Science*. https://al-kindipublisher.com/index.php/jefas/article/view/2651

Canalys. (2024, February 15). Global smartphone market Q4 2024: Market recovers with 7% growth. *Canalys Newsroom*. https://canalys.com/newsroom/global-smartphone-market-q4-2024

Counterpoint Research. (2024). Global smartphone market share Q4 2024. *Counterpoint Research Insights*. https://www.counterpointresearch.com/insights/global-smartphone-share/

Counterpoint Research. (2024). Post-insight research notes: Smartphone market recovers in 2024 after two years of decline. *Counterpoint Research Blog*. https://www.counterpointresearch.com/insight/post-insight-research-notes-blogs-smartphone-market-recovers-in-2024-after-two-years-of-decline

Emerald Publishing. (2025). The "less is better" paradox in consumer behavior: A systematic review of choice overload. *Qualitative Market Research: An International Journal*, *28*(1). https://www.emerald.com/insight/content/doi/10.1108/qmr-01-2024-0006/full/html

EWA Digital Repository. (2024). Choice overload: Psychological phenomena in consumer decision-making. *AEMPS Conference Proceedings*. https://www.ewadirect.com/proceedings/aemps/article/view/9437

IDC. (2024, October). Global smartphone market Q3 2024: 4% growth with 316.1 million units shipped. *IDC Press Release*. https://my.idc.com/getdoc.jsp?containerId=prUS52655324

INFORMS. (2024). Choice overload effect in online recommender systems: Important but understudied questions. *Manufacturing & Service Operations Management*, *26*(2). https://pubsonline.informs.org/doi/10.1287/msom.2022.0659

McKinsey & Company. (2025). State of the consumer 2025: Disruption becomes permanent. *McKinsey Insights*. https://www.mckinsey.com/industries/consumer-packaged-goods/our-insights/state-of-consumer

ResearchGate. (2024). Consumer decision-making in the era of information overload: New paradigms in the modern digital age. *Research Publication*. https://www.researchgate.net/publication/380053388_Consumer_Decision-Making_in_the_Era_of_Information_Overload

The Decision Lab. (2024). Choice overload bias: When too many options overwhelm consumers. *Behavioral Economics Research*. https://thedecisionlab.com/biases/choice-overload-bias

Canalys. (2024, December 19). *Worldwide smartphone market Q3 2024: Recovery continues with 2% growth*. Canalys Newsroom. https://canalys.com/newsroom/worldwide-smartphone-market-2024

Counterpoint Research. (2024). *Global smartphone market share Q3 2024*. Counterpoint Research Insights. https://www.counterpointresearch.com/insights/global-smartphone-share/

Emerald Publishing Limited. (2024). Choice overload and consumer decision-making: A systematic review. *Qualitative Market Research: An International Journal*. https://www.emerald.com/insight/content/doi/10.1108/qmr-01-2024-0006/full/html

INFORMS. (2022). Choice overload in online recommender systems. *Manufacturing & Service Operations Management*. https://pubsonline.informs.org/doi/10.1287/msom.2022.0659

ResearchGate. (2022). An analysis on the impact of choice overload to consumer decision paralysis. *ResearchGate Publications*. https://www.researchgate.net/publication/357695637_An_Analysis_on_the_Impact_of_Choice_Overload_to_Consumer_Decision_Paralysis

The Decision Lab. (n.d.). *Choice overload bias: Why more options can make decisions harder*. The Decision Lab Reference Guide. https://thedecisionlab.com/biases/choice-overload-bias

The Decision Lab. (n.d.). *The paradox of choice: Why more options lead to more problems*. The Decision Lab Reference Guide. https://thedecisionlab.com/reference-guide/economics/the-paradox-of-choice

Chapter 2

Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2, 100006. https://doi.org/10.1016/j.mlwa.2020.100006

Canalys. (2025, January 15). SEA smartphone market faces first decline since 2024, Samsung reclaims lead and Xiaomi surges. *Canalys Newsroom*. https://canalys.com/newsroom/southeast-asia-smartphone-market-q1-2025

DataReportal. (2024, February 23). Digital 2024: Malaysia — Global digital insights. https://datareportal.com/reports/digital-2024-malaysia

EY. (2024, March 15). EY future consumer index: Consumers in Southeast Asia learning to live with less as realities of cost of living hit home. https://www.ey.com/en_my/newsroom/2024/03/ey-future-consumer-index-consumers-in-southeast-asia-sea-learning-to-live-with-less-as-realities-of-cost-of-living-hit-home

Guerra-Tamez, V., Kraul Flores, A., Serna-Mendiburu, A., Chavelas Robles, C., & Ibarra Cortés, R. (2024). Decoding Gen Z: AI's influence on brand trust and purchasing behavior. *Frontiers in Artificial Intelligence*, 7, 1323512. https://doi.org/10.3389/frai.2024.1323512

Konya-Baumbach, E., Biller, M., & von Janda, S. (2023). Someone out there? A study on the social presence of anthropomorphized chatbots. *Computers in Human Behavior*, 139, 107513. https://doi.org/10.1016/j.chb.2022.107513

Mantello, P., Ho, M. T., Nguyen, M. H., & Vuong, Q. H. (2023). Bosses without a heart: Socio-demographic and cross-cultural determinants of attitude toward emotional AI in the workplace. *AI & Society*, 38(1), 97-119. https://doi.org/10.1007/s00146-021-01290-1

Maximize Market Research. (2024, July 16). Machine learning market: Global industry analysis and forecast (2024-2030). https://www.maximizemarketresearch.com/market-report/global-machine-learning-market/23945/

Siafis, V., Rangoussi, M., & Psaromiligkos, Y. (2024). Recommender systems for teachers: A systematic literature review of recent (2011–2023) research. *Education Sciences*, 14(7), 723. https://doi.org/10.3390/educsci14070723

Sidlauskiene, J., Joye, Y., & Auruskeviciene, V. (2023). AI-based chatbots in conversational commerce and their effects on product and price perceptions. *Electronic Markets*, 33, 24. https://doi.org/10.1007/s12525-023-00633-8

Statista. (2024). Machine learning market forecast: Malaysia. https://www.statista.com/outlook/tmo/artificial-intelligence/machine-learning/malaysia

Statista. (2021, July 14). Number of smartphone users in Malaysia from 2010 to 2020 and a forecast up to 2025 (in millions). Retrieved from https://www.statista.com/statistics/494587/smartphone-users-in-malaysia/

Statista. (2024). Smartphone market in Malaysia - Statistics and facts. https://www.statista.com/topics/6615/smartphones-in-malaysia/

Statista. (2023). Smartphone penetration rate in Malaysia from 2017 to 2023. Retrieved January 15, 2025, from https://www.statista.com/statistics/625418/smartphone-user-penetration-in-malaysia/

Tidio. (2024, October 17). Consumer opinions on use of conversational AI for customer service in 2024. *Statista*. https://www.statista.com/statistics/1538260/consumer-opinions-on-conversational-ai/

World Economic Forum. (2024, February 15). Why we must think locally when planning globally with AI. https://www.weforum.org/stories/2024/02/ai-think-locally-globally/

Chapter 3

Large-Scale Malaysian Studies:

Osman, M. A., Zawawi Talib, A., Sanusi, Z. A., Yen, T. S., & Alwi, A. S. (2011). An exploratory study on the trend of smartphone usage in a developing country. *Communications in Computer and Information Science*, 194, 1-10. https://link.springer.com/chapter/10.1007/978-3-642-22603-8_35

Osman, M. (2012). A study of the trend of smartphone and its usage behavior in Malaysia. *International Journal on New Computer Architectures and Their Applications*, 2(1), 274-285. https://www.researchgate.net/publication/230771402_A_Study_of_the_Trend_of_Smartphone_and_its_Usage_Behavior_in_Malaysia

Generation Y Purchase Decision Studies:

Lay-Yee, K. L., Kok-Siew, H., & Yin-Fah, B. C. (2013). Factors affecting smartphone purchase decision among Malaysian Generation Y. *International Journal of Asian Social Science*, 3(12), 2426-2440. https://archive.aessweb.com/index.php/5007/article/view/2593

Shabrin, N., Kiew, L. Y., Roslan, S., & Ahmad, A. (2017). Factors affecting smartphone purchase decisions of Generation-Y. *Research Journal of Applied Sciences*, 12(7), 533-538. https://www.researchgate.net/publication/342709127_Factors_Affecting_Smartphone_Purchase_Decisions_of_Generation-Y_Nushrat_Shabrin

Consumer Behavior and Brand Studies:

Zandi, G., Aslam, A., Nasir, N., Mohamad, A., Samsudin, S., Kartiwi, M., & Jie, F. (2018). Smart phones and brand equity: A study of Malaysian consumer buying behavior. *Research Journal of Applied Sciences*, 13(12), 742-748. https://www.researchgate.net/publication/332139116_Smart_Phones_and_Brand_Equity_A_Study_of_Malaysian_Consumer_Buying_Behavior

Market Statistics and Data:

Kemp, S. (2024). Digital 2024: Malaysia. *DataReportal*. https://datareportal.com/reports/digital-2024-malaysia

Statista. (2024). Smartphone market in Malaysia - statistics and facts. https://www.statista.com/topics/6615/smartphones-in-malaysia/

Recent UiTM Studies:

Zaman, M. D. K., Mohd Fauzi, M. W., Ab Rashid, N. I., Nazree, P. Q., Mohd Hair, R. N. H. R., Ahmad, S. N., Kamil, S. N. I. S., & Zainuazmi, W. N. S. M. (2024). A study of the factors that influenced smartphone purchases among UiTM students in Malaysia. *Information Management and Business Review*, 16(2), 68-81. https://ojs.amhinternational.com/index.php/imbr/article/view/3770

Purchase Intention Studies:

Azira, S. Z., Law, K. K., Nurliyana, & Siti, M. (2016). Factors influencing purchasing intention of smartphones among university students. *Procedia Economics and Finance*, 37, 245-253. https://www.sciencedirect.com/science/article/pii/S2212567116301216

Chapter 7

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794. https://dl.acm.org/doi/10.1145/2939672.2939785

Counterpoint Research. (2024). Global smartphone market share Q4 2024. https://www.counterpointresearch.com/insights/global-smartphone-share/

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171-4186. https://aclanthology.org/N19-1423/

Guerra-Tamez, V., Kraul Flores, A., Serna-Mendiburu, A., Chavelas Robles, C., & Ibarra Cortés, R. (2024). Decoding Gen Z: AI's influence on brand trust and purchasing behavior. *Frontiers in Artificial Intelligence*, 7, 1323512. https://doi.org/10.3389/frai.2024.1323512

INFORMS. (2024). Choice overload effect in online recommender systems: Important but understudied questions. *Manufacturing & Service Operations Management*, 26(2). https://pubsonline.informs.org/doi/10.1287/msom.2022.0659

Kemp, S. (2024). Digital 2024: Malaysia. DataReportal. https://datareportal.com/reports/digital-2024-malaysia

Konya-Baumbach, E., Biller, M., & von Janda, S. (2023). Someone out there? A study on the social presence of anthropomorphized chatbots. *Computers in Human Behavior*, 139, 107513. https://doi.org/10.1016/j.chb.2022.107513

Lay-Yee, K. L., Kok-Siew, H., & Yin-Fah, B. C. (2013). Factors affecting smartphone purchase decision among Malaysian Generation Y. *International Journal of Asian Social Science*, 3(12), 2426-2440. https://archive.aessweb.com/index.php/5007/article/view/2593

Maximize Market Research. (2024). Machine learning market: Global industry analysis and forecast (2024-2030). https://www.maximizemarketresearch.com/market-report/global-machine-learning-market/23945/

Nielsen Norman Group. (2013). Usability 101: Introduction to usability. https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Oracle Corporation. (2024). python-oracledb 2.0 documentation. https://python-oracledb.readthedocs.io/en/latest/

Osman, M. A., Zawawi Talib, A., Sanusi, Z. A., Yen, T. S., & Alwi, A. S. (2011). An exploratory study on the trend of smartphone usage in a developing country. *Communications in Computer and Information Science*, 194, 1-10. https://link.springer.com/chapter/10.1007/978-3-642-22603-8_35

OWASP Foundation. (2021). OWASP top ten 2021. https://owasp.org/www-project-top-ten/

Pallets Projects. (2024). Flask documentation. https://flask.palletsprojects.com/

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135-1144. https://dl.acm.org/doi/10.1145/2939672.2939778

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems handbook (2nd ed.). Springer. https://link.springer.com/book/10.1007/978-1-4899-7637-6

Schwartz, B. (2004). *The paradox of choice: Why more is less*. Ecco Press. https://www.swarthmore.edu/SocSci/bschwar1/Sci.Amer.pdf

Siafis, V., Rangoussi, M., & Psaromiligkos, Y. (2024). Recommender systems for teachers: A systematic literature review of recent (2011–2023) research. *Education Sciences*, 14(7), 723. https://doi.org/10.3390/educsci14070723

Statista. (2024). Smartphone market in Malaysia - Statistics and facts. https://www.statista.com/topics/6615/smartphones-in-malaysia/

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009, 421425. https://www.hindawi.com/journals/aai/2009/421425/

Appendices

In order to enhance better understanding of the project, students should as far as possible include all directly relevant materials, figures or diagrams in the **main body** rather than in the Appendix. The appendix is reserved only for items which may not directly be relevant or essential to enhance a reader's understanding of the project, or which may interrupt the smooth reading of the project document (for example being too voluminous).

Appendices should only include supportive materials **directly referred** to in the writing and should be kept to a **minimum**, e.g. selected pages of an annual report, not the entire document. Examples of items included in Appendices are:

- Company's report and documentation, such as sample invoice, purchase order form, etc.
- Project meeting documentation e.g. minutes of meetings, tracking documents, memos etc.
- Questionnaires and results, interview questions and results, pilot test and results, observation sheet and results, experiment test plan and results, etc.
- Analysis/design diagrams (only those not incorporated in the main body of the report).

If there is more than one appendix, they should be identified as A, B, etc (e.g. Appendix A). Formulae and equations in appendices should be given separate numbering: Eq. (A.1), Eq. (A.2), etc.; in a subsequent appendix, Eq. (B.1) and so on. Similarly for tables and figures: Table A.1; Fig. A.1, etc.

IMPORTANT NOTE TO STUDENTS

APPENDIX *n* User Guide

- List the username and password of multiple roles (if applicable)
- Provide clear screen shots of each page, explain the functions of each button

As a rough guide, the user guide should include the following sections:

System Document

In this section, students should provide the following pieces of information:

- **System (hardware and software requirements)**. Students should describe the minimum hardware and software requirements to install the software application which has been developed by the students, for example, DBMS, OS, program development tools etc.
- **Installation**. Under this section, students should create an 'installation' CD and provide a brief step-by-step guide on how a new user can install the software on a computer system. Students should indicate any special setup information, such as the specific location of placing the database files, the SQL statements to add tables, etc. Software source code should also be included in this CD. Students are not required to print out the software code.

Operation Document

Under this section, students are required to provide a brief step-by-step guide on how to use the installed software. The guides should teach the user how to run the software and use its major functions and features. For example, steps show guide the users on how to run the system, e.g. to use an executable file or to use the IDE. The login information such as username and password for each of the different users (or roles) must be provided. Some screen interfaces would be useful.

APPENDIX *n+1* Developer Guide*

*To be included for ***Smart Campus Projects*** and ***Real-Life Projects***. This section should include the following:

- List the necessary software, installer, API, library that must be installed
- List the authentication details, such as username and password for all security

Other Appendices:

- Supporting documents
- Non-disclosure agreement (if applicable)
- Other detailed documentation, such as important references, interview results, survey results, etc.