



Strathmore
UNIVERSITY

Strathmore University
SU+ @ Strathmore
University Library

[Electronic Theses and Dissertations](#)

2017

A Recommender system for rental properties

Nicolas Nahimana Guy
Faculty of Information Technology (FIT)
Strathmore University

Follow this and additional works at <http://su-plus.strathmore.edu/handle/11071/5604>

Recommended Citation

Guy, N. N. (2017). *A Recommender system for rental properties* (Thesis). Strathmore University.

Retrieved from <http://su-plus.strathmore.edu/handle/11071/5604>

This Thesis - Open Access is brought to you for free and open access by DSpace @ Strathmore University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DSpace @ Strathmore University. For more information, please contact librarian@strathmore.edu

A RECOMMENDER SYSTEM FOR RENTAL PROPERTIES

Guy Nicolas Nahimana

Master in Computer-Based Information Systems

June 2017

A RECOMMENDER SYSTEM FOR RENTAL PROPERTIES

Guy Nicolas Nahimana

**Submitted in partial fulfilment of the requirements for the
Degree of Master in Computer-Based Information Systems at
Strathmore University**

Faculty of Information Technology

Strathmore University

Nairobi, Kenya

June 2017

This dissertation is available for Library use on the understanding that it is copyright material and that no quotation from this dissertation may be published without proper acknowledgement.

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another author except where due reference is made in the dissertation itself.

© No part if this dissertation may be reproduced without the permission of the author and of Strathmore University.

Guy Nicolas Nahimana

Signature

Date

Approval

The dissertation of Guy Nicolas Nahimana has been reviewed and approved by the following:

Dr. Bernard Shibwabo

Lecturer, Academic Director, Faculty of Information Technology
Strathmore University

Dr. Joseph Orero

Dean, Faculty of Information Technology
Strathmore University

Associate Professor Ruth Kiraka

Dean, School of Research & Graduate Studies
Strathmore University

ABSTRACT

Locating products or services online that meet users' needs is increasingly difficult due to the large pool of choices to consider before arriving at the desired one. A user may spend a considerable amount of time exploring numerous online resources to locate items that fit his requirements. Furthermore, users may not always express their preferences in a manner that easily matches them to items that could meet them. Searching for items online has been done mainly through database queries that return a list of the most suitable items. Recommender systems technology can be applied to ease the task of locating desired items online. This study proposes a recommender system that enables users to carry out a preference-based search on rental properties and enables them to refine those preferences using example-critiquing in case they are not satisfied with initial search results. This recommendation approach has been shown to provide more accurate search results. This research adopted the Object-Oriented Systems Analysis and Design (OOAD) approach to the development of the system. The system was developed as a Web application using the Ruby on Rails framework. Furthermore, the system was tested to ascertain that it performed as designed.

TABLE OF CONTENTS

Declaration	iii
Approval	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF EQUATIONS	ix
OPERATIONAL DEFINITION OF TERMS	x
ACKNOWLEDGEMENTS	xi
DEDICATION	xii
CHAPTER 1: INTRODUCTION	1
1.1. Background to the study	1
1.2. Problem definition	2
1.3. Research objectives	3
1.4. Research questions	3
1.5. Significance of the study	3
1.6. Scope of the study	3
CHAPTER 2: LITERATURE REVIEW	4
2.1. Introduction	4
2.2. Recommender systems	4
2.2.1. <i>Evolution of recommender systems</i>	4
2.2.2. <i>Recommender systems function</i>	5
2.2.3. <i>Recommendation approaches</i>	6
2.2.3.1. Collaborative filtering	7
2.2.3.2. Content-based filtering	7
2.2.3.3. Knowledge-based systems	8
2.2.3.4. Demographic-based systems	8
2.2.3.5. Community-based systems	9
2.2.3.6. Hybrid recommender systems	9
2.3. Critiquing-based recommender systems	9
2.3.1. <i>Natural language dialogue-based systems</i>	11
2.3.2. <i>System-suggested critiquing systems</i>	12
2.3.3. <i>User-initiated critiquing systems</i>	13
2.4. Recommending rental properties	13

2.4.1.	<i>User preference elicitation using preference-based search</i>	15
2.4.2.	<i>System recommendation based on user initial preference</i>	15
2.4.3.	<i>Capturing user feedback using example-critiquing</i>	16
2.4.4.	<i>Building a preference model for a user</i>	18
2.4.5.	<i>Making model-based suggestions</i>	19
2.5.	Algorithm for generating suggestions	24
2.6.	Conceptual framework	25
CHAPTER 3: RESEARCH METHODOLOGY		27
3.1.	Introduction	27
3.2.	System Development Methodology used	27
3.2.1.	<i>Requirements gathering</i>	28
3.2.1.1.	<i>Population</i>	29
3.2.1.2.	<i>Sampling</i>	29
3.2.1.3.	<i>Data collection</i>	30
3.2.2.	<i>Systems analysis</i>	31
3.2.3.	<i>Systems design</i>	31
3.2.4.	<i>Systems implementation</i>	31
3.3.	Research design	31
3.4.	Research quality – validity, reliability and objectivity of the research	33
3.5.	Ethical considerations	34
CHAPTER FOUR: SYSTEMS ANALYSIS AND DESIGN		35
4.1.	Introduction	35
4.2.	Use cases	36
4.2.1.	<i>Use case 1: “Add property”</i>	37
4.2.2.	<i>Use case 2: “Locate property”</i>	38
4.2.3.	<i>Use case 3: “Delete property”</i>	38
4.2.4.	<i>Use case diagram</i>	39
4.3.	Object-oriented analysis	40
4.3.1.	<i>System requirements</i>	41
4.3.1.1.	<i>Ability to run on a web browser</i>	41
4.3.1.2.	<i>Ability to perform a preference-based search (PBS)</i>	41
4.3.1.3.	<i>Ability to display search results</i>	41
4.3.1.4.	<i>Ability to select a preferred rental property</i>	41
4.3.1.5.	<i>Ability to add rental property to the system, edit them and remove them</i>	41
4.3.1.6.	<i>Ability to register and log in a staff member</i>	42
4.3.2.	<i>Classes</i>	42
4.4.	Object-oriented design	43
4.4.1.	<i>Classes</i>	43

4.4.2.	<i>Class diagram</i>	44
4.4.3.	<i>Entity Relationship Diagram</i>	45
CHAPTER FIVE: SYSTEMS IMPLEMENTATION AND TESTING		47
5.1.	Introduction	47
5.2.	User interfaces	48
5.2.1.	<i>User interface for adding a new property to the database</i>	48
5.2.2.	<i>User interface for editing details of a property</i>	48
5.2.3.	<i>User interface for removing a property from the database</i>	49
5.2.4.	<i>User interface for searching property</i>	49
5.2.5.	<i>User interface for reviewing search results, selecting a preferred rental property, and revising preferences</i>	50
5.2.7.	<i>User interface for registering as a staff member</i>	52
5.2.8.	<i>User interface for authenticating the staff member</i>	52
5.3.	Organisation of the code base	53
5.4.	Database	53
5.5.	Application test results	54
CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS		57
6.1.	Introduction	57
6.2.	Conclusion	57
6.3.	Limitations of the prototype	58
6.4.	Recommendations	59
6.5.	Future works	59
REFERENCES		60
APPENDIX 1: ORIGINALITY REPORT FROM TURNITIN		63

LIST OF FIGURES

Figure 2.1: The typical interaction between users and critiquing-based recommender systems	11
Figure 2.2: Example-critiquing interaction	14
Figure 2.3: Algorithm for checking dominance between two options	25
Figure 2.4: Conceptual framework	26
Figure 4.1: Number of websites that mention relevant attributes for selecting rental properties	36
Figure 4.2: Use case 1 “Add property”	37
Figure 4.3: Use case 2 “Locate property”	38
Figure 4.4: Use case 3 “Delete property”	39
Figure 4.5: Use case diagram for the proposed system	40
Figure 4.6: Class Diagram of the Proposed Recommender System	45
Figure 4.7: Entity Relationship Diagram for the proposed recommender system	46
Figure 5.1: Application architecture of the proposed recommender system	47
Figure 5.2: User interface for adding new properties to the database	48
Figure 5.3: User interface for editing details of a property	49
Figure 5.4: User interface for removing a property from the database	49
Figure 5.5: User interface for searching property	50
Figure 5.6: User interface for reviewing search results, selecting a preferred rental property, and revising preferences	51
Figure 5.7: User interface for selecting a preferred rental property	52
Figure 5.8: User interface for registering as a staff member	52
Figure 5.9: User interface for authenticating a staff member	53
Figure 5.10: A snapshot of the ‘properties’ table in the database	54

LIST OF EQUATIONS

Equation 2.1: Utility function	16
Equation 2.2: Piecewise function for one preference expressed on one attribute	17
Equation 2.3: Piecewise function for multiple preferences expressed on the same attribute	17
Equation 2.4: Cost function	20
Equation 2.5: Computing probability of breaking the dominance in qualitative domains	22
Equation 2.6: Computing preference for one value in qualitative domain	23
Equation 2.7: Computing directional preferences	23
Equation 2.8: Step function for preference when smaller values are preferred	24
Equation 2.9: Step Function for Preference when Larger Values Are Preferred	24
Equation 2.10: Generating multiple suggestions	24
Equation 3.1: Formula for calculating a sample size	30

OPERATIONAL DEFINITION OF TERMS

A recommender system is any system that produces individualised recommendations as output or has the effect of guiding the user in a personalised way to interesting or useful objects in a large space for possible options (Burke, 2002).

Preference-based search: Given a collection $O = \{o_1, \dots, o_n\}$ of n options, preference-based search (PBS) is an interactive process that helps identify the most preferred option, called the *target* option o_t , based on a set of preferences that they have stated on the attributes of the target (Viappiani, Faltings, & Pu, 2006).

ACKNOWLEDGEMENTS

I am indebted to my research supervisor, Dr. Bernard Shibwabo, for his valuable and constructive feedback and his time that he generously dedicated to guiding my research work. I also thank Dr. Joseph Orero and Dr. Vincent Omwenga for their sage advice and guidance.

DEDICATION

To my family: for your love and support.

CHAPTER 1: INTRODUCTION

1.1. Background to the study

Recommender systems are software tools and techniques providing suggestions for items to be of use to a user, (Ricci, Rokach, Shapira, & Kantor, 2011). Users rely on these suggestions to make various decisions include what movies to watch, what news articles to read, what apartment to rent, etc. These systems have proved to be useful in helping users deal with overwhelming amount of information while they search for various items online. This study will result in the development of a recommender system for rental properties for the city of Nairobi.

As online product catalogues evolved to include high-value products such as apartments and computers, the task of locating the desired choice among a large set of options is becoming increasingly intimidating for the average customer (Chen & Pu, 2012). Consequently, many customers may experience difficulties finding what they want. Search engines can be very useful in determining what users want; however many people find it hard to match their preferences to a search query that can produce results likely to satisfy their requirements, (Viappiani & Faltings, 2006; Viappiani, Faltings, & Pu, 2006).

Recommender systems can be applied to address the problem of mapping user preferences to objects that are likely to fit them (Viappiani et al., 2006). A recommender system is any system that produces individualised recommendations as output or has the effect of guiding the user in a personalised way to interesting or useful objects in a large space for possible options (Burke, 2002).

Some of the business applications of recommender systems technology include *News Dude* for news articles, Netflix for movies and TV shows, Amazon for a variety of products, and *DubLet* for rental properties as proposed by Hurley and Wilson (2001).

The majority of Kenyan urban dwellers live in rented properties; specifically, only 18

percent of urban households own their home (The World Bank, 2011). Three factors have contributed to the rising cost of building a house thus putting homeownership out of the reach of many urban dwellers. They include high urban population estimated at 11.36 million in 2016 (The World Bank, 2016), the purchasing power of a growing middle class (Kenya Bankers Association, 2015), and the demand for houses that outweighs its supply by at least 156, 000 units annually (The World Bank, 2011).

The residential and commercial rental property sector has become a major industry in Kenya. In 2014, the real estate market was estimated at USD 4.5 billion (The Standard, 2014). The Hass Property Index estimated that 75% of clients who purchased apartments in 2014 did so to rent them (HassConsult Limited, 2014). With the increased access to Internet and the ubiquity of smartphones, many people search for rental properties online. However, locating rental properties online remains a challenge in Kenya due to a wide range of options to consider and numerous websites to visit. Applying recommender systems technology in this sector has the potential of making the search for rental properties easier, user-friendly and personalised.

1.2. Problem definition

Searching for rental properties online in Nairobi is a challenging task. Hurley and Wilson (2001) note that a prospective tenant may spend hours or days exploring numerous disparate sources of online property advertising to locate suitable candidates. Furthermore, searching real estate properties online does not benefit homebuyers in terms of time, flexibility, and intuitive results (Yuan, Lee, Kim, & Kim, 2013). This led to the research problem addressed in this study, namely that people may not find what they are looking for online, and that the available tools are largely inadequate (Viappiani et al., 2006).

As indicated above, recommender systems technology can be used to efficiently support users in matching their preferences with items that satisfy those requirements. Among the many approaches to recommendation, this study proposes the use of *preference-based search* to elicit users' requirements and *example-critiquing* to

support them in refining their preferences (Viappiani et al., 2006). The proposed system will deliver personalised rental property search results while balancing the accuracy of offered recommendations and the efforts required of users.

1.3. Research objectives

- (i) To review the factors relating to rental property recommendation
- (ii) To evaluate systems and algorithms that can be applied to recommend rental properties
- (iii) To develop a prototype of a recommender system for rental properties
- (iv) To validate the proposed system

1.4. Research questions

- (i) What are the factors that relate to rental property recommendation?
- (ii) What systems and algorithms that can be applied to recommend rental properties ?
- (iii) How can a prototype of a recommender system for rental properties be developed?
- (iv) How can the proposed system be validated?

1.5. Significance of the study

The researcher has not found prior study that attempted to apply recommender systems technology to online search for rental properties in Kenya. This study proposes a novel approach to searching and locating rental properties in Kenya and will result in a user-friendly recommender system for rental property. Furthermore, this study will guide other researchers who are interested in recommender technology.

1.6. Scope of the study

The scope of the study is to review the recommender technology currently is use, analyse algorithms that can be applied to a recommender system for rental

properties, develop the proposed recommender system for rental properties and test it.

CHAPTER 2: LITERATURE REVIEW

2.1. Introduction

This chapter accounts for the evolution of recommendation technology. Furthermore, it discusses various recommender system functions and approaches to recommendation. From this point, the chapter focuses on the specific recommendation technology adopted by this study, preference-based search using example-critiquing, and discusses in details how this technology will be applied to recommend rental properties. Finally, the chapter provides a conceptual framework that summarises the recommendation process adopted in this study.

2.2. Recommender systems

2.2.1. *Evolution of recommender systems*

The concept of recommendation dates way back before the emergence of computers. Early forms of recommendation existed among ants, cave dwellers and other animals (Sharma & Singh, 2016). Ants spread into space looking for food. If one of them finds food, it goes back to the group leaving a scented trail that guides the rest of the community to the location of the food. Individual cave dwellers discovered new food by either trying it themselves or watching others try it. If one ate a new herb and got sick, that was a recommendation to others not to eat the herb. Otherwise, the herb was considered safe for consumption.

Sharma and Singh (2016) argue that in ancient human civilisations (4,000 to 1,200 BCE), recommendations took the form of what crop to cultivate, and what religion to profess. Much later between the 14th and 18th centuries, recommendations were about which territory to conquer. Very recently, senior family members found suitable individuals to marry their younger relatives. People also asked others where to buy the best food and what destinations to visit for holidays.

The emergence of computers brought new possibilities for recommendations. The capacity of computers to provide recommendations was recognised fairly early in

the history of computing (Ekstrand, Riedl, & Konstan, 2010). *Grundy*, a computer-based librarian, used stereotypes derived from interviews to recommend books to readers who fell in those stereotypes. Soon, *Tapestry* was proposed to address overload in online information spaces. It enabled users to filter through their emails separating those from known contacts from the rest (Sharma & Singh, 2016).

Automated recommender systems based on collaborative filtering emerged in the 1990s. Some of these included Ringo for music, BellCore Videos Recommender for movies, and Jester for jokes, among others. Perhaps the most recognisable business application of recommender system is Amazon. Based on the user purchase history, browsing history, the current item the user is viewing, and other users' behaviour, Amazon can recommend items for the user to consider purchasing (Ekstrand et al., 2010).

Recommender technology has gone beyond collaborative filtering to include content-based, Bayesian inference and case-based reasoning methods (Schafer, Konstan, & Riedl, 2001). Research on recommender systems gained momentum with the launch of the Netflix Prize, a one-million-dollar reward for research that could improve by 10% the accuracy of Netflix recommendations for movies (Ekstrand et al., 2010).

2.2.2. Recommender systems function

Recommender systems play a variety of roles. These functions fall into two categories: the roles recommender systems play on behalf of the service provider and for the end-user (Pazzani, 1999). According to Pazzani (1999), a service provider may wish to use a recommender system to achieve the following:

Increased sales: the service provider would like to sell more items than those he could sell without any recommender system. This goal is achieved because the recommended items suit clients' needs. The primary purpose of using a recommender system then is to increase the conversion rate i.e. the number of customers that accept a recommendation and consume an item, compared to others

who do not do so.

Diversity of sold items: recommender systems also help users find things they may not have discovered in the absence of an explicit recommendation for those items. This way, a service provider can sell items that are unpopular in general, but that may suit specific users.

Increase user fidelity: users are more likely to revisit a website that recognises returning users and treats them as special visitors. Since recommender systems use information from previous user behaviour (ratings), the more a user interacts with the system, the more refined his user preference model becomes.

A better understanding of users' wants: recommender systems develop a description of users' preferences collected either explicitly or implicitly. The service provider can reuse this information to achieve other goals such as improving the management of the item's production or stock.

Recommender systems also play numerous roles on behalf of the end-user. According to Pazzani (1999), some these functions include the following:

Find some useful items: this involves recommending to a user a ranked list of things with predictions of how much the user would like them. Some systems do not show the predicted rating.

Find all useful items: this involves recommending all the objects that may meet users' needs. It is mostly common when the number of articles to suggest is small and in mission-critical situations such as medical and financial applications.

Suggest a sequence: instead of recommending individual items, this involves suggesting a series of articles that is pleasing as a whole. Some examples are recommending a TV show or a compilation of musical tracks.

Recommend a bundle: this involves suggesting a set of items that go well together. For instance, one may recommend a travel plan with destinations, attractions, restaurants and hotels in particular area.

2.2.3. Recommendation approaches

Various approaches have been used to develop recommender systems. This section briefly describes some of these approaches namely collaborative filtering, content-based filtering, hybrid recommender systems, knowledge-based recommender systems, community-based recommender systems, and demographic-based recommender systems.

2.2.3.1. Collaborative filtering

Collaborative filtering is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behaviours of other users in the system (Ekstrand et al., 2010). Collaborative filtering methods build a preference model by collecting and analysing data on the user's past behaviour and preferences and predict what the user will like based on similar decisions made by other users (Pazzani, 1999).

The assumption made by this algorithm is that if users agree on the quality or relevance of an item, they most probably will agree on other things. If some users like the same movies as Jane does, it is likely that Jane will like a movie these users like even if she has not seen it yet. Amazon is a great example of this type of system. When a user purchases an item, Amazon recommends to her similar products using such statements such as "Customers who bought this item also bought ..." and displays those items.

Collaborative filtering can focus either on users or items. In user-based collaborative filtering, a group of users with similar behaviour to that of the current user regarding past ratings is identified, and their scores are used to predict what might interest the current user (Sharma & Singh, 2016). In item-based collaborative filtering, the set of items the user has rated is considered, and the algorithm

computes how similar they are to item i and then selects the k most similar items $\{i_1, i_2, \dots, i_k\}$ (Karypis et al., 2001).

2.2.3.2. *Content-based filtering*

Content-based filtering approaches make recommendations by analysing the description of the items rated by the users and the description of the items to be recommended (Pazzani, 1999). In content-based filtering, keywords are used to describe the items, and a user profile is built to indicate things the user may like. The similarity of products is calculated based on the features associated with the compared items (Ricci et al. 2011). If Mary likes watching comedies or movies featuring Denzel Washington, it may be reasonable to recommend to her other comedies or movies starring Denzel Washington.

2.2.3.3. *Knowledge-based systems*

Knowledge-based recommender systems use the knowledge about users and products to pursue a knowledge-based approach to generating a recommendation, reasoning about what products meet the users' requirements (Burke, 2000). Case-based and constraint-based recommenders are two main types of knowledge-based recommenders. Case-based recommender systems use a similarity function to estimate how much user requirements (problem description) match the recommendation (solution to the problem) (Ricci et al., 2011). The similarity score represents the utility of the recommendation to the user.

Case-based and constraint-based recommenders are similar concerning the knowledge they use and their functionality. The systems collect user requirements, make recommendations based on the knowledge of how well they meet the requirements, repair inconsistencies in situations where no solutions were available, and provide explanations for recommendation results (Ricci et al., 2011).

The significant difference between the two types concerns the calculation of solutions: case-based recommenders make recommendations based on similarity

metrics whereas constraint-based recommenders exploit a predefined *knowledge base* that contains explicit rules on how to relate user requirements to product features (Felfernig et al., 2011).

2.2.3.4. *Demographic-based systems*

Demographic information may be used to identify the type of users that like a particular object. *LifeStyle Finder* classifies users in 62 predefined clusters and makes recommendations to a particular user based on the group he belongs to (Pazzani, 1999). Demographic-based recommendations can also be made using the region/country of the user. Netflix offers movies based on the area in which the user lives (Sharma & Singh, 2016). Therefore, only movies available in the user's location can be part of a recommendation.

2.2.3.5. *Community-based systems*

Community-based recommender systems claim that people are more likely to rely on recommendations from friends rather than from other similar but unknown individuals. Thus, the approach makes recommendations based on the preferences of the user's friends (Ricci et al., 2011). Community-based systems differ from collaborative filtering in that the former relies on similarity with friends while the latter is based on similarity with any users. The approach uses the social network of the user's friends and their ratings to make its recommendations.

2.2.3.6. *Hybrid recommender systems*

Hybrid recommender systems combine two or more approaches in an attempt to remedy shortcomings of one using the advantages of the other. For instance, collaborative filtering methods cannot recommend a new item with no ratings. However, the content-based approach does not face this problem since its recommendations are based on the description of the article (Ricci et al., 2011). Therefore, a combination of the two approaches can help in producing a recommendation where one approach would not have been able to produce one.

Netflix uses a mix of collaborative and content-based approaches (Sharma & Singh, 2016). It recommends movies taking into account the user's preference and the similarity with other users.

2.3. Critiquing-based recommender systems

Traditional recommendation approaches, collaborative and content-based filtering, are not well-suited in situations of high-value items such as vehicles, electronics and real-estate assets, which are not purchased as frequently as other items. Since people do not buy these types of products regularly, they do not express their opinions on them often. Therefore, it is infeasible to collect many ratings on such items, and potential customers may not be satisfied with years-old preferences expressed on such items (Felfernig et al., 2011).

Knowledge-based recommender systems can be used to overcome these challenges by exploiting explicit user requirements and knowledge underlying the product domain for the calculation of the recommendations. Furthermore, knowledge-based recommenders do not have cold-start problems since users state their preferences during a recommendation session. However, knowledge-based recommenders suffer from knowledge-acquisition bottlenecks associated with the initial efforts required to generate the domain knowledge (Felfernig et al., 2011).

Critiquing-based recommender systems have emerged and have been broadly recognised as an efficient preference-based search and recommender technology, using a feedback mechanism called *critiquing* (Chen & Pu, 2012). These systems make recommendations based on the current user preferences and then elicit user feedback in the form of critiques such as "*I would like a similar apartment with lower rent.*" This requirement elicitation cycle continues until the customer can settle on a preferred product. A typical customer has many preferences and constraints that are not stated up front, and the process enables him to be aware of these latent preferences when proposed solutions violate them (Pu & Faltings, 2000).

Critiquing-based recommender systems follow a four-step user-system interaction model (Chen & Pu, 2012):

- Step 1: the user is asked to provide some preferences on product features;
- Step 2: the system returns one or more recommended items based on the user's initial preferences;
- Step 3: the user either selects the desired item and terminates the process or provides feedback on the presented items (*critiquing*);
- Step 4: once the user provides the critiques, the system will update its recommendations and return them for the next interaction cycle.

This user-system interaction model is illustrated in Figure 2.1.

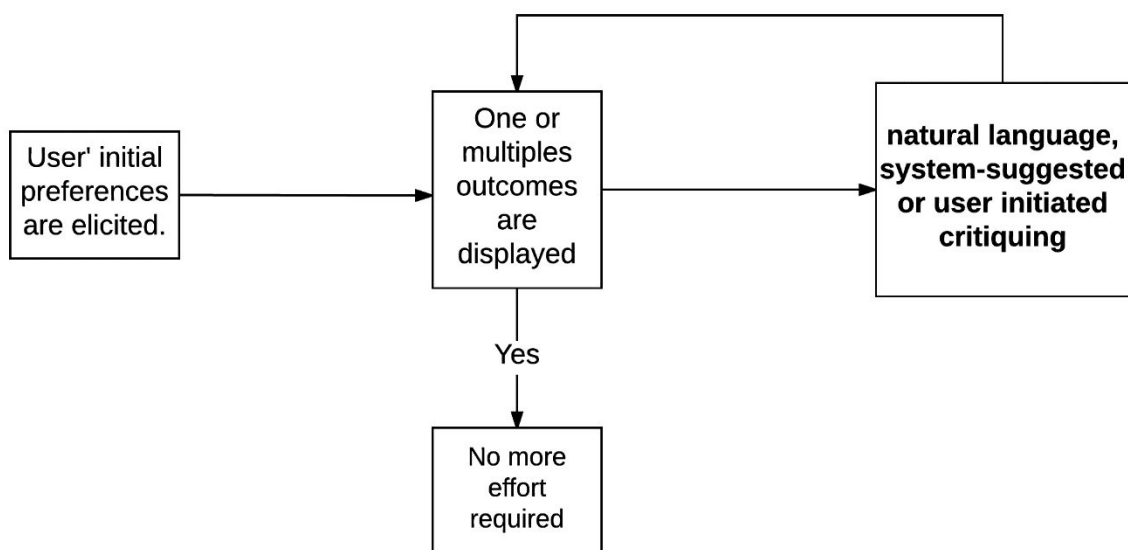


Figure 2.1: The typical interaction between users and critiquing-based recommender systems (Adapted from Chen & Pu (2012))

Chen and Pu (2012) conducted a comprehension survey on various critiquing-based approaches proposed by different scholars. The study identified three main types of critiquing-based recommender systems: natural language dialogue-based recommender systems, system-suggested critiquing systems and user-initiated critiquing systems. The following paragraphs briefly describe these approaches.

2.3.1. Natural language dialogue-based systems

Natural language dialogue-based recommender systems act as an artificial

salesperson and interact with the customer through a dialogue interface. *ExpertClerk*, a system that imitates a human sales clerk, and *Adaptive Place Advisor*, a system that provides personalised place recommendations, are two examples of such recommenders. Chen and Pu (2012) argue that these natural language dialogue-based recommender systems are suitable for recommendations delivered by speech (e.g. when the user is driving) but are not ideal for e-commerce environments where users are shopping online.

2.3.2. *System-suggested critiquing systems*

System-suggested critiquing systems proactively generate a set of knowledge-based critiques that the user may accept as a way of improving suggestions. *FindMe* uses its *tweaking* feature to enable the user to critique the current recommendation by allowing users to select pre-defined tweaks such as “bigger”, “cheaper”, etc. These systems are not able to adjust to user’s changing needs and only allow critiquing on a single attribute (*unit critiquing*) (Chen & Pu, 2012). In response to these challenges, other approaches have been proposed.

Dynamic critiquing enables critiquing on multiple attributes (*dynamic compound critiques*) and uses the association rule mining to discover different sets of value differences between the current recommendation and the remaining un-recommended items (Chen & Pu, 2012).

The MAUT-based compound critiques proposed by Zhang and Pu (2006) aimed at curing a challenge faced by dynamic critiquing, namely that the latter does not take into account the users’ interests in the suggested critiques. The Multi-Attribute Utility Theory (MAUT) takes into account the conflicting value preferences and produces a score for each item to represent its overall satisfaction degree with the user preferences, (Chen & Pu, 2012).

Preference-based organisation interface sought to address a disadvantage faced by the MAUT-based approach, in that each MAUT-based compound critique

corresponds to one product only and that not many recommendations can be displayed to the users (Chen & Pu, 2006; Pu & Chen, 2007c). It does so by generating critiques adaptive to users' MAUT-based preference model and applying the association mining rule to discover compound critiques that can be used to represent the remaining datasets. Then, it diversifies critiques and their contained products to assist users to refine and accumulate their preferences more efficiently (Chen & Pu, 2012).

2.3.3. User-initiated critiquing systems

The user-initiated critiquing systems show examples to users and stimulate the users to make self-motivated critiques. These systems allow users to make both unit and compound critiques over any combination of features in freedom. The aim of these systems is to enable users to execute trade-off navigation (Pu & Chen, 2005), that is finding a product that has more optimal values on important attributes while accepting compromised values on less important attributes (Chen & Pu, 2012).

Example-critiquing combines a preference-based search tool and example-critiquing capabilities (Pu et al. (2008). In such a system, the user starts the search by specifying a few preferences in the query area; each preference is composed of an acceptable attribute value and its corresponding importance (i.e.: weight); and the system builds an MAUT-based preference model (Chen & Pu, 2012). From the initial preference model, the search engine ranks alternatives by their corresponding scores and returns k top ones (Chen & Pu, 2012). The ideal value of k should range between 5 and 20 (Faltings, Pu & Torrens, 2003). If the user is satisfied with an item, he picks it; otherwise, he triggers the trade-off navigation to refine his preference.

2.4. Recommending rental properties

Having considered a variety of critiquing-based approaches to recommending items to consumers online, this study will adopt the *preference-based search* tool and will combine it with the *example-critiquing* approach. Preference-based search is a tool for the elicitation of users' initial preferences, whereas example-critiquing is an

approach that enables users to refine their preferences to locate the ideal item that suits their requirements (Viappiani et al., 2006).

This method invites users to state their preferences (preference-based search) explicitly. Viappiani et al. (2006) formally defined the problem as:

Given a collection $O = \{o_1, \dots, o_n\}$ of n options, preference-based search (PBS) is an interactive process that helps users find the most preferred option, called the *target* option o_t based on a set of preferences that they have stated on attributes of the target.

They define the target option as the option most preferred by the user among all the possibilities.

Once the user has expressed his preferences and the preference model has been developed, the system can then generate and display a set of examples for the user to consider. These examples include *candidates*, items that are optimal for the current preference query, and *suggestions*, items that are used to stimulate the expression of further preferences (Viappiani et al., 2006). The user revises his preference model by critiquing examples, a process that can take several iterations. When the user locates the target item, he terminates the process. This process is illustrated in Figure 2.2.

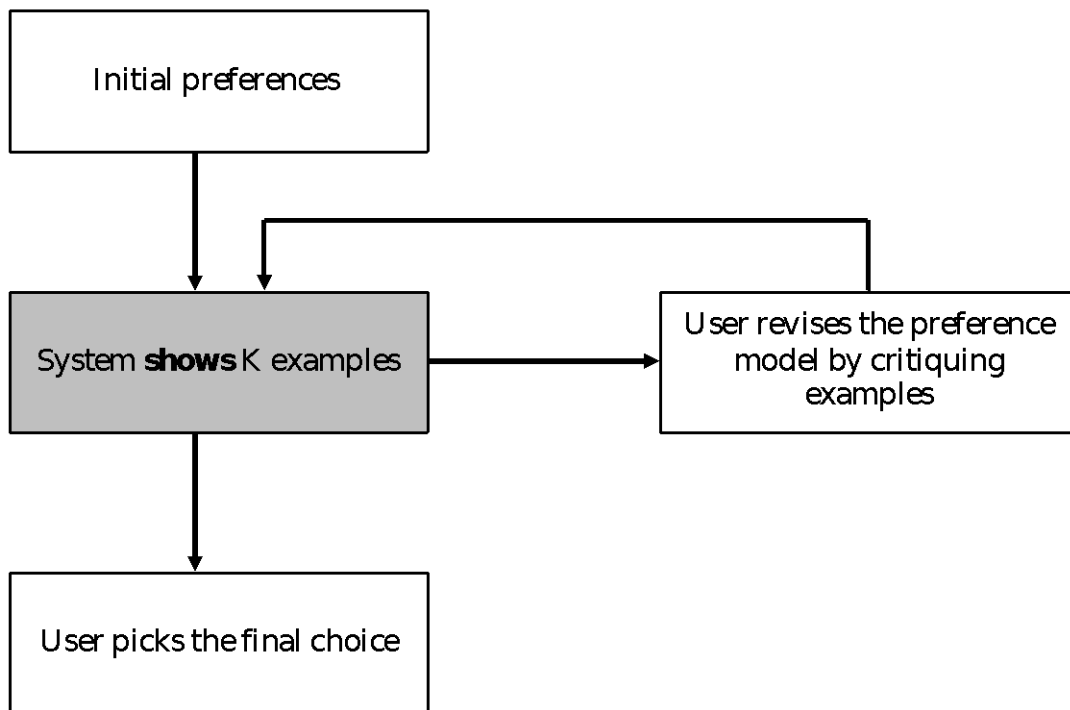


Figure 2.2: Example-critiquing interaction (Adapted from Viappiani et al. (2006))

Viappiani et al. (2006) proposed effective strategies for generating suggestions based on the current preference model. These strategies assume that the user will minimise his efforts and will only add preferences to his model only when he thinks they will impact the solution. Updating the preference model is only useful in cases when:

- (i) the user can see several options that differ in a possible preference,
- (ii) these choices are relevant, i.e. they could be acceptable options, and
- (iii) they are not already optimal for the already stated preferences.

Indicating additional preferences in all other cases is irrelevant. That is when all options would evaluate the same way, or when the preference only has an effect on choices that would not be eligible regardless or that are already the best choices (Viappiani et al., 2006). They further argue that upon displaying a suggested outcome whose optimality becomes clear if a particular preference is stated, the user can recognise the importance of stating that preference. Consequently, they developed the *look-ahead* principle, according to which suggestions should not be optimal under the current preference model, but should provide a high likelihood of

optimality when a user adds new preferences to the model.

2.4.1. User preference elicitation using preference-based search

The user expresses her preferences through a search engine on five attributes of rental properties: type, location, rent, the number of bedrooms, and the number of bathrooms. The researcher settled on these attributes after conducting an online survey of websites that advertise rental properties in Kenya and identifying the most common attributes used in searching rental properties. The user starts the search by specifying one or more preferences in a search interface. Furthermore, she indicates the importance (i.e.: weight) of each preference. Based on the first preference, the system will identify and display a set of matching results (Chen & Pu, 2007).

2.4.2. System recommendation based on user initial preference

To generate recommendations for a particular user, we must first define his preference model. The user's preference on all rental properties is represented as a weighted additive form of value functions based on the multi-attribute utility theory (MAUT), according to Chen and Pu (2007). They formally define a preference model as the pair $(\{V_1, \dots, V_n\}, \{w_1, \dots, w_n\})$ where V_i is the value function for each attribute A_i , and w_i is the relative importance (i.e.: weight) of A_i . The utility of each item $(\langle a_1, a_2, \dots, a_n \rangle)$ can then be computed using the utility function as shown in Equation 2.1.

$$U(\langle a_1, a_2, \dots, a_n \rangle) = \sum_{i=1}^n w_i V_i(a_i)$$

Equation 2.1: Utility function (Adapted from Chen and Pu (2007))

2.4.3. Capturing user feedback using example-critiquing

Chen and Pu (2012) argue that most users searching for information are not familiar with the set of available items and their characteristics. Example-critiquing can then

be used to enable these users to construct their preference models as they learn about possibilities progressively (Chen & Pu, 2012). Various scholars have used example-critiquing in systems with preference models and those without preference models.

In systems without preference models, the user tweaks the current best example to make it fit his preferences. A prospective tenant may say “I would like a similar apartment but cheaper.” Examples of such systems include *FindMe*, *ExpertClerk* and the dynamic critiquing systems (Chen & Pu, 2012). In systems with explicit preference models, each critique is added to the preference model to refine the query. Some of these systems are ATA system, *SmartClient* and incremental critiquing systems (Chen & Pu, 2012).

These systems with example-critiquing and an explicit user preference model have the advantage of resolving users’ preference conflicts, according to Chen and Pu (2012). This study will adopt this approach.

In example-critiquing, each critique can be considered as a soft constraint, and the preference model can be developed by simply collecting critiques, according to Chen and Pu (2012). They define a soft constraint as a function of an attribute or a combination of attributes to a number that indicates the degree to which the constraint is violated. When the value of an attribute violates the constraint, it is mapped to 1, otherwise it is assigned to 0.

For example, a prospective tenant may be willing to pay a monthly rent of KES 50,000 and may tolerate the violation of this constraint up to KES 5,000. This preference can be expressed using a piecewise function as shown in Equation 2.2. (x represents the rent amount):

$$\begin{array}{ll}
1 & \text{if } x > 55,000 \\
0.2(x - 50,000) & \text{if } 50,000 \leq x \leq 55,000 \\
0 & \text{if } x < 50,000
\end{array}$$

Equation 2.2: Piecewise function for one preference expressed on one attribute
(Adapted from Chen and Pu (2012))

The user may wish to express several preferences on the same attribute. For example, a prospective tenant may indicate that she needs an apartment with monthly rent ranging from KES 30,000 to KES 50,000 and that she is willing to tolerate a KES 5,000 violation on this preference. The resulting piecewise function for this preference can take the form shown in equation 2.3.:

$$\begin{array}{ll}
1 & \text{if } x > 55,000 \\
0.2(x - 50,000) & \text{if } 50,000 \leq x \leq 55,000 \\
0 & \text{if } 30,000 \leq x \leq 50,000 \\
0.2(30,000 - x) & \text{if } 25,000 \leq x \leq 30,000 \\
1 & \text{if } x < 25,000
\end{array}$$

Equation 2.3: Piecewise function for multiple preferences expressed on the same attribute
(Adapted from Chen and Pu (2012))

All user preferences are seldom satisfied at the same time. Therefore, users are required to make trade-offs: that is accepting an outcome that is undesirable in some respects while advantageous in others, (Pu & Faltings, 2004). They have identified three main strategies employed by users to make trade-offs:

- (i) *value* trade-off: the user changes the preference value of a particular attribute value combination;
- (ii) *utility* trade-off: the user changes the weight of preference in the combined ranking;
- (iii) *outcome* trade-off: the user adds additional preferences that increase the utility of an outcome they want.

When soft constraints model preferences, these trade-off strategies can be implemented by either revising the current set of soft constraints, or adding or retracting constraints, posit Pu and Faltings (2004). They argue that to enable this

type of user-system interaction, the system should be able to support the following actions:

- (i) revising user preferences;
- (ii) providing trade-off scenario to resolve user preference conflicts;
- (iii) revise the importance (weight) attached the preference; and
- (iv) elicit hidden preferences.

Once a user provides critiques, the recommendations can then be updated based on these critiques.

2.4.4. Building a preference model for a user

The set of objects on which users can express preferences is the collection of options $O = \{o_1, \dots, o_n\}$ with a fixed set of k attributes $A = \{A_1, \dots, A_k\}$, associated with domains D_1, \dots, D_n . Each option is characterised by the values $a_1(o), \dots, a_k(o)$, where $a_i(o)$ represents the value that o takes on attributes A_i (Viappiani et al., 2006).

A preference r is an order relation \preceq_r of the values of an attribute a ; and \sim_r means that two values are equally preferred. Therefore, according to Viappiani et al. (2006), a preference model R is a set of preferences $\{r_1, \dots, r_m\}$. They argue that preferences are assumed to be independent and expressed on individual attributes. Furthermore, they assert that since a preference r_i always applies to the same feature a_i , the notation can be simplified to use \preceq_{r_i} and \sim_{r_i} directly to the options: $o_1 \preceq_{r_i} o_2$ iff $a_i(o_1) \preceq_{r_i} a_i(o_2)$. The use of \preceq_{r_i} indicates that \preceq_{r_i} stands but not \sim_{r_i} .

Any rational decision-maker will prefer an option to another if the first is at least as good in all criteria and better for at least one criterion, argue Viappiani et al. (2006). They further assert that Pareto-dominance, which is a partial order relation of the options, can express this concept. An option o is Pareto-dominated by an option o' on R if and only if for all $r_i \in R$, $o \preceq_{r_i} o'$ and for at least one $r_j \in R$, $o \preceq_{r_j} o'$ (Viappiani, Faltings, & Pu, 2006). The notations $o \preceq_R o'$ and $o' \succ_R o$ mean that o is

Pareto-dominated by o' .

Viappiani et al. (2006) make an important assumption about preference combination functions: they must be dominance-preserving. A preference combination function is dominance-preserving if and only if whenever an option o' dominates another option o in all individual orders, then o' dominates o in the combined order.

2.4.5. Making model-based suggestions

Strategies proposed by Viappiani et al. (2006) on making model-based suggestions are based on the principle of choosing options that are most likely to become optimal. This is done by considering new preferences and describing the probability that they make a choice optimal. Viappiani et al. (2006) present two qualitative notions of optimality, one based on Pareto-optimality and another based on the combination function used to generate candidates. These concepts are discussed before describing the strategies used to produce model-based suggestions.

The first notion is Pareto-optimality. An option is Pareto-optimal if and only if another option does not dominate it. Pareto-optimality applies to any preference model as long as the combination function is dominance-preserving. For any dominance-preserving combination function, an option o^* that is most preferred in the combined preference order is Pareto-optimal, since any option o' that dominates it would be more preferred (Viappiani et al., 2006).

The second notion concerns a *dominating set* and an *equal set*. A dominating set of an option o on a set of preferences R is a set of all the options that dominate o : $O_R^>(o) = \{o' \in O : o' >_R o\}$. One can omit R if it is clear in the context and just write $O^>(o)$. An *equal set* of an option o on a set of preferences R is a set all of all the options that are equally preferred to o : $O_R^=(o) = \{o' \in O : o' \sim_R o\}$. One can use O_R^{\geq} for $O^{\geq} \cup O^=$ (Viappiani et al., 2006).

When a user states a new preference r_i , a dominated option can become

Pareto-optimal. Viappiani et al. (2006) argue that a dominated option o on a set of preferences R becomes Pareto-optimal on $R \cup r_i$ if and only if o is strictly better with respect to r_i than all options that dominate it on R ; and not worse with respect to r_i than all options that are equally preferred on R .

The model-based suggestion strategies proposed by Viappiani et al. (2006), the counting strategy and the probabilistic strategy, are based on the look-ahead principle according to which suggestions should not be optimal under the current preference model but have a high likelihood of becoming optimal when a user adds a new preference. The assumption made here is that the system is aware of a subset R of the user's preference model \bar{R} . The best suggestion is the option o that is dominated in the current (partial) preference model R but that is ultimately optimal with respect to the full preference model \bar{R} (Viappiani et al., 2006).

This study will adopt the probabilistic strategy as, according to Viappiani et al. (2006), it provides a more accurate estimate of the likelihood that a particular choice will become Pareto-optimal.

Probabilistic strategy

General assumptions

The first assumption made in this strategy is that a cost function c_i expresses each preference r_i . For the purpose of having a well-designed interface, we must restrict these functions to a family of functions parameterised by one or more parameters. Such a function with one parameter take the form shown in equation 2.4.

$$c_i = c_i(\theta, a_i(o)) = c_i(\theta, o)$$

Equation 2.4: Cost function (Adapted from Viappiani et al. (2006))

The second assumption is that the following probability distributions express possible preferences: p_{ai} , the likelihood that the user has a preference over an attribute a_i ; and $p(\theta)$, the probability distribution of the parameter associated with the cost function of the considered feature.

To calculate the likelihood that a preference on attribute i makes o_1 be preferred to o_2 , we integrate over the values of θ for which the cost of o_1 is less than that of o_2 .

We use the Heaviside step function: $H(x) \equiv \mathbf{if} (x > 0) \mathbf{then} 1 \mathbf{else} 0$:

$$\delta_i(o_1, o_2) = \int_{\theta} H(c_i(\theta, o_2) - c_i(\theta, o_1)) p(\theta) d\theta$$

For qualitative domains, we compute by iterating over θ and summing up the probability contribution for the cases in which the value of θ makes o_1 preferred to o_2 :

$$\delta_i(o_1, o_2) = \sum_{\theta \in D_i} H(c_i(\theta, o_2) - c_i(\theta, o_1)) p(\theta)$$

For the purpose of determining the likelihood of breaking the dominance relation with all options in the dominating set through a_i , all dominating options must have a less preferred value for a_i than that of the considered value. This approach for determining the probability of breaking the dominance relation does not assume independence between options and directly examines the distribution of all the dominating options (Viappiani et al., 2006).

For numerical domains, we computed that probability as follows:

$$\delta_i(o, O^>) = \int \left[\prod_{o' \in O^>} H(c_i(\theta, o') - c_i(\theta, o)) \right] p(\theta) d\theta$$

For qualitative domains, the integral is replaced by the summation over θ and adding a new term into the integral to account for the fact the no new dominance

relations with options in the equal set should be created.

$$\delta_i(o, O^{\geq}) = \int \left[\prod_{o' \in O^>} H(c_i(\theta, o') - c_i(\theta, o)) \prod_{o'' \in O^=} H^*(c_i(\theta, o'') - c_i(\theta, o)) \right] p(\theta) d\theta$$

Equation 2.5: Computing probability of breaking the dominance in qualitative domains (Adapted from Viappiani et al. (2006))

where H^* is a modified Heaviside function that assigns value 1 to whenever the difference between two costs is 0 or greater ($H^* \equiv \text{if } (x \geq 0) \text{ then } 1 \text{ else } 0$).

We assume that the user only has one hidden preference. We consider the probability of becoming Pareto-optimal when a preference is added as the combination of event that the new preference is for a particular attribute, and the chance that the preference on this feature will make the option be preferred over all values of the dominating options (Viappiani et al., 2006):

$$F_p(o) = \sum_{a_i \in A_u} P_{a_i} \delta_i(o, O^{\geq})$$

Preference functions and suggestion computation

Preference for a single value in the qualitative domain

These functions will be used to compute suggestions on qualitative attributes of a rental property namely location and type. Let θ be the value preferred by a prospective tenant; the function $c_i(\theta, x)$ gives a penalty to any value of attribute a_i except θ . For instance, a potential tenant may say: "I would like an apartment in Kileleshwa", meaning that he prefers apartments in this neighbourhood to those in other neighbourhoods.

$$c_i(\theta, x) \equiv \text{if } a_i(x) = \theta \text{ then } 0 \text{ else } 1.$$

The likelihood of breaking a dominance relation between option o_1 and o_2 is the probability that the value of option o_1 for attribute i is preferred when it differs from the value of o_2 .

$$\delta_i(o_1, o_2) = \begin{cases} p[\theta = a_i(o_1)] & \text{if } a_i(o_1) \neq a_i(o_2) \\ 0 & \text{otherwise} \end{cases}$$

Consequently, assuming a uniform distribution, $p(\theta) = \frac{1}{|D_i|}$ for any θ (any value in the domain is equally likely to be the preferred value), the probability becomes $\frac{1}{|D_i|}$ when $a_i(o_1) \neq a_i(o_2)$ otherwise, it is equal to 0.

$$\delta_i(o_1, o_2) = \begin{cases} 1/|D_i| & \text{if } (\forall o' \in O^>) a_i(o_1) \neq a_i(o_2) \\ 0 & \text{otherwise} \end{cases}$$

Equation 2.6: Computing preference for one value in qualitative domain (Adapted from Viappiani et al. (2006))

Note that in this structure of preference, $\delta_i(o, O^>) = \delta_i(o, O^>)$, because an option o can only break the dominance relation only if $a_i(o)$ takes the preferred value, any other option can be strictly better on that preference.

Directional preferences

These functions will be used to compute suggestions in numeric domains when the preference order can be assumed to have a direction. They apply to attributes of a rental property such as rent, the number of bedrooms, and the number bathrooms. Regarding rent, lower is always preferred other things being constant; and concerning the number of bedrooms and of bathrooms, more is always preferred when other things are held constant. These functions are shown in equation 2.7.

$$\delta_i(o_1, o_2) \begin{cases} \text{if } a_i(o_1) < a_i(o_2) \text{ then } 1 \text{ else } 0 & a_i \text{ numeric, natural preference } < \\ \text{if } a_i(o_1) > a_i(o_2) \text{ then } 1 \text{ else } 0 & a_i \text{ numeric, natural preference } > \end{cases}$$

Equation 2.7: Computing directional preferences (Adapted from Viappiani et al. (2006))

In directional preference, the cost function is a monotone function of the attribute value.

When a step function represents the preference $LessThan(\theta)$, an option is preferred over a set of options with minimum value l_i if the preference value θ falls in

between the values of the given option and l_i . For a set of options O^{\geq} whose values on a_i lies between l_i and h_i , we have

$$\delta_i(o, O^{\geq}) \begin{cases} 1 & \text{if } a_i(o) < l_i \\ 0 & \text{otherwise} \end{cases}$$

Equation 2.8: Step function for preference when smaller values are preferred
(Adapted from Viappiani et al. (2006))

when smaller values are preferred, and

$$\delta_i(o, O^{\geq}) \begin{cases} 1 & \text{if } a_i(o) > h_i \\ 0 & \text{otherwise} \end{cases}$$

Equation 2.9: Step Function for Preference when Larger Values Are Preferred (Adapted from Viappiani et al. (2006))

when larger values are preferred.

Generating a set of suggestions

The strategies we described are used to produce only one suggestion. However, it is possible to provide a set I of suggestions simultaneously. In doing so, we should choose a group G of suggested options by maximising the probability $p_{opt}(G)$ that at least one of the suggestions in the set G become optimal through a new user preference (Viappiani et al., 2006):

$$p_{opt}(G) = 1 - \prod_{a_i \in A_u} (1 - P_{a_i} (1 - \prod_{o' \in G} (1 - \delta_i(o', O^{\geq}(o')))))$$

Equation 2.10: Generating multiple suggestions (Adapted from Viappiani et al. (2006))

2.5. Algorithm for generating suggestions

Algorithm: Dominance-Check (o_1, o_2, R)

The dominance-check between two options o_1 and o_2 with respect to preferences in R .

Legend: $<$ dominated, $>$ dominates, \approx not comparable, \equiv equivalent

Dominance can take values in $\{<, >, \simeq, \equiv\}$

This algorithm is presented in Figure 2.3.

```
dominance := VOID;
for all  $r_i \in R$  do
  if ( $o_1 \succ_{r_i} o_2$ ) then
    if (dominance != "<") then
      dominance ← ">";
    else
      return "≈";
  if ( $o_1 \prec_{r_i} o_2$ ) then
    if (dominance != ">") then
      dominance ← "<";
    else
      return "≈";
  if ( $o_1 \sim_{r_i} o_2$ ) then
    if (dominance == "≡" OR dominance == "VOID") then
      dominance ← "≡";
return dominance;
```

Figure 2.3: Algorithm for checking dominance between two options (Adapted from Viappiani & Faltings (2006))

2.6. Conceptual framework

This conceptual framework summarises the process of eliciting user preference using preference-based search and refining them using example-critiquing until the user finds the items that best meets his needs. The conceptual framework is illustrated in Figure 2.4.

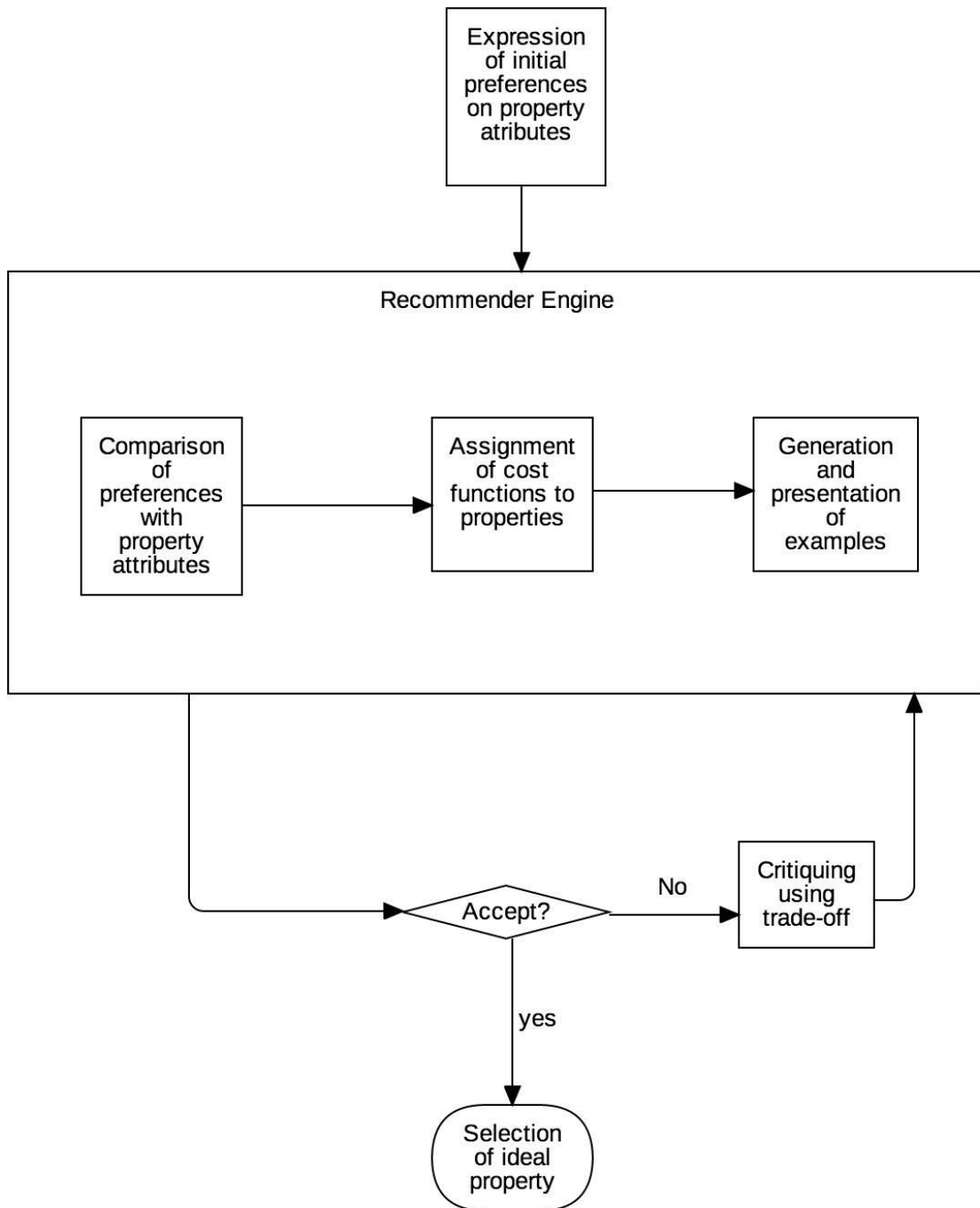


Figure 2.4: Conceptual framework

CHAPTER 3: RESEARCH METHODOLOGY

3.1. Introduction

Research is a careful, systematic investigation in some field of knowledge, undertaken to establish facts or principles or to find answers to a problem (Grinnell, 1993). Research methodology is the science of doing research (Bhatnagar & Singh, 2013). This chapter is intended to present how data was collected and analysed with the view of enabling readers to evaluate the validity and reliability of this study (University of Southern California, 2016).

3.2. System Development Methodology used

The *System Development Life Cycle* is the process of determining how an information system (IS) can support business needs, designing the system, building it, and deliver it to users (Dennis et al., 2012). This process contains four major phases namely system planning, analysis, design and implementation.

This research aimed at developing a prototype of a recommender system for rental property. It adopted the Object-Oriented Systems Analysis and Design (OOAD) approach to develop the prototype. While this method can use any of the traditional methodologies, the study employed the Rapid Application Development (RAD) methodology, an iterative development with which it is mostly associated (Dennis et al., 2012). The advantage of this methodology to application development over others is the reduced development cycle it offers. This aspect is essential to the success of the study considering the limited amount of time allocated to the research.

Traditional approaches to systems development tend to be either process-centric or data-centric. However when modelling real world processes and data, one soon realises that processes and data are closely intertwined. Decomposing processes and data is, therefore, a major challenge for these approaches. The OOAD uses an RAD-based sequence of System Development Life Cycle (SDLC) but attempts to balance between process and data (Dennis et al., 2012). This is achieved by focusing on decomposing problems in objects that contain both data and processes (Dennis et

al., 2012).

The process of developing a system using OOAD starts with the planning phase, then proceeds to the creation of use cases. From this stage, the first iteration comprising of the analysis, design and implementation phases starts. The *planning phase* is a fundamental process of understanding *why* an information system should be built and determining how the project team would go about building it (Dennis et al., 2012).

This chapter did not focus on system planning phase because many of its steps did not apply to this case, as this is an academic endeavour. These steps include identifying opportunity, analysing technical and economic feasibility, developing a work plan, identifying staff project and controlling and directing the project.

Any object-oriented approach must be (i) *use case driven*, (ii) *architecture-centric* and (iii) *iterative and incremental* (Dennis et al., 2012). According to (Dennis et al., 2012), use case driven means that the behaviour of the system is modelled through use cases; architecture-centric implies that the underlying architecture of the evolving system drives the specification, construction and documentation of the system; and iterative and incremental mean that each iteration brings the system closer and closer to the final requirements of the end-users.

3.2.1. *Requirements gathering*

Requirements determination is performed to transform high-level statement of business requirements into a more detailed, precise list of what the system must do to provide the needed value to users. A requirement simply refers to a statement of what the system must do or what characteristic it needs to have (Dennis et al., 2012). This section addressed the population, sampling, and data collection aspects of the study.

3.2.1.1.

Population

The end users of the proposed recommender system are prospective tenants searching for rental properties online. In order to identify criteria they used to searching properties online, the researcher reviewed websites operated by property management firms. Therefore, the population relevant to this study comprises of real-estate development firms and property management companies that advertise rental properties online. In September 2016, the Estate Agents Registration Board, a regulatory body for estate agents in Kenya, had 331 registered members (Estate Agents Registration Board, 2016). Real estate firms with functioning websites, selected from these real estate agents, therefore, formed the population of this study.

3.2.1.2. *Sampling*

When conducting research, one must determine the sample size whose responses he needs to understand the problem at hand. Research requires an understanding of the statistics that drive sample size decisions (Smith, 2013). A few concepts need to be addressed before calculating the sample size for this research.

Population size refers to how many respondents are of interest to the researcher. *Margin of error (confidence interval)* determines how higher or lower than the population mean is the researcher willing to let the sample mean fall (Smith, 2013). This is because no sample can perfectly represent the entire population.

Confidence level refers to how confident the researcher wants to be that the mean falls within the confidence interval. According to Smith (2013), the most common confidence intervals are 90% confident, 95% confident, and 99% confident. This research used a confidence level of 95% whose corresponding Z-score is 1.96. *Standard deviation* is a measure of the variance the researcher expects in the responses. Smith (2013) argues that the safe decision is to use 0.5 as this number ensures that the sample will be large enough.

The formula for sample size calculation is provided in Figure 3.1.

$$\text{Sample size} = \frac{\frac{z^2 \times p(1-p)}{e^2}}{1 + \left(\frac{z^2 \times p(1-p)}{e^2 N}\right)}$$

Equation 3.1: Formula for calculating a sample size (Adapted from (Smith, 2013))

Where N refers to population size; z refers to the z-score; e refers to the margin of error; and p refers to the standard deviation.

Removing duplicates (agents working for the same company) and removing agents who worked as individuals, the researcher identified 187 real estate companies registered by this Board. Among the 187, 85 had functioning websites. These formed the target population for the study. Using the formula for calculating a sample size; with a population size of 85, a confidence level of 95%, and margin of error of 5%, the researcher found that the appropriate sample size would be 70.

3.2.1.3. *Data collection*

The data of interest to the research is the criteria used by prospective tenants to select rental properties. This data was collected through content analysis by reviewing websites run by these agencies as they contain information relevant the criteria at issue. Content analysis is a research technique used to make replicable and valid inferences by interpreting and coding textual material. By systematically evaluating texts (e.g.: documents, oral communication and graphics) qualitative data can be converted into quantitative data (University of Georgia, Terry College of Business, 2012).

Parameters used for property search are a good indication of the criteria that property managers believe prospective tenants consider when they are selecting properties. Therefore, this data collected on websites was used to identify the most important criteria for prospective tenants when they search for rental properties. The

data informed the design and implementation of the prototype of the rental property recommender system.

3.2.2. *Systems analysis*

As indicated in earlier, systems analysis is the second of the four main phases of the System Development Life Cycle. The *analysis phase* answers the questions of *who* will use the system, *what* the system will do, and *where* and *when* it will be used (Dennis et al., 2012). The output of the system analysis phase is a system requirement document. The detailed analysis of the proposed system is provided in *Chapter Four*.

3.2.3. *Systems design*

Systems design is the third phase of the System Development Life Cycle. The *design phase* decides how the system will operate in terms of hardware, software, and network infrastructure that will be in place, the user interface, forms and reports that will be used; and the specific programs, databases and files that will be needed (Dennis et al., 2012). A detailed systems design is provided in the *Chapter Four*.

3.2.4. *Systems implementation*

Systems implementation is the fourth phase of the System Development Life Cycle. During the *implementation phase*, the system is actually built (or purchased in case of a software design and installed). The objective of this phase is to deliver a fully functioning and documented system (Dennis et al., 2012). The implementation of the system is described in more details in *Chapter Five*.

3.3. **Research design**

Research design refers to the arrangement of conditions for collection and analysis of data in a manner that aims to combine the relevance to the research purpose with economy in procedure (Selltitz, Jahoda, Deutsch, & Cook, 1967). Research design is the conceptual structure within which the research is conducted; it constitutes the blueprint for the collection, measurement and analysis of data (Kothari, 2004).

The type of research undertaken in this study is *applied research*. Applied research is aimed at finding a solution of an immediate problem facing society or an industrial/business organisation (Kothari, 2004; Cooper & Schindler, 2011). The justification for this choice of type of research is that this research attempts to address a perceived *real world* problem and to suggest a solution to it.

According to Kothari (2004), a good research design should have four main parts. These parts are described in the following paragraphs and an account of how this study addressed them is provided.

The *sampling design* deals with the method of selecting items to be observed in the study. The study selected property management firms to be observed from the list of registered real estate agents provided by the Real Estate Agents Board in Kenya. Specifically, websites belonging to these firms were observed.

The *observational design* relates to the conditions under which the observations are to be made. In this study, the websites were observed in the conditions in which they appear to users (that is in a web browser).

The *statistical design* concerns the question of how many items are to be observed and how the data collected is to be analysed. This study used a formula to compute the sample size. This formula is given in section 3.2.1.2 (Sampling). The data was analysed using Microsoft Excel.

The *operational design* deals with the techniques by which the procedures specified in the sampling, statistical and observational designs are to be carried out.

The researcher observed specific characteristics of the websites, namely the criteria used to select rental properties, and recorded them in a Microsoft Excel sheet. When a particular criterion was observed, the researcher indicated that with the letter T,

representing the value True, next to the Uniform Resource Locator (URL) of the website associated with the firm. When the criterion was not observed, the researcher recorded this with the letter F, representing the value False.

The researcher used the functionalities of Microsoft Excel to compute the total number of websites in which each criterion was observed. After that, the researcher used the same software to generate a graphical representation of these findings, which appear in *Chapter Four* of this study.

3.4. Research quality - validity, reliability and objectivity of the research

Validity is the property of a research instrument that measures its relevance, precision and accuracy (Sarantakos, 2005). It tells the researcher whether a tool measures what is supposed to measure and whether this measurement is accurate and precise. This study used *face validity* as a measure of validity. An instrument has face validity, 'on the face of it', if it measures what is expected to measure (Sarantakos, 2005). This study collected and analysed data on criteria used by prospective tenants to select rental properties. The study satisfied the requirement for face validity because the researcher actually examined the criteria as mentioned above while collecting and analysing data on them.

Reliability refers to the capacity of measurement to produce consistent results (Sarantakos, 2005). He argues that a method is reliable if it provides the same results whenever repeated, and is not sensitive to the researcher, the research conditions or the respondents. In this study, the researcher recorded whether or not a website mentioned a particular attribute of rental properties. This approach produces consistent results, unless the content of the websites changes over time.

Objectivity is the empiricist doctrine that the research process and design must be free of personal bias and prejudice (Sarantakos, 2005). It ensures that personal values and views of the investigator are kept out of the research process, argues Sarantakos

(2005). In this study, objectivity was measured through representativeness and generalizability.

Representativeness is a research principle that reflects the capacity of a social research to produce results that are consistent with (representative of) what is observable in the target population (Sarantakos, 2005). Generalizability is the ability of the research to extrapolate the relevance of its findings beyond the sample, which is the extent to which the study can generalise its findings to from the sample to the whole population (Sarantakos, 2005). To achieve representativeness, the researcher used a formula to compute a sample size that is representative of the entire population. Consequently, the result findings are representative of and generalizable to the entire population.

3.5. Ethical considerations

The researcher also maintained the confidentiality of any data collected on the population. Where required, the data was presented in aggregated form and without identifying specific property management firms that were considered in the study. Furthermore, the researcher used this data solely for purposes of the study. Finally, the researcher cited the work obtained from others authors and gave credit where it was due.

CHAPTER FOUR: SYSTEMS ANALYSIS AND DESIGN

The data collected on important criteria used to select rental properties online informed systems analysis and design for the proposed system. These are, in order words, attributes of rental properties that tenants considered in deciding which properties fit their requirements.

To identify these attributes, the researcher reviewed websites of 70 real estate agents registered by the Estate Agents Registration Board. Among the 70 websites, 53 contained relevant information about these attributes. Following a systematic review of information contained in those websites, the researcher identified the most common characteristics of rental properties that tenant found relevant.

Some of the attributes that were found to be common after the review of these websites include property location, property type, rent, number of bedrooms, number of bathrooms, the size of the living space, availability of parking space, whether or not the rental properties were furnished, and whether or not they are serviced. The researcher used the word *True* (T) to say that a particular website mentioned an attribute, and the word *False* (F) to mean that the website did not mention that attribute. Figure 4.1 presents the total number of websites that mention each of the above attributes as a criterion to select rental properties.

Figure 4.1: Number of websites that mention relevant attributes for selecting rental properties

As Figure 4.1. shows that property location, property type, rent, number of bedrooms, and number of bathrooms are mentioned in 52, 50, 52, 50, 33 websites respectively as attributes of rental properties relevant to online search of rental properties. Based on these findings, the researcher decided to use these attributes, found in most websites for search rental properties, to develop the prototype of the proposed recommender system.

As earlier described, the system development process based on Object-Oriented Systems Analysis and Design, starts with describing use cases, followed by several iterations comprising of the analysis, design and implementation phases of the System Development Life Cycle. The following paragraphs present the final outputs of these iterations.

4.2. Use cases

In OOAD, use cases are the primary modelling tools employed to define the behaviour of the systems (Dennis et al., 2012). They further argue that a *use case* describes how a user interacts with the system to accomplish certain tasks such as placing an order or searching for information. The study produced the required use

cases to model system behaviour.

4.2.1. Use case 1: “Add property”

This use case depicts the interactions of a staff with the system while he adds a new rental property to the database. The use case starts with staff navigating to the login page for authentication. Upon authentication, the system displays a new page with all the rental properties currently in the database. The system further provides a link to a page where the staff can add a new rental property. The staff specifies all the attributes of the rental property he wishes to add to the database before attempting to save it.

The use case may have two exceptions. The first is an authentication exception when the staff’s credentials are not valid. The second is the staff does not specify all attributes of a rental property. In both cases, the system displays appropriate error messages. This use case is described in details in Figure 4.2.

Use Case Name: Add new rental property		ID: UC-1	Priority: High
Actor: System administrator (Administrator)			
Description: The administrator adds a new rental property to the database by entering its details and saving it.			
Trigger: The administrator wants to add a new rental property to the database.			
Type: External			
Preconditions: <ol style="list-style-type: none"> 1. The website is available 2. The rental property catalogue is up-to-date and on-line. 			
Normal course:		Information for steps:	
1.0. Add a new rental property <ol style="list-style-type: none"> 1. The administrator navigates to the login page. 2. The system displays the login page. 3. The administrator provides his/her login credentials 4. The system authenticates the administrator 5. The system displays the all properties currently in the database and a successful login message. 6. The administrator navigates to the “Add New Property” page. 7. The administrator enters details of the new rental property 8. The administrator attempts to save the new rental property in the database. 9. The system administrator logs out. 		← Email and password of the website administrator ← Property details	
Exceptions E.1: Login credentials are not valid (occurs at step 4) <ol style="list-style-type: none"> 1. The system displays an error message. 2. The administrator is no logged in. 3. The system displays the login page. E.2: The administrator does not specify one or more property details <ol style="list-style-type: none"> 1. The system displays a message alerting the administrator of the missing attributes 2. The system does not save the new rental property. 3. The system displays the “Add New Property” page 			
Postconditions: <ol style="list-style-type: none"> 1. One rental property is added to the database. 			
Summary		Destination	
Inputs	Source	Outputs	Destination
Email password Property details	System administrator System administrator	New Property	Property Catalogue/Database

Figure 4.2: Use case 1 “Add property”

4.2.2. Use case 2: “Locate property”

This use case depicts the interactions of a prospective tenant (user) with the system while she locates a desired rental property. The use case starts with the user navigating to the landing page and where she is prompted by a search panel. She specifies at least one attribute of rental properties she is interested in and indicates its importance. The she clicks on the search button to initiate the search. The use case continues as the user refines her preferences, if necessary, and ends when she locates an ideal property. This use case is described in details in Figure 4.3.

Use Case Name: Locate rental property		ID: UC-2	Priority: High
Actor: Prospective Tenant (user)			
Description: The user locates rental properties by expressing his or her preferences on attributes of rental properties and indicating the importance such attributes.			
Trigger: The user needs a house to rent Type: External			
Preconditions: <ol style="list-style-type: none"> 1. The website is available 2. The rental property catalogue is up-to-date and on-line. 			
Normal course: 1.0. Locate rental property <ol style="list-style-type: none"> 1. The user navigates to the landing page. 2. The system displays the landing page. 3. The user provides the desired attributes of a rental property and indicates each attribute's importance 4. The user searches for properties matching his or her preferences 5. The system generates examples and presents them to the user 6. The user selects the most preferred rental property. 		Information for steps: 	
Alternatives Courses : 1. User refines his or her preferences on rental property after reviewing presented examples (branch at step 5) <ol style="list-style-type: none"> 1. The user critiques the presented examples by making trade-offs on attributes 2. The user search for properties that matches the new preferences 3. The system generates a new set of examples and presents them to the user 4. The user selects the most preferred rental property. 			
Postconditions: <ol style="list-style-type: none"> 1. One rental property is selected by the user. 			
Summary			
Inputs	Source	Outputs	Destination
Property attributes Importance of attributes	Property catalogue User	List of properties Selected property	User User

Figure 4.3: Use case 2 “Locate property”

4.2.3. Use case 3: “Delete property”

This use case depicts the interactions of a staff member with the system while he deletes a rental property from the database. The use case starts with the staff navigating to the login page for authentication. Upon authentication, the system displays the list of all rental properties currently in the database. The staff selects any property he wants to delete and navigates to new page displaying only the property at issue. Then he clicks on a “Delete” button to delete it before being rerouted to the list of all properties again. This use case is described in details in Figure 4.4.

Use Case Name: Delete an existing property		ID: UC-3	Priority: Low
Actor: System administrator (administrator)			
Description: The administrator deletes an existing rental property from the database.			
Trigger: The administrator wants to delete an existing rental property from the database.			
Type: External			
Preconditions: <ul style="list-style-type: none"> 1. The website is available 2. The rental property catalogue is up-to-date and on-line. 			
Normal course: <ol style="list-style-type: none"> 1.0. Delete an existing rental property <ol style="list-style-type: none"> 1. The administrator navigates to the login page. 2. The system displays the login page. 3. The administrator provides his/her login credentials (Alternative course 1.1.) 4. The system authenticates the website administrator 5. The system displays the all properties currently in the database page and a successful login message. 6. The administrator enters clicks on a rental property he wants to delete 7. The system displays the said property on a new page. 8. The administrator clicks the "Delete" button deletes the rental property from the database. 9. The website administrator logs out. 		Information for steps: <p>← Email and password of the website administrator</p>	
Alternatives Course 1.1. : <ol style="list-style-type: none"> 1.1. The login credentials of the website administrator are not valid. <ol style="list-style-type: none"> 1. The system displays an error message. 2. The system displays the login page. 			
Postconditions: <ol style="list-style-type: none"> 1. One rental property is deleted from the database. 			
Summary			
Inputs	Source	Outputs	Destination
Email and password of the website administrator	Website administrator		

Figure 4.4: Use case 3 "Delete property"

4.2.4. Use case diagram

The use case diagram provides a visual summary of the various interactions that different actors have with the system while attempting to accomplish various tasks. A staff member performs three main tasks: adding new properties to the database, editing and removing some properties from the database. He can also perform tasks performed by regular users including searching, viewing and selecting properties. The users of the system perform one main task: locating properties that meet their preferences. This action includes searching, viewing and selecting properties. The use case diagram appears in Figure 4.5.

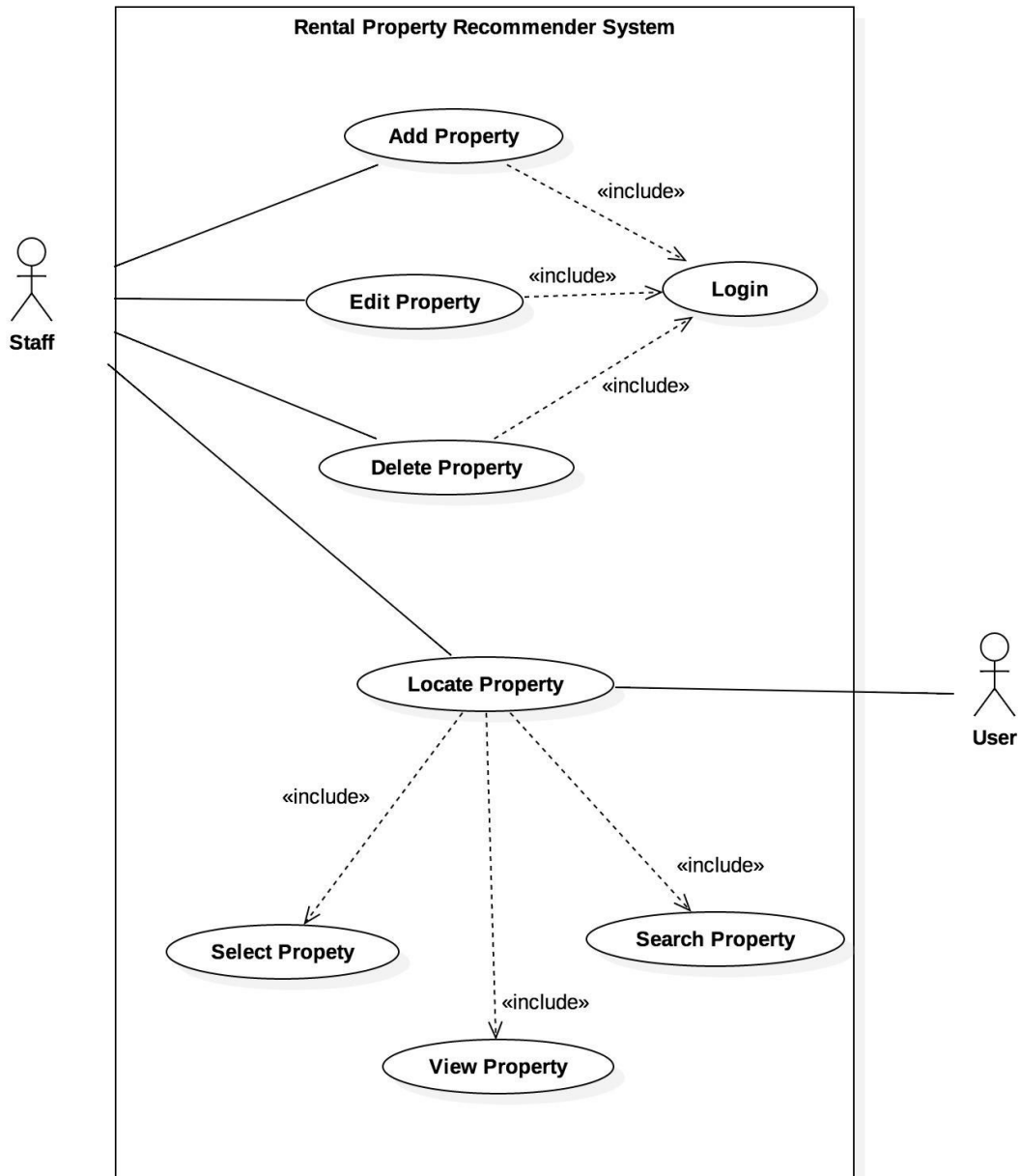


Figure 4.5: Use case diagram for the proposed system

4.3. Object-oriented analysis

This phase addresses the questions of *who* will use the system, *what* the system will do, and *where* and *how* it will be utilised (Dennis et al., 2012). The object-oriented analysis is concerned with determining system requirements and identifying classes and relationships among these classes. To understand system requirements, we must identify the users or actors in the system and know how they use it.

4.3.1. System requirements

The main functional requirements of the prototype of a recommender system for rental properties, as proposed in this study, include the following:

4.3.1.1. Ability to run on a web browser

The proposed system can run on all popular web browsers including Google Chrome, Internet Explorer, Mozilla Firefox, and Safari.

4.3.1.2. Ability to perform a preference-based search (PBS)

The system displays a search panel in which users can perform preference-based search by entering attributes of rental properties they consider relevant and indicating how important those attributes are to their search.

4.3.1.3. Ability to display search results

Once a user has specified her preferences and clicked on the search button, the system displays a set of rental properties that best match her preferences.

4.3.1.4. Ability to select a preferred rental property

Once, a prospective tenant has located a preferred rental property; the system provides him with the functionality to select this property.

4.3.1.5. Ability to add rental property to the system, edit them and remove them

The proposed system provides a staff member with the functionality to add rental properties to the database, edit their details, and remove them from the database.

4.3.1.6. *Ability to register and log in a staff member*

The proposed system also provides the functionality for authenticating a staff member. This is the only user who requires authentication to perform his tasks. In doing so, the system enables the staff to register his credentials and performs authentication when he attempts to log in using those credentials.

4.3.2. *Classes*

For the purpose of this research, the prototype of a recommender system for rental properties requires only two types of users: a staff member (for the property management firm) and prospective tenants. The staff performs three main tasks, namely adding new rental properties to the database, editing them and deleting them from the database. The staff member needs to be authenticated to ensure that only authorised individuals can carry out such sensitive tasks. The authentication process requires an email and a password, which require registration before authentication. The staff member also performs tasks carried out by regular users: searching, viewing and selecting properties.

Prospective tenants, the target users of the system, will perform the tasks of locating rental properties. This task includes searching, viewing and selecting rental properties. As such, the system should provide functionalities that satisfy these user requirements. The system provides a search panel, where a user can enter attributes of rental properties that are relevant to her search and indicates the importance of such characteristics. The system also provides a search results panel that displays to the user the rental properties that match his or her preferences.

The user can select a rental property if he believes that it meets his preferences. The system further provides functionalities to enable the user to refine the search preferences in case the displayed examples do not satisfy the user's preferences. This search panel also displays the preferences the user has already stated, thus allowing him to modify them. These are major processes the system supports.

From the above description of processes supported by the recommender system for rental properties, a few possible classes can emerge. The first class is the *user class*, which represents users of the system and actions they can perform. This class has no attributes as no information about users is required for them to carry out the tasks described above. The second class is the *staff class*. This class represents the staff members of a particular property management firm and contains their attributes and actions they can perform. The third class is the *property class* that depicts rental properties and their attributes. The fourth class is the *preference model class*, which aggregates the user preferences expressed on rental properties.

The relationships between the various classes can be described in the following ways. A staff member manages many properties while a property is managed by one staff member. A user searches for many properties, and a property is searched by many users. A user has one preference model, and a preference model belongs to a user.

4.4. Object-oriented design

The objective of this phase is to design the classes and the user interface defined in the analysis phase. In particular, a Class Diagram was designed. User interfaces are also designed in this phase, and they are made of mark-ups with no implementation code. To avoid duplication, the researcher decided to skip the design of user-interfaces using mark-ups and instead provided user interfaces generated from working code in the next chapter.

4.4.1. Classes

The *Staff* class has three attributes and five methods. The attributes are name, email, and password. The methods are 'register', 'login', 'add new property', 'edit existing property' and 'delete existing property'. These attributes and methods adequately satisfy the requirements of this class, for the purpose of this study.

The user class has no attributes but has four methods. These are 'express preferences', 'search for property', 'view property', and 'select property'. For the purpose of the research, these methods will adequately capture the behaviour of the prospective tenant in the system.

The property class has the following primary attributes: type, location, rent, number of bedrooms and number of bathrooms. In addition to these, it has derived attributes including the cost function with respect to each primary attribute, an aggregate cost function, a set of dominating properties with respect to the current preference model, and a probability of escaping these dominance relations.

The preference model class aggregates the preferences expressed by a prospective tenant on properties. Its attributes include the value and importance of the preferences expressed on each attribute of a rental property (type, location, rent, number of bedrooms and number of bathrooms).

4.4.2. Class diagram

A class diagram provides a visual representation of all these classes, their attributes and methods, and the relationships between them. The class diagram for the proposed system appears in Figure 4.6.

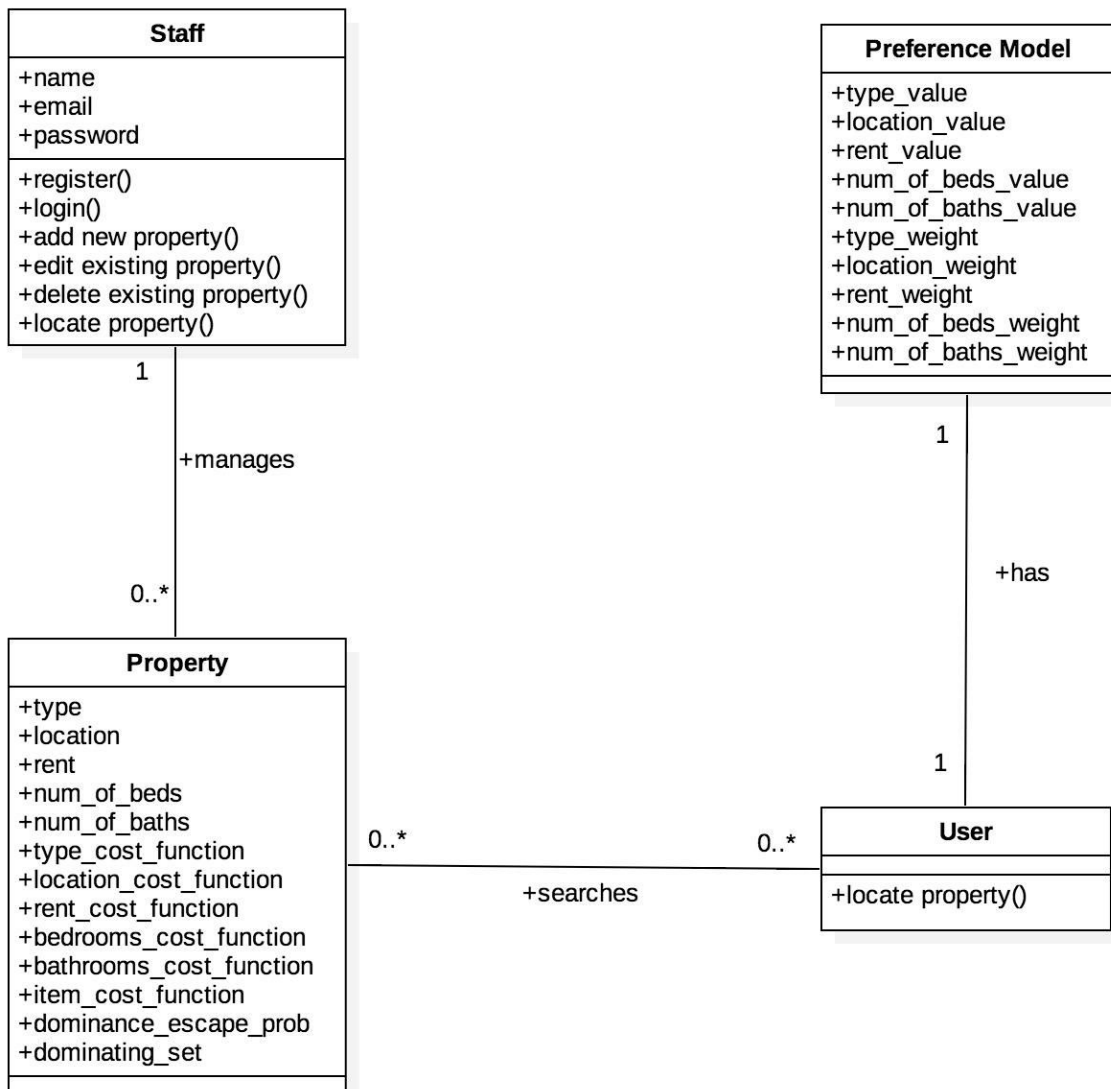


Figure 4.6: Class Diagram of the Proposed Recommender System

4.4.3. Entity Relationship Diagram

An Entity Relationship Diagram (ERD) is a picture that shows the information that is created, stored and used by an information system (Dennis et al., 2012). An ERD has three main component namely entities, attributes and relationships. Entities are the basic building blocks for a data model. They group together similar information in boxes. Attributes are information that is captured about entities. They included in the boxes that represent entities. Relationships are associations between entities. They indicate how separate entities relate to each other.

In the context of this study, we have four entities: Staff, User, Property, and Preference Model. The relationships between these entities are explained as follows: a staff member manages zero or many properties while one staff member manages a property; a user locates zero or many properties while a property is located by zero or many users; a user has one preference model, and a preference model belongs to one user. The attributes of various entities appear in the boxes representing those entities as shown in Figure 4.7.

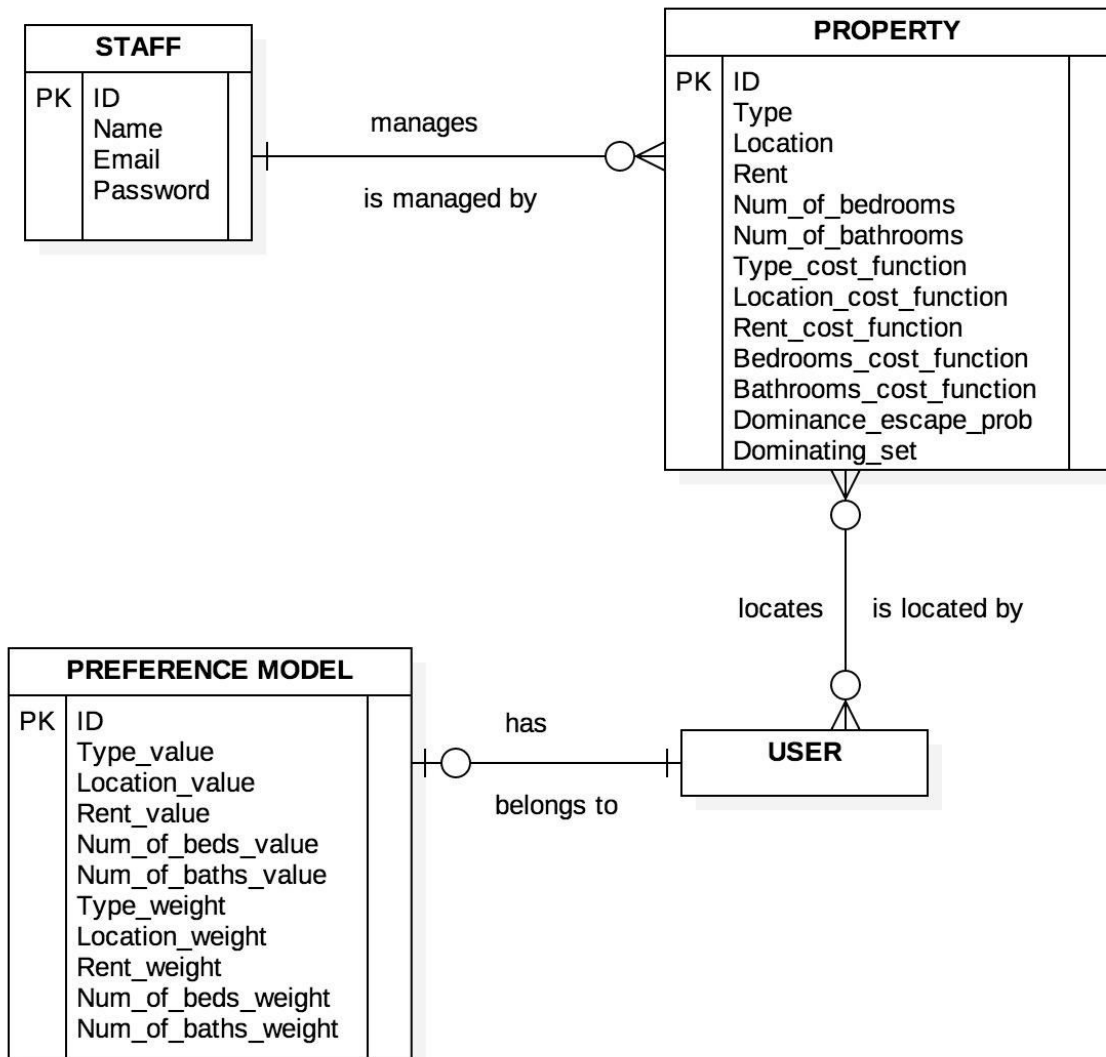


Figure 4.7: Entity Relationship Diagram for the proposed recommender system

CHAPTER FIVE: SYSTEMS IMPLEMENTATION AND TESTING

5.1. Introduction

This chapter addresses the last phase of SDLC: systems implementation. In this phase, the researcher built the code base of the system and tested the various functionalities of the system to ensure that it performed as designed. The researcher used the Ruby on Rails framework to develop the system. The framework contains various types of files. The logic of the system was built using ruby files, while the user interface was built using HTML and CSS files. The researcher used SQLite3 as a development database, PostgreSQL as a production database, and WEBrick as the web server. The system was deployed on Heroku; a cloud Platform-as-a-Service used as web application deployment model.

The architecture of the proposed recommender system has three main components: a database, a webserver and a web client. The database stores data on properties, users, and preference models of the users each in its own table. The web server receives and processes HTTP requests to perform CRUD (create, read, update and delete) actions on the data stored in the database. The web client provides the user interface that enables users to interact with the system. The system architecture is illustrated in Figure 5.1.



Figure 5.1: Application architecture of the proposed recommender system

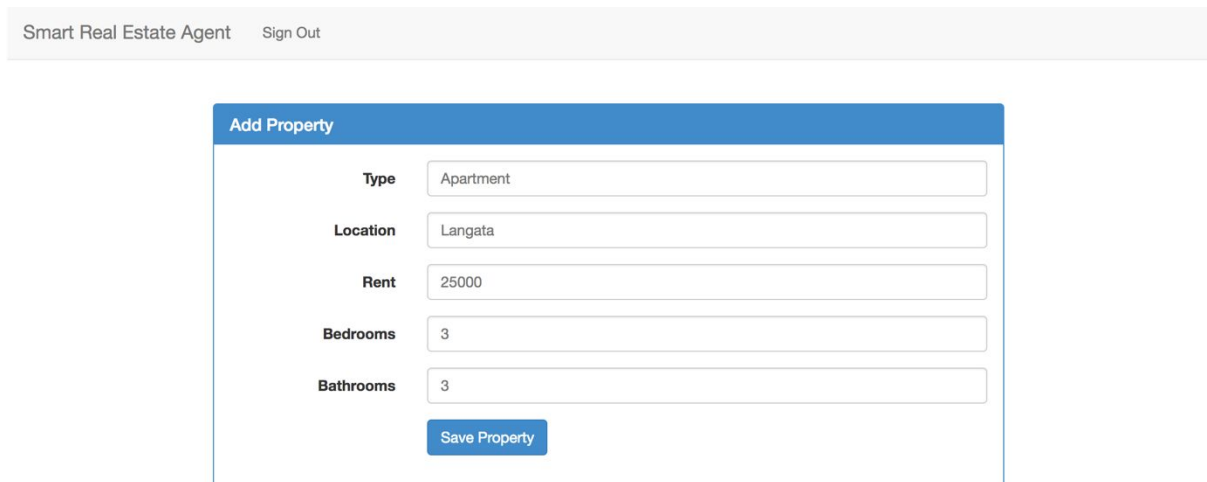
5.2.

User interfaces

User interfaces were built using HTML for content and CSS for styling. Instead of writing CSS from scratch, the researcher leveraged the functionalities of Twitter's Bootstrap front-end framework for web application development to quickly and efficiently style HTML pages. The following are the main user-interfaces required in the proposed system:

5.2.1. User interface for adding a new property to the database

This interface is used by a staff member to populate the database with rental properties that users can search. This process must be accomplished before other users can start using the system. The landing page of the system does not explicitly make reference to this functionality as it is only used by the staff. Furthermore, the system must authenticate the staff in order for him to perform this task. This user interface is illustrated in Figure 5.2.



The screenshot shows a web application interface. At the top, there is a navigation bar with the text "Smart Real Estate Agent" and a "Sign Out" link. Below this is a modal window titled "Add Property". The modal contains a form with the following fields:

Type	Apartment
Location	Langata
Rent	25000
Bedrooms	3
Bathrooms	3

At the bottom of the form is a blue button labeled "Save Property".

Figure 5.2: User interface for adding new properties to the database

5.2.2. User interface for editing details of a property

This functionality is only used by a staff member to edit details of a property that exist in the database. This action requires authentication by the system. This user-interface is illustrated in Figure 5.3.

Figure 5.3: User interface for editing details of a property

5.2.3. *User interface for removing a property from the database*

This functionality is only used by a staff member to remove from the database any rental property whose existence in the database may no longer be required. This action requires authentication by the system and is only performed by the staff member. This user-interface is illustrated in Figure 5.4.

Property Type	Property Location	Rent Amount	Bedrooms	Bathrooms	Edit	Delete
Apartment	South B	KES 35,000	2	2	Edit	Delete

[Back](#)

Figure 5.4: User interface for removing a property from the database

5.2.4. *User interface for searching property*

This is the main functionality of the recommender system: it enables users to perform preference-based search against the database with the view of locating preferred rental properties. This interface displays a search panel in which the user specifies attributes of rental properties relevant to their search, indicates the importance of each attribute, and clicks on the 'Search' button to initiate the search.

The system is configured to require the expression of preference on at least one attribute. If the user does not specify any attribute, the system will display an error message and prompt the user to indicate at least one attribute. Furthermore, the system initially sets values of the importance or weight of those preferences to a default value of 3, which is neutral. This user-interface is illustrated in Figure 5.5.

Smart Real Estate Agent

State initial preferences

Please select any feature of a rental property and indicate how important this feature is to you.

Property Type	<input type="text" value="Apartment"/>	Importance	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5
Property Location	<input type="text" value="Langata"/>	Importance	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5
Rent Amount	<input type="text" value="20000"/>	Importance	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5
No. of Bedrooms	<input type="text" value="2"/>	Importance	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5
No. of Bathrooms	<input type="text" value="2"/>	Importance	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5

Figure 5.5: User interface for searching property

5.2.5. *User interface for reviewing search results, selecting a preferred rental property, and revising preferences*

When the preference-based search is performed, the system will display a new webpage containing three main elements. These include the search panel with the previously stated preferences, a results panel displaying five rental properties that best satisfy the already-expressed preferences (current preference model), and a suggestions panel containing five rental properties that are supposed to encourage the use to express more preferences. In the case the user decides to revise her preferences, she does so by amending the preferences in the search panel and clicking on the 'improve your search' button. The same webpage will reload but now it will contain updated data that reflects the new preferences. This user-interface is illustrated in Figure 5.6.

Your preference has been recorded

You stated the following preferences

You can change any preference based on the results shown.

Property Type	<input type="text" value="Apartment"/>	Importance	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input checked="" type="radio"/> 5
Property Location	<input type="text" value="Langata"/>	Importance	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input checked="" type="radio"/> 4	<input type="radio"/> 5
Rent Amount	<input type="text" value="35000"/>	Importance	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input checked="" type="radio"/> 5
No. of Bedrooms	<input type="text" value="3"/>	Importance	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input checked="" type="radio"/> 4	<input type="radio"/> 5
No. of Bathrooms	<input type="text" value="2"/>	Importance	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input checked="" type="radio"/> 4	<input type="radio"/> 5

[Improve Your Search](#)

Results

Property Type	Property Location	Rent Amount	Bedrooms	Bathrooms	Select
Apartment	Syokimau	KES 32,000	3	2	Select
Apartment	Roysambu	KES 35,000	3	2	Select
Apartment	Langata	KES 45,000	3	2	Select
Apartment	Langata	KES 50,000	3	2	Select
Apartment	South B	KES 35,000	2	2	Select

These are the results that match your query the best.

Suggestions

Property Type	Property Location	Rent Amount	Bedrooms	Bathrooms
Apartment	South B	KES 35,000	2	2
Apartment	Ruaka	KES 50,000	3	2
Apartment	Lavington	KES 50,000	2	1
Apartment	Kiambu Road	KES 45,000	2	2
Apartment	Ongata Rongai	KES 28,000	3	1

Examine these suggestions and consider whether you need to revise some of your preferences.

Figure 5.6: User interface for reviewing search results, selecting a preferred rental property, and revising preferences

5.2.6. User interface for selecting a preferred rental property

This interface is a result of the user clicking on the 'Select' button next to a property they prefer. It shows the selected property on its own webpage. The interface is illustrated in Figure 5.7.

Details of the rental property				
Property Type	Property Location	Rent Amount	Bedrooms	Bathrooms
Apartment	Syokimau	KES 32,000	3	2

[Back](#)

Figure 5.7: User interface for selecting a preferred rental property

5.2.7. User interface for registering as a staff member

This interface is used to register the credentials of a staff member to enable authentication when he performs actions that modify the database. This user interface is illustrated in Figure 5.8.

Register	
Name	<input type="text"/>
Email	<input type="text"/>
Password	<input type="password"/>
	<input type="button" value="Register"/>

Figure 5.8: User interface for registering as a staff member

5.2.8. User interface for authenticating the staff member

This user interface is used to authenticate the staff member. If the credentials provided by the staff are not valid, the system will display an error message and will not perform login action. Furthermore, system is configured to *protect* actions that require authentication from any attempt to perform them without authentication. This means that if any user attempts to navigate to the parts of the application that handle adding rental properties to and removing rental properties from the database, the system displays an 'Accessed Denied!' error message and redirects the user to the login page.



The image shows a web form titled "Sign In" with a light green header. Below the header, there are two input fields: "Email" and "Password". The "Email" field is a simple text input, and the "Password" field is a text input with a small eye icon on the right side to toggle visibility. Below the "Password" field is a green "Sign In" button.

Figure 5.9: User interface for authenticating a staff member

5.3. Organisation of the code base

The code that implements the core algorithms used in this prototype is written in the ruby programming language. Ruby on Rails, being MVC (Model-View-Controller) framework, these algorithms are housed in models (that map to classes defined in *Chapter Four*). In particular, these algorithms that implement the logic of generating results for the search defined in the 'property' model. The 'Views' are responsible for displaying user interfaces, while controllers are responsible for managing the flow of data between the database, the models and the views.

5.4. Database

This study employed the SQLite3 database in the development environment as this is the default database for the development environment. When the application was deployed to the production environment, the database used was PostgreSQL, as it is the default database for, Heroku, the cloud-based platform used in this study to deploy the application. Figure 5.10 shows a snapshot of the 'properties' table in the database in the development environment.

property_type	property_location	rent_amount	num_of_bedrooms	num_of_bathrooms
Apartment	South B	35,000	2	2
Apartment	Ruaka	50,000	3	2
Apartment	Lavington	50,000	2	1
Apartment	Kiambu Road	45,000	2	2
Apartment	Ongata Rongai	28,000	3	1
House	Ongata Rongai	50,000	4	3
Apartment	Roysambu	35,000	3	2
Apartment	Ruaka	45,000	2	2
Apartment	Kilimani	45,000	1	1
House	Karen	50,000	1	1
Apartment	South B	45,000	2	2
Apartment	Mlolongo	25,000	2	2
Apartment	Syokimau	25,000	2	2
Apartment	Syokimau	32,000	3	2
Apartment	Kilimani	12,000	1	1
Apartment	Karen	40,000	2	1
Apartment	Uthiru	50,000	3	3
Apartment	Karen	25,000	1	1
Apartment	Ruaka	45,000	2	2
Apartment	Ongata Rongai	9,000	1	1
Apartment	Langata	45,000	3	2
Apartment	Ruaka	35,000	1	1
Apartment	Langata	50,000	3	2
Apartment	Langata	40,000	2	1
Apartment	Langata	32,000	2	1
Studio	Mlolongo	17,000	0	1
Studio	Syokimau	17,000	0	1
Apartment	Kilimani	20,000	1	1
Apartment	Imara Daima	50,000	3	3
House	Ongata Rongai	45,000	4	2
Apartment	Syokimau	40,000	3	3
Apartment	Mlolongo	23,000	1	1
Apartment	Karen	18,500	2	1
Apartment	Langata	15,000	1	1

Figure 5.10: A snapshot of the 'properties' table in the database

5.5. Application test results

The researcher used the Test-Driven Development approach to ensure that the system performed as designed. In doing so, the researcher carried out three different types of test: unit tests, functional tests and integration tests. Unit tests examine different components in isolation. Functional tests are carried out to test multiple components in collaboration. Integration tests follow a business process: components work together to achieve a business objective.

Rails, the framework used to develop the prototype, being a Model-View-Controller (MVC) framework, unit tests were carried on the "User" and "Property" models. Functional tests were carried out on the "Properties Controller", "Users Controller", "Sessions Controller", and "Preference Model Controller." Integration tests were carried on major business processes including adding, viewing, editing, and deleting a property; and viewing all properties. Furthermore, locating a property (an action that includes searching, viewing and selection a property) was also tested. Finally, the actions of registering and authenticating staff members were also tested.

Table 5.1 provides a summary of business processes tested during integration tests.

Functionality	Tested?	Passed?
Register a staff member	Yes	Yes
Authenticate a staff member	Yes	Yes
Add a new property to the database	Yes	Yes
View all properties in the database	Yes	Yes
View a particular property	Yes	Yes
Edit a particular property	Yes	Yes
Delete a particular property from the database	Yes	Yes
Search for properties	Yes	Yes
Generate and display search results	Yes	Yes
View search properties	Yes	Yes
Select a preferred property	Yes	Yes

Table 5.1: Summary of integration tests results

The prototype proposed in this study was tested on various web browser including Google Chrome (Version 57.0.2987.98 (64-bit)), Mozilla Firefox (52.0.1), Safari (Version 10.0.3 (12602.4.8) and Internet Explorer 11 both on Windows machine and a Mac machine where possible.

The researcher further carried out performance tests to determine how much time it took for a user to perform certain tasks. Performance testing is a non-functional testing technique to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing concerns the quality attribute of the system such as scalability, reliability and resource usage (Performance Testing, n.d.). These quality attributes can include speed, scalability, stability and reliability. The researcher tested the speed quality of the system to assess how quickly the system responded on user actions.

Functionality	Tested?	Time in milliseconds
Search properties	Yes	1,890
Improve search results	Yes	1,580
Log in	Yes	98
Add new property	Yes	150
Edit property	Yes	163
Delete property	Yes	77

Table 5.2: Summary of performance test results for speed testing

CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS

6.1. Introduction

Locating items of interest online has become increasingly challenging for users given the large pool of choices available to review while searching for items that meet their needs. The situation is also true for searching for rental properties online in Kenya. Currently available tools for querying databases fall short of the expectation of users, and many of them may be unsatisfied with the results. In light of this, the researcher proposed the development of a prototype of a recommender system for rental properties. This chapter discusses the conclusions and recommendations of this study.

6.2. Conclusion

This study had four main objectives including (i) to review recommender systems technology currently in use, (ii) to evaluate algorithms that can be applied to recommend rental property, (iii) to develop a prototype of a recommender system for rental property, and (iv) to validate the proposed system. The next paragraphs indicate how these objectives were achieved in the course of this research endeavour.

The study achieved the first objective (to review recommender systems technology currently in use) in Chapter 2 Section 2. In this section, the study examined the currently available recommender system technologies focusing on the definition and evolution of the recommender systems. It further discussed various functions of a recommender system and reviewed briefly various recommendation approaches. These include collaborative filtering, content-based, knowledge-based, community-based, demographic-based, hybrid approaches.

The research achieved the second objective (to evaluate algorithms that can be applied to recommend rental property) through Sections 3, 4 and 5 of Chapter Two.

Section 3 discussed various types of critiquing-based recommender systems including natural language-based systems, system-suggested critiquing systems and user-initiated critiquing systems. These systems use various algorithms to arrive at their recommendations.

Section 4 describes in details the user-system interactions used in user-initiated critiquing systems focusing on the user preference elicitation, system recommendation, user feedback and selection of desired item. These are steps used by algorithms found in this type of recommender system. Section 5 presents one particular algorithm used in this study to generating suggestions for recommendation. The algorithm is presented in the form of pseudo-code.

The research achieved the third objective (to develop a prototype of a recommender system for rental property in Chapter Four and Five. In Chapter Four, Section 2 dealt with the creation of use cases; Section 3 addressed system analysis through requirements gathering and class definition; Section 4 dealt with systems design by designing relevant classes and producing a class diagram. In Chapter Five, Section 1 presents the application architecture of the system; Section 2 displayed user-interfaces; Section 4 presented a snapshot of the database.

The study achieved the fourth objective (to validate the proposed system) by presenting the results of testing the system to ascertain that it performed as the researcher intended.

6.3. Limitations of the prototype

If the number of attributes in which users express preferences increasing, the user efforts also increase making it harder to the system to be used. The main objective of this recommender system was to facilitate locating rental properties. Therefore, if the number of attributes of rental properties increases, this may undermine the primary objective of building such a system. Consequently, the researcher limited the number of attributes on which users can express preferences to only five. Expressing

attributes on five attributes is not overwhelming to users and thus the objective of facilitating locating rental properties can be achieved.

6.4. Recommendations

With the increased availability of Internet connection in Kenya and the reducing costs of accessing the Internet, coupled with increased interest in online commerce, many consumers are turning to online resources to locate items that are of interest to them. It is, therefore, important for businesses to invest in technology that supports their customers to easily access the information they require to find what they are looking for and ultimately make purchases. This analysis applies to the rental properties market. The development of the prototype of a recommender system for rental property in this study is an effort in this direction.

6.5. Future works

Many Kenyans use the Internet to access information. Many of those who access information through the Internet use their mobile phones. The majority of smartphones in the Kenyan market run on Android platform. Research work that attempts to implement a recommender system for rental properties or any other items on Android devices would be an effort worth making both for its academic value and market potential.

REFERENCES

- Abdellaoui, M., & Gonzales, C. (2009). Multiattribute Utility Theory. In D. Bouyssou, D. Dubois, M. Pirlot, & H. Prade (Eds.), *Decision-making Process: Concepts and Methods*. London: ISTE Ltd.
- Babikir, H., Ali, A. B., & elWahab, A. M. (2009). Research Methodology Step by Step Guide for Graduate Students. *Sudanese Journal of Paediatricians*, 9, 9-22.
- Buetner, R. (2016). Predicting user behavior in electronic markets based on personality-mining in large online social networks: A personality-based product recommender framework. *Electroni Markets: The International Journal on Networked Business*, 1-19.
- Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of Library and Information Science*.
- Burke, R. D. (2002). Hybrid Recommender Systems: Survey and Experiments: User modeling and User-Adapted Interaction. *The Journal of Personalization Research*, 331-370.
- Butler, J., Morrice, D. J., & Mullarkey, P. W. (2001). A Multiple Attribute Utility Theory Approach to Ranking and Selection. *Management Science*, 47(6), 800-816.
- Chen, L., & Pu, P. (2007). Preference-based Organisation Interfaces: Aiding User Critiques in Recommender Systems. In C. Conati, K. McCoy, & G. Paliouras (Eds.), *User Modeling 2007: 11th International Conference* (pp. 77-86). Springer Berlin Heidelberg.
- Chen, L., & Pu, P. (2012). Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1), 125-150.
- Cooper, D. R., & Schindler, P. S. (2011). *Business research method* (11 ed.). New York, USA: McGraw-Hill Higher Education.
- Dennis , A., Wixom, B., & Roth, R. (2012). *System Analysis and Design*. Hoboken, New Jersey, USA: John Wiley & Sons, Inc. .
- Ekstrand, M., Riedl, T., & Konstan, J. (2010). Collaborative Filtering Recommender Systems. *Foundations and Trends in Human-Computer Interaction*, 4(2), 81-173.
- Estate Agents Registration Board. (2016). *List of Registered Members*. Retrieved 11 23, 2016, from Estate Agents Board: <http://estateagentsboard.or.ke/>
- Faltings, B., Pu, P., & Torrens, M. (2003). Solution Generation with Qualitative Models of Preferences. *Computational Intelligence*, 20(2), 1-22.
- Felfernig, A., Friedrich, G., Jannach, D., & Zanker, M. (2011). Developing Constraint-based Recommenders. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor, *Recommender Systems Handbook*. New York.
- Grabisch, M., Kojadinovic, I., & Meyer, P. (2008). A review of methods for capacity identification in Choquet integral based multi-attribute utility theory:

- Applications of the Kappalab R package. *European Journal of Operational Research*, 186(2), 766-785.
- Hurley, G., & Wilson, D. C. (2001). DubLet: An Online CBR System for Rental Property Recommendation. In D. W. Aha, & I. Watson (Eds.), *Case-Based Reasoning Research and Development* (pp. 660-674). Springer Link.
- Kailiponi, P. (2010). Analyzing evacuation decisions using multi-attribute utility theory (MAUT). *Procedia Engineering* . 3, pp. 163-174. Elsevier Ltd.
- Karypis, G., Konstan, J., Sarwar, B., & Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. *10th International World Wide Web Conference* (pp. 285-295). Hong-Kong: WWW10.
- Kenya Bankers Association. (2015). *The State of Urban Home Ownership in Kenya: A Survey*. Center for Research on Financial Markets and Policy. Nairobi: Kenya Bankers Association.
- Konstan, J., & Riedl, J. (2012, 04). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22, 101-123.
- Kothari, C. R. (2004). *Research Methodology: Methods and Techniques* (2 ed.). New Delhi: New Age International Publishers.
- Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering . *Artificial Intelligence Review*, 13, 393-408.
- Performance Testing*. (n.d.). Retrieved 04 01, 2017, from Tutorials Point: https://www.tutorialspoint.com/software_testing_dictionary/performance_testing.htm
- Pu, P., & Faltings, B. (2000). Enriching buyers' experiences: the SmartClient approach. *SIG- CHI Conference on Human Factors in Computing Systems* (pp. 289-296). New York: ACM.
- Pu, P., & Faltings, B. (2004). Decision Tradeoff Using Example-Critiquing and Constraint Programming. *Constraints*, 9(4), 289-310.
- Pu, P., Chen, L., & Kumar, P. (2008). Evaluating product search and recommender systems for E-commerce environments. *Electronic Commerce Research*, 8(1), 1-27.
- Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (2011). Introduction to Recommender System Handbook. In *Recommender System Handbook* (pp. 1-35). Springer.
- Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). *Recommender Systems Handbook*. New York: Springer.
- Rogers, E. M. (2003). *Diffusion of innovations*. New York: Free Press.
- Sarantakos, S. (2005). *Social Research*. New York: Palgrave MacMillan.
- Scarrett, D. (1995). *Property Asset Management*. London, UK: E & FN Spon.
- Schafer, J., Konstan, J., & Riedl, J. (2001, 01). E-commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, 5(1), 115-153.
- Selltiz, C., Jahoda, M., Deutsch, M., & Cook, S. W. (1967). *Research Methods in Social Relations*. New York: Holt, Winehart and Winston.
- Sharma, R., & Singh, R. (2016, 05). Evolution of Recommender Systems From

- Ancient Time to Modern Era: A Survey. *Indian Journal of Science and Technology*, 9(20), 1-12.
- Smith, S. (2013, 4 8). *Determining Sample Size: How to Ensure You Get the Correct Sample Size*. Retrieved from qualtrics: <https://www.qualtrics.com/blog/determining-sample-size/>
- The Standard. (2014, 10 14). *Landlords in Kenya now shunning property agents*. Retrieved 11 03, 2016, from Standard Media: <http://www.standardmedia.co.ke/sports/article/2000138202/landlords-in-kenya-now-shunning-property-agents>
- The World Bank. (2011). *Developping Kenya Mortgage Market*. The International Bank for Reconstruction and Development . New York: The World Bank.
- The World Bank. (2016). *Food and Agriculture Organization and World Bank population estimates*. Retrieved 11 03, 2016, from World Bank Group: <http://data.worldbank.org/indicator/EN.POP.DNST?locations=KE>
- The World Bank. (2016). *Urban population (% of total)*. Retrieved 11 03, 2016, from World Bank Group: <http://data.worldbank.org/indicator/SP.URB.TOTL.IN.ZS?locations=KE>
- United Nations. (2014). *World Urbanization Prospects: The 2014 Revision* . United Nations, Population Division , Department of Economic and Social Affairs . New York: United Nations.
- University of Georgia, Terry College of Business. (2012). *Research & Methodology Content Analysis as a Research Technique*. Retrieved 04 06, 2017, from Terry College of Business: <https://www.terry.uga.edu/management/contentanalysis/research/>
- Viappiani, P., & Faltings, B. (2006). *Design and Implementation of Preference-based Search*. Ecole Polytechnique Fédérale de Lausanne, Artificial Intelligence Lab, Lausanne.
- Viappiani, P., Faltings, B., & Pu, P. (2006, 12). Preference-based Search using Example-Critiquing with Suggestions. *Journal of Artificial Intelligence Research*, 465-503.
- Yuan, X., Lee, J.-H., Kim, S.-J., & Kim, Y.-H. (2013). Toward a user-oriented recommendation system for real estate websites. *Information Systems*, 38(2), 231-243.
- Zeng, J., Li, F., Liu, H., Wen, J., & Hirokawa, S. (2016). A Restaurant Recommender System Based on User Preference and Location in Mobile Environment. *5th International Congress on Advanced Applied Informatics*, (pp. 55-60). Kumamoto.