VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECISION SUPPORT SYSTEM FOR TOURISM

By
Pham Minh Quan – ITITIU21289

A thesis submitted to the School of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Information Technology/Computer Science/Computer Engineering

Ho Chi Minh City, Vietnam
2025

DECISION SUPPORT SYSTEM WEB APPLICATION FOR TOURISM

APPROVED BY:

_____ ,
Assoc. Prof. Dr. Nguyen Thi Thuy Loan, Chair

_____
Assoc. Prof. Dr. Nguyen Van Sinh

_____
Dr. Nguyen Thi Thanh Sang

THESIS COMMITTEE

ACKNOWLEGMENTS

It is with deep gratitude and appreciation that I acknowledge the professional guidance of Assoc. Prof. Nguyen Van Sinh, whose constant checking and support helped me to keep the project on time and achieve my goal.

My gratitude goes to the other members of the laboratory, mainly, Mr. Nguyen Trung Nghia His technical help has always been useful for me ever since the first course and his friendly demenor contribute to years of great learning experience. I am grateful to the faculty of the Department of Computer Science of the International University, especially Dr. Le Hai Duong for his support guidance since the beginning of my studies. Gratitude is also expressed to the members of my reading and examination committee.

TABLE OF CONTENTS

ACKNOWLEGMENTS 3
TABLE OF CONTENTS 4
LIST OF FIGURES 5
LIST OF TABLES 5
ABSTRACT 6
INTRODUCTION 7
1.1 Background 7
1.2 Problem Statement 7
1.3 Scope and Objectives 8
1.4 Assumption and Solution 10
1.5 Thesis structure 11
LITERATURE REVIEW 13
2.1 Overview 13
2.2 Related work 13
2.3 Literature review 16
METHODOLOGY 26
3.1. Overview 26
3.2. Thesis Procedure 26
3.3. User requirement analysis 27
3.4. Sequence Diagram 32
3.5. System Design 33
3.5.1. System Architecture: 33
3.5.2. Design Pattern 35
3.5.3. Datadase design 36
3.5.4. User Interface (UI) design 39
IMPLEMENT AND RESULTS 43
4.1. Implement 43
4.1.1. Overview 43
4.1.2. Code Implementation 44
4.2. Results 52
4.2.1. HomePage / Landing Page 52
4.2.2. User Form 53
4.2.3. Result Page 55
4.2.4. Destination Search Page 56
DISCUSSION AND EVALUATION 60
5.1. Discussion 60
5.2.1. Compare with TravelMyne 60
5.2.2. Compare with Wonderplan 62
5.2.3. Conclusion 63
5.3. Evaluation 64
5.3.1. Effectiveness of the System 64
5.3.2. Strength and Limitations 64
CONCLUSION AND FUTURE WORK 66
REFERENCES 68
APPENDIX 68

LIST OF FIGURES

Figure 1.1 Structure Diagram of Thesis 12
Figure 2.1. Landing Page of Wonderplan 13
Figure 2.2. Landing Page of Wonderplan 15
Figure 3.1. Use case diagram of DSS for tourism choice 29
Figure 3.2. Decision Support System Sequence Diagram 32
Figure 3.3. Simple illustration of Monolith Architecture 33
Figure 3.4. Illustration of MVC design pattern 35
Figure 3.5. Models used in DSS for tourism choice 38
Figure 4.1. Frontend Folder Structure 44
Figure 4.2 Backend Folder Structure 45
Figure 4.3 Landing Page 53
Figure 4.4.1 User Form Page 54
Figure 4.4.2 User Form Expanded 54
Figure 4.5.1. Result Page 55
Figure 4.5.2. User Preferences Tab, now open 56
Figure 4.6.1. Destination Finder 57
Figure 4.6.2.a & 4.6.2.b Detailed Location Popup 58
Figure 4.7. 404 Not Found Page 59

LIST OF TABLES

Table 3.1: Use Case Document 30
Table 3.2: Use Case Detail 37
Table 3.3: sort() Pseudo Code 41
Table 3.4: slice() Pseudo Code 42
Table 4.1 createNewResult() Pseudo Code 46
Table 4.2 calculatedBudgetPoint() Pseudo Code 47
Table 4.4. calculatedActivityPoint() Pseudo Code 50
Table 4.5. calculatedActivityPoint() Pseudo Code 52
Table 5.1 Comparision Table with TravelMyne 62
Table 5.2 Comparision Table with Wonderplan 63

ABSTRACT

Choosing a place to stay on holiday has always been a challenging task given the diverse location on our planet ranging from bustling and vibrant cities with crowd of people participating in various activities, to tropical and pristine beaches in tropical countries. While these days, how one approach to researching for where to go has been easier than ever thanks to the invention of Internet, with many aspects to juggling it still possible to miss small details, resulting in a disappointing vacation. And, considering the rising cost of today life, not only does the failed trip waste precious time of your life, it also will cost you a fortune that you will undoubtedly regret.

To solve this issue, in this thesis, I propose a solution for choosing a suitable place for your holiday, using your preferences and condition, in the form of a Decision Support System. This decision support system aims to simplify the process of thinking and considering of the user by asking for user's input on the subject and recommending the satisfactory place in no time using the website's algorithm and up-to-date database. The system will use a combination of point-based algorithm to score and rank each place depending on how similar that place is to the user preferences, and a weighted point system where user can freely adjust the priority of each category to fine tune the system to their liking.

CHAPTER 1

INTRODUCTION

1.1 Background

Tourism as a concept has existed since medieval time. It is said that during that period, nobles of great wealth often travelled to learn more about the world's history, art, and culture heritage. With the advancement in technology, travelling has become easier and easier and more assessible for anyone no matter how wealthy they are.

There are many reasons why one may want to go on a holiday trip. Aside from the obvious benefit of enriching knowledge, travelling can also can be seen as an break from the boring, mudane everyday tasks, helping people to relax, and providing crucial times to self-reflecting for personal growth as well as building connections both old and new.

Nowadays, Tourism has quickly become an intergral part in our society. Despite the damage caused by Covid pandemic, international travelling has practically recovered to previous number by 2024 [1]. Additionally, as of 2023, Travel and Tourism sector accounted for nearly 10% of global GDP, and provided public with 27 million new jobs, an increase of 9.1% over 2022 [2]. While geopolitical and economic headwinds still proven to be a problem, overall, it seems that the future is bright for tourism.

1.2 Problem Statement

Such positive development, however, also come with their fair share of downsides, most noticeably, a ginormous spike in holiday destination often leads to travellers unable to identify their ideal holiday location, causing massive headache for many through a phenomenon psychologists dubbed as "choice overload". For Vietnam along, one quick search on the Internet will provide you with a plethora of destinations for you to choose from, ranging from big cities such as Ho Chi Minh, to beautiful beaches in Ha Long Bay.

Indecisiveness, or the inability to make a perfect choice, can be a problem for a good proportion (roughly 20%) of grownups [3]. In average, a normal people need to make around 35,000 decisions per day, ranging from insignificant such as what to eat today to important ones

like what jobs should they apply for. Therefore, indecisiveness could have detrimental effects, leading to stresses, and missed opportunities.

There are many factors one must consider when making a decision. It not only about logic and reason, but emotions also play a key role when considering all the available options. Picking the ideal location for your holiday can be difficult when you do not know where to go beforehand. While e-booking sites provide plentiful options for you to pick from pre-made tour, often time, they are only useful when you already have an idea of where to go. From places, prices and food to temperature and landscape, there are many elements that must be considered when choosing their holiday destination, and understandably, it can be overwhelming at times.

With that in mind, is there a system, or algorithm that can help aid people in making their ideal holiday? Well, there is. Implementation of a decision support system can help simplify the process of choosing their stay. Rather than let emotions control people choice, a decision support system only relies on facts and logic to provide people with what could be the perfect choice. Decision support system can also help to narrow down all possible options and give advice on places for people when needed, thus, should be a suitable tool for such task.

1.3 Scope and Objectives

1.3.1 Scope

This project only focuses on building a decision support system to assist users in choosing a location depending on their preferences for their holiday, such as scenery, weather, and budget, therefore, the website should only be used during the planning phrase of your holiday.

Specifically, the website will only be able use its algorithm to compare places in its database to show you information on what it considers the most suitable place depending on your preferences. The website itself will not have the ability to provide any additional support during your planning stage such as booking the tour for your stay or completing other travelling-related activities such as buying goods and applying for visa.

As the system is designed only with some aspects in mind (likes scenery, activities, food, budget and weather), additional niches preferences user may want may make the final result unsatisfactory. Further more, the system is built only for choosing holiday location for a single person, undesirable result may be provided when the website is used outside of its intended

purpose such as using it for work, or to provide the location for a group of people with different preferences.

Finally, the system usage of information from online preview and established travelling database, thus, some niche location might be missed when recommending options to the user.

1.3.2 Objectives

The aim of this thesis is to create a website application for suggesting holiday location for users to stay. Here are some key objectives that must be considered:

About the decision support system logic:

- The system must be designed so that multiple types of decision support system are incorporated to provide the best result
- The algorithm must be able to handle multi-criteria decision making approach and handle sorting and recommending option appropriately.
- The algorithm should be able to handle blank inputs and provide default value when considering its options
- The algorithm should consider each category's priority when sorting and user should be able to change the priority list to their desire.
- User should have the option to log in to save their preferences or use the system annonymously

About the website UI:

- The website should be easy to understand and use
- The website should be able to provide options to successfully represent user's preferences
- Information from the website must be display clearly and without any error

About the system's database:

- The database should have all table necessary for the website and the database must not have any duplicated entities.
- The database must be a relational database, and all relationship between table must be considered
- For user's table, their password must be encoded before being inserted into the table

About optimization and debugging:

- When using the website, user should be able to see the result in a short time with little to no wait.

- The website must not contain no major bugs or errors, and any bugs effect user's experience must be fixed immediately

1.4 Assumption and Solution

1.41. Assumption

For this website to work, some assumption will be made so that project will proceed smoothly and without any major setbacks.

The first assumption was about the interaction between users and the website. Users were assumed to honestly fill in all interactable field on the website so that result are as accurate as possible. Users' liking was also assumed to not change in between the filling of information for the website. Moreover, it was assumed that this website will only make recommendation for a single person reference and is not meant to be used to advice for a group of tourists.

The second assumption was about the information about location provided as a result of using the website. All information was assumed to be verified, and up to date whether it was about the price or activities to do. Additionally, all information was also assumed to remain the same for the foreseeable future, meaning no major changes to the place will occur and affect the scoring in the website's system.

Thirdly and finally, it was assumed that technologies used on the website were capable of handing the data given by each user and effectively provide the result in a timely manner and without any error.

1.42. Solution

This thesis proposes a decision support system or DSS as an advisory system for user to choose their ideal holiday location, hotel, as well as explore its rich culinary culture. In the system, user will be able to choose their preference using input field as well as selection box to provide their preference for the system to use. As for the system, a multi-criteria decision making approach will be employed when weighting all options in the database, and users will be able to change the significant of each category from 1 to 5 through a provided slider inside the website. The result will be provided after the user had finished entering all their preferences

together with the detailed information about the location, and explanation for the reason this result is chosen, and user will also be able to see other option in a separated page. From there, it is possible for user to look at all the top rated hotels with best deals, as well as see recommended food to try when they are there.

1.5 Thesis structure

Structure Diagram Explaination: The structure of the thesis is broken down into six chapters, starting with chapter 1: Introduction, then moving onto reviewing existing knowledge with chapter 2: Literature Review, discussing the solution in chapter 3: Methodology, presenting project with chapter 4: Implements and Results, and finally evaluating and discussion about the project in chapter 5: Discussion and Evaluation to come into a conclusion in chapter 6

Chapter 1 - Introduction: This chapter goes into the background of the project, states the problem statement for the problem of choosing holiday location, and provides solution in form of a decision support system as well as states limitations and gives objective for the thesis project.

Chapter 2 - Literature Review: This chapter go in depth into theory and existing knowledge related to the topic, mainly the concept of a decision support system, the benefits and drawbacks of the system, as well as the theory of all the technology used when building the project, including frontend, backend, and database. This chapter also shows two example of related works in TravelMyne and Wonderplan to be compared with the project.

Chapter 3 - Methodology: This chapter contains detailed information on how the thesis is conducted and what method used to do this research as well as the design of the research in both algorithm and User Interface. This chapter also analyzes user requirements for the project and showcases the database design for the project.

Chapter 4 - Implememtation and Results: This chapter goes in-depth into all function of the project and describe what each of them do through the usage of pseudo code. This chapter also showcases the final product.

Chapter 5 - Discussion and Evaluation: This chapter provides discussion on the implication of this project, and evaluate the project on its performance, acknowledge any limitation it has as well as compared it in detail with similar applications listed in chapter 2

Chapter 6 - Conclusion: This chapter gives quick overview of all findings and finish the thesis project. This chapter will only contain information already appeared in other chapters.

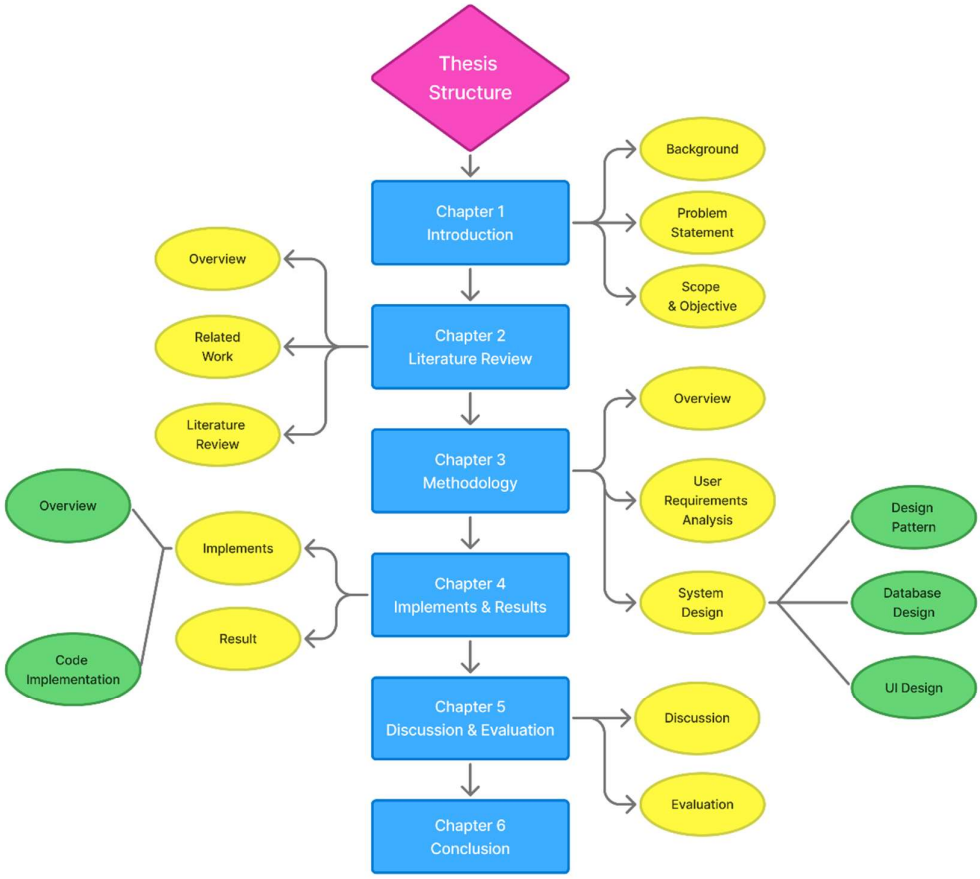

Figure 1.1 Structure Diagram of Thesis

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

The literature review section aims to contextualizing what will be look into, determines the gaps in exisiting knowledge, as well as identifies key terms and concept used to justify this project importance.

To do this, first, we will be looked into two cases of existing works, one using standard approach by asking user to fill out a preference form and use it to determine the most suitable places to recommend through logic: Travelmyne.com, and other using LLM to assist user in choosing the destination: Wonderplan. After that, we will go in-depth into some key concepts related to the study, mainly what classified as a decision support system, what types of decision support system there are, and how multi-criteria decision support system is used to fix one of ordinary decision support system weakness: unstructured tasks, and what are the methods that those multi-criteria decision support system use.

2.2 Related work

2.1.1 Travelmyne.com: A traditional, data and logic reliance Decision Support System

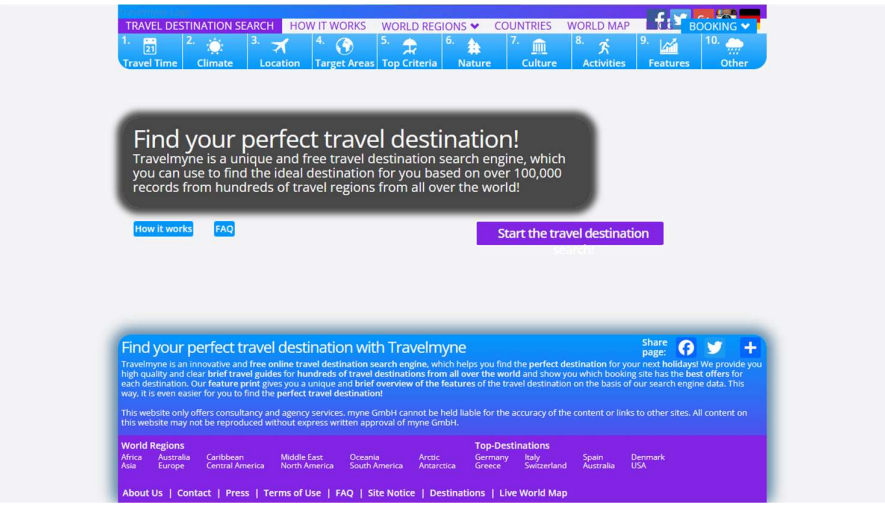

Figure 2.1. Landing Page of Wonderplan

TravelMyne is one of a few non-AI integrated website-based search engine that is used to help user find their next holiday location [4]. The website uses a form which contain 10 questions designed to filter and sort out the ideal place for the user. In their form, TravelMyne use a combination of checkboxes, text fields, as well as sliders to help user customize their preference before clicking on the search button to begin their search.

To get started, user will first need to fill in 5 question which will provide the website engine with the time of their trip, their preferred climate style, their location, as well as target region and prioritized criteria that they look for in the location. When finish, they then given the choice to start the search immediately with TravelMyne "Quick search" option, or to further customize their form by answering the remaining question to provide input on nature, culture, activities, features as well as other field to further narrow down the search. The result user obtain is a list of places that the website deems similar to user's preference, ranking by the most suitable. To get this list of recommendation, each location is ranked on a score-based system with algorithm to calculate depending on each criteria chosen. Each recommended location also comes with their own navigation bar for booking a complete trip with hotels and restaurant, general information such as price, weather condition, temperature. And if user want further information on the location, they can click on the photo of the location to get directed to the full information page of the site, including description on various topic, language and money type used, as well as information on major cities in that place.

For the advantages, TravelMyne is a simple, easy to use engine that successfully recommend user the list of location to consider when planning for their next trip and is a good example of a decision support system that can be used for recommending user where they should go on holiday on a timely manner and without any error. TravelMyne website also include a FAQ section for answering or helping user with popular questions which helps first time user quickly able to get the hang of things and start using the website effectively.

As for the disadvantages, TravelMyne use a simple point base system with equal point across all criteria, which restrict user on focusing on a single, or multiple criteria that they see as more important than the other. Furthermore, it is unknown whether the information on each location is up-to-date, and the website does not have a way to verify such information, leading to potentially outdated information that user unknowingly uses.

2.1.2 Wonderplan: an AI –based DSS

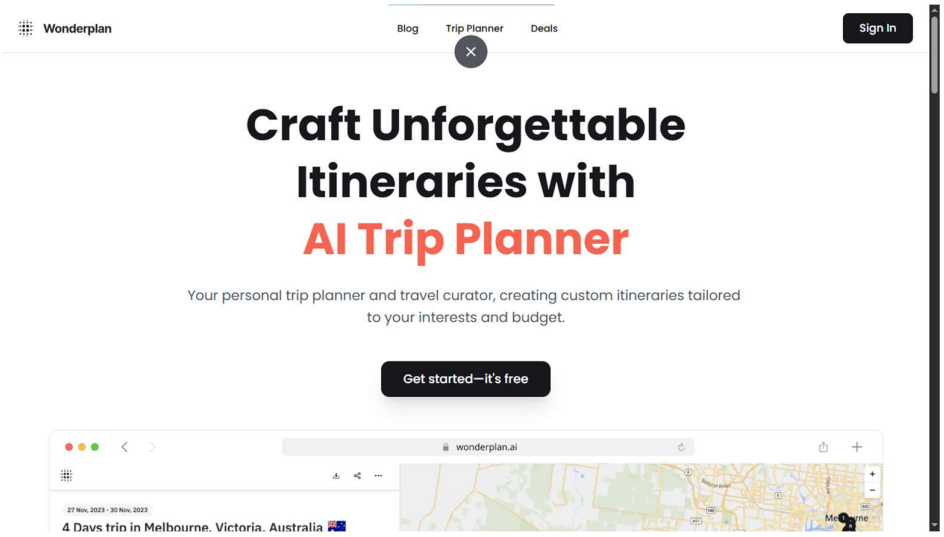

Figure 2.2. Landing Page of Wonderplan

Wonderplan is a free website aims to mimic traveling angency to guide users on choosing where to go on their holiday [5]. Wonderplan uses AI technology and algorithm to accurately tailor a perfect trip that meets all your needs. By using a form for user to fill in their preference, Wonderplan is both efficient, and easy to use, and user can get their recommendation with just a few buttons clicks, an in no time.

To get started, users are required to input some basic information to the form provided by the website such as their destination, when going, and budget. After clicking the submit button, the website will take the user's preferences and put inside an AI algorithm. The algorithm will then generate a trip with many points of interest by taking into account all your needs, ensuring every aspect you wanted is tackled and handled. When a plan is found, user will be able to see every information about the region such as general information, description, and its history. The website will also provide user with nearby hotels recommendation together with their information and price, as well as an estimated budget for the full trip at the bottom of the page. Users will also be able to take a good look of the daily activities planned by the website to view each point of interest information and can either add new recommendation or delete unwanted ones.

Wonderplan main advantages compare to other DSS is its highly personalize approach to recommending. Depending on the input, the recommended list will be varied greatly from person to person. Moreover, contrary to how popular locations are often suggested by traditional DSS, AI integrated DSS is able to provide lesser-known locations that are often

overlooked that align with user's interest. And, by learning from previous interaction, AI algorithm will be able to grow and becomes better as time goes by.

On the other hand, like all other technologies, AI integrated system also has many challenges that need to be dealt with. For starter, AI can be expensive and will cost a lot for the server to run on a long period of time. Given the nature of AI algorithm, sometimes it can be hard for human user to understand the choice of it. The algorithm itself also requires a great array of information to work properly, and the lack of usable information will negatively affect the quality of the recommended list for the user. Finally, while Wonderplan provides great suggestions, the page requires user to already have at least some ideas of where to go, meaning it cannot deal with people who just want to try out new things and generally does have any plan yet.

2.3 Literature review

2.2.1 Decision Support System (DSS)

DSS as a concept can be dated back to as early as 1960s, where computers were explored for their ability to plan and make decision. Scott Morton, a Harward University student, was the first to assert and implement a model-driven decision system in 1964 [6]. By late 1970s, dicussions surrond practice and theory issues related to DSS were held at academic conferences including the American Institute for Decision Sciences meetings and the ACM SIGBDP Conference on Decision Support Systems in San Jose, CA in January 1977, and later on, forums for idea sharing, theory discussions and information exchange were provided to many researchers. During this time, Ralph Sprague and Eric Carlson's book "Building Effective Decision Support Systems" published in 1982 provided important knownledge on the development of DSS. And with it, DSS as a concept was expanded beyond initial bussinesss concept as definition and classification of DSS became more and more refined.

In general, DSS is a term used to talk about interactive, software-based system whose purpose is to assist people in making a difficult choice using information provided to the system. Rather than being influenced by the rationality of human user, DSS use logical and factual reasoning provided in the system to calculate, identify, and determine the optimal solution for any problems, as well as giving insights that might be useful for the users. As an assisting tool, DSS goal is only to provide users with information and knownledge, and listening to any recommendation offered by the system is the choice users can make.

DSS, as a recommendation tool, has many characteristics that help define it. As stated by Sprague and Carlson (1982), DSS main usage is in solving less well-structured problems that people can encountered in their life. DSS combine analytical model with traditional database to make a flexible system that can adapt to changes and provide users with the best answer. DSS are built to be simple to use and contain features that help noncomputer people effectively use the system. Additionally, while DSS are used to help people in their decision, they do not make the decision themselves, but it is up to the user to have the final word in their choice. [7]

A DSS should composed of six components: decision models, interactive computer hardware and software, a data base, a database management system, graphical and other sophisticated displays, a modelling language that is 'user friendly', as defined by King (1983) [7]. Decision models contain the logic and algorithm, interactive hardware and software provide a method for users to input and receive information, and database is used to store all necessary information that the system may use.

Currently, DSSs are appliable in two main type of problem: Structured and semi-structured, ranging from simple day to day activities like chooisng foods to complex, and problematic issues such as marketing analysis and financial planning.

However, as useful as a DSS maybe, there are still many limitations that restrict the usage of DSS. One main aspect that DSS often fail to consider is the personality of their users. Humans are complex and depending on what answer the system provide, a group of people might be satisfied while others are not. Other common problems with DSS are data capture and collection, data integrity and security, management of DSSs, cost-effectiveness, standardization, group DSS, data that are not independent of spreadsheets, and dealing with unstructured-problems.

Still, given the usefulness of DSSs, with proper invesment and development, DSSs will become a key tools for many people in the future.

2.2.2 Interating Decision Support System

DSS technology has evolved significantly in the past few decades. While it has been considered as one of the most popular areas in information systems, it seems that DSS has experience a downward trend since 1990s because of many existing problems isolated DSSs still have to deal with, mainly, technology shifts from database to data warehouse and OLAP (On-Line Analysis Processing), from mainframe to client/server architecture, and from single user model to World Wide Web access; growing interconnection with more dynamic business

environment and intelligence that has been addressed by many other information systems such as ERP (enterprise resource planning), SCM (supply chain management) and CRM (customer relationship management); and increasing complexity of the decision situations which puts enormous cognitive workload on decision makers where a user needs to have considerable knowledge and must exercise initiative to perform decision-related tasks [8].

To solve this problem, it is crucial that DSS has to be expanded and evolved to compatible with modern technologies, and with it, a new concept has been born which is integrated DSSs (IDSS).

Rather than having only three basic components in form of a data management component, a model management component and a dialogue management component, IDSS makes improvement on the traditional concept in order to take accounts of the environment as well as the available technology. According to the authors, there are five perspectives to consider when integrating IDSS. Firstly, for data management component, IDSS include retrieval of information based on data warehouses. Secondly, for the model, it also has been expanded to include business model such as ERP. Thirdly, asides from existing function, others have been added to IDSS to provide better decision suggestion and data mining. Fourth, IDSS is built only to reduce handling variable in decisions. And finally, process units should inter-operate with each other as desired and support each other to result in better performance for the IDSS.

In a DSS, data integration is used to provide information for the decision making. Therefore, the integration must address two main concerns which are how data should reflect information, and how the system can deal with data from multiple sources. While it true that data and information integration for DSS has been widely researched and reported, nearly all published paper fails to state why they have been done.

Model integration, in DSS, defined as the act in which individually developed sub-models are logically combined to create a large, unified model. According to the author, in DSS, it is more beneficial to implement multiple models rather than a single large model. Through multi-model integration, DSS can be used to solve problems concerning diverse data sets resulting in more prominence effect and better result for users.

A DSS also need to consider process integration when being built. For the integration to be considered well-integrated, the components of the DSS must make similar assumption about the range of the constraints they recognise and respect.

Service integration is also considered a key aspect when building a DSS. Throught service integration, the system will be able to provide support to flexible function combinations in an

integrated decision support environment. Service integration and model integration are also naturally inter-related. While model integration provide foundation for the system, service integration help combines both provider's and user's viewpoint when using the DSS.

Presentation integration is also another aspect to consider. Presentation helps guide user and if is integrated correctly, should provide user with easier time when using the system through appearance and behaviour integration, as well as interaction-paradigm integration. Therefore, presentation should not be ignored when building a DSS.

IDSS classification based on implementation technologies

Through technologies used inside DSS, the IDSS can be categorized into four different types: knowledge-based system, data mining, intelligent agents enhanced, and web-enhanced DSS.

Knowledge-based systems (KBS), or expert systems, is used to perform tasks that often require human experts. With the name implies the core of KBS is its knowledge bases. Within IDSS, they are termed as model bases where hold the models of decision conditions and solutions. There are three approaches to integrating an KBS: rule-based reasoning, case-based reasoning, and hybrid reasoning or other reasoning method. In rule-based reasoning (RBR) approach, models of IDSS are represented as production rules, in form of IF, THEN, ELSE statements. The system also contains multiple models to response to many criteria such as technology, geometry, performance, economy and productivity. Case-based reasoning (CBR) is a solution to some limitation of RBR approach. Different from RBR, CBR can represent exceptions in for of cases, which helps generalize integration of information in the system, and offer more flexibility for the system. Aside from RBR and CBR, some other reasoning methods are also in used when developing IDSS, such as Bayesian Belief Network (BBN). BBN are probabilistic inference engines that can be used to reason under uncertainty using "what if" analysis which consider vagueness, ambiguity and uncertainty prevalent in real word systems.

In IDSS, data mining helps identify new information, interesting patterns and trends gathered in data to provide user with valuable intelligence. While data mining has been incorporated in IDSS since 1990s, there still are many optimizations that can be made. Data mining enhanced IDSS can be categorized into two types: general or specific application-oriented with most IDSS fall into the latter category [9]. By using data mining enhanced IDSS, many benefits can be achieved, such as removing of discrepancies and inconsistencies in the data used by the system and providing organisational knowledge-based decision making rather than relying on raw data, which further improve on the system's performance.

Intelligent agents enhanced IDSS is a category of IDSS that aim to reduce the passiveness of user-system interaction through intelligence, autonomy, pro-activeness, purposefulness, competence, reasoning capability, and interaction with environment and other agents. Through various research and publications, intelligent agents enhanced IDSS has changed greatly from its original concept of using only software angents to using multi-agent systems in combination with software computing to provide higher level of decision making and more comprehensive integration services. In intelligent agents enhanced IDSS, agents help performing integration and coordination of processes as well as specific tasks or processes to assist achieving solutions in dynamic and unpredictable environments. By using intelligent agents, IDSS can work as experts, servants and mentors and able to provide valid advice and criticisms to the users

Web-enhanced IDSS is term used for computerised system that delivers decision support information or decision support tool using a Web browser such as Chrome or Firefox. In web-enhanced IDSS, three types of service integration have been actively researched in supporting decision-making: interface-wrapping services, Multi-Agent System (MAS) services and Web services with web services currently the most adopted services out of all three [9]. While some limitations remained, using Web technology IDSS provide many benefits that are hard to ignore, such as global access of services, improvement on decision making framework, and promotion of consistent decision making on repetitive tasks.

2.2.3 Multiple Criteria Decision Analysis and Multiple Criteria Decision Support System

Multiple Criteria Decision Analysis

When dealing with complex situation, a correct decision is often not as clear as one might think. Choosing a correct decision involves not only weighting all advantages and disadvantages of all available options but also the context and situation currently at hand.

In tradition approach to analyse such situation using decision tree, to reach a conclusive answer, three main assumptions must be made [10]. These assumptions, however, are often difficult to hold by both Gilberto Montibeller and Alberto Franco. Therefore, both researchers recommend that Multiple Criteria Decision Analysis (MCDA) should be the main evaluation tools for making decision.

Rather than a single solution to each situation, MCDA provide a set of solutions depending on each scenario. By combining with tools for structuring objects, such as extensively cognitive maps, MCDA not only helps aligning the vision of the organisation with its strategic objectives, but also better scoping of its choices to consider.

Robustness is one key consideration when applying MCDA to multiple scenarios. Rather than maximize utility, it often more importance to have a strategy that has usable performance across all situations [10]. Robustness, as the authors suggest, is the use of min-regret, and in practice, is helpful for making choices when combining with visual aids.

Another rather important consideration one must make when using MCDA is the long-term consequences. Although simple solution has been given, it still is an issue that needs further research.

Multiple Criteria Decision Support System

Multiple Criteria Decision Support System (MCDSS) built on the concept of MCDA to fix one main weakness of traditional DSS model that is unstructured tasks. And, with the need to satisfy all factors when planning, it comes as no surprise that MCDSSs have become one of the most importance research topic in recent time [10].

Unstructured tasks, while have no definitive description, can be categorized into two types: object oriented or problem-solving process oriented. Object oriented unstructured tasks usually are difficult to know which factor affect the choice and are often difficult to predict the outcome of decision choice. Proble-solving process oriented unstructured tasks, on the other hand, are tasks that have no method for solving and have no clear criteria to consider. Unstructured tasks usually happen because of conflicting in objective, uncertain environment, and complex variable structure, and must be figured out before any model can be applied to analyse.

MCDSSs, same with traditional DSSs, are computer-based analytic tools that aid in the process of making decision, utilizing database and logical algorithm to solve and suggest a answer to problem one might encounter. Unlike traditional DSSs, MCDSSs have other features such as implementing Multiple Criteria Decision Making (MCDM) that help distinguish them which helps MCDSSs deal with unstructured tasks with conflicting objective better.

One way to implement MDSSs is through a Multiple Objective Decision Making approach (MODM) using Goal Programming (GP) Model. GP model help transform a decision model into a satisficing mode, which allow for better representation of human consideration to achieve the best possible result. GP models operate on a goal-constraint system.

By effectively integrating MCDSSs, decision makers are provided with valuable benefits, such as purpose-oriented decision making, information and judgment-based decision making, decision making for multiple objectives, efficient management of capital, energy, and materials, more systematic contingency management, and increased attention to group decision making and group decision support systems.

2.2.4 Multi-Criteria Decision Making Technique Using Weighted Methods

Another approach to making MCDSSs is through assinging weighted value for each criterion to determine their priorities. Therefore, determine the weight of each criterion can been seen as a key aspect as well as the main challenge when considering such approach. Throughout the history of DSS, various proposal had been made on how one can calculated the weight and applied for problem solving such as Analytic Hierarchy Process (AHP), weighted score method, VIKOR, TOPSIS, etc. However, these methods in practice are difficult to use, and often times, the decision makers have easier time assigning rank to each category rather than supplying the system with numerical relative weight [11]. Despite that, the validity of criteria weights obtained from different weighting methods cannot be ignored so as not to avoid any misuse of the MCDM models and getting reliable model results. MCDM methods can help to improve the quality of decisions by making the decision-making process more explicit, rational, and efficient.

Weights assigned to criteria in multi-criteria evaluation has both qualitative and quantitative data so as to make sure that the weight is taking into account for better and more accurate decision making. To limit the preference of the maker, a numerical scale of "1– 9" has been proposed to transform qualitative data into quantitative with "1" means "equals" and "9" means "extreme importance". Weights classification can also be grouped into three categories: Subjective, objective and integrated or combined weighting approach.

Subjective weight determination, such as Analytical Hierarchy analysis (AHP), is based on expert opinion, and in order to get the subjective judgments, analysts normally present the decision makers a set of questions in the process. Objective weighting methods, on the other hand, are derived from information gathered from each criterion using a mathematical function to compute the weights without the interference of the decision maker. Examples of this method are entropy method, mean weight, standard deviation, statistical variance procedure, and criteria importance through inter-criteria (CRITIC). And finally, Integrated weighting method or combines weighting methods are derived from both subjective and objective information on

criteria weights. Integrated weighting method determines the weights of each categorise by solving a mathematical model and takes into consideration both subjective and objective factors in order to overcomes the shortages which occur in either a subjective or an objective approach. Notable example of this approach includes Multiplication and Additive synthesis, Optimal weighting based on sum of squares, and Optimal weighting based on relational coefficient of graduation.

Next, let analyse some algorithm deployed by both the Subjective and Objective weighting methods.

The point allocation method: One of the simplest methods in Subjective weighting approach used to determine the weight of each criterion, point allocation method has decision maker allocates certain numerical value from a predetermined number to the criteria based on its priority. The greater the numeric value that categorise has, the greater is relative importance is. One example of this method is a Project Portfolio Management (PPM) likes Microsoft Project Online, which is widely used by company to evaluate and score each ongoing projects before funding them

The ranking method: Is also considered one of the simplest Subjective approaches to assign criteria weights. There are three approaches to ranking criteria, including rank sum (RS), rank expotent (RE), and rank reciprocal (RR).

In *rank sum (RS),* criteria's weight is computed from the individual ranks normalized by dividing the sum of the rank via expression:

$$w_j(RS) = \frac{n - p_j + 1}{\sum_{k=1}^{n} n - p_k + 1}$$

Where p: rank of j-th criterion, j =1, 2, …, n

Rank *expotent (RE)* method applied a simlimar equation to rank sum, with one difference that is the value is raised to an exponential of a parameter p which may be estimated by a decision maker as a result of the most important criterion.

Do math

$$w_j\ (RE) = \frac{(n - p_j + 1)^p}{\sum_{k=1}^{n} (n - p_k + 1)^p}$$

Where p: rank of j-th criterion, j =1, 2, …, n

Finally, *reciprocal (or inverse) weights (RR)* method uses the normalized reciprocal of the criterion rank

$$w_j\ (RR) = \frac{1/p_j}{\sum_{k=1}^{n}(1/p_k)}$$

Where p: rank of j-th criterion, j =1, 2, …, n

One example of this method is PROMETHEE or Preference Ranking Organization METHod for Enrichment Evaluation, which is used by city planers or authorities for energy planning, environmental impact assessment, transportation planning, infrastructure project ranking and policy analysis.

Entropy Method: is an objective-based approach used in weighting criteria in a given problem. The entropy works based on a predefined decision matrix. For each critetion, the relative importance of its is calculated by the equation:

$$E_j = -[\sum_{i=1}^{m} p_{ij} \ln(p_{ij})] / \ln(m)$$

Where j = 1, 2…, n and i = 1, 2, …, m

The p_{ij} form the normalized decision matrix and is given by equation

$$p_{ij} = \frac{r_{ij}}{\sum_{i=1}^{m} r_{ij}};\ i = 1, 2, \dots, n;\ j = 1$$

Where r_{ij} is an element of the decision matrix, k is a constant of the entropy equation and E_j as the information entropy value for jth criteria.

And, the weight w is obtained through expression:

$$w_j = \frac{1 - E_j}{\sum_{j=i}^{n}(1 - E_j)}; j = 1, 2, \dots, n$$

Where $(1 – E_j)$ is the degree of diversity of the information involved in the outcomes of the jth criterion.

PyMCDM, Python Multi-Criteria Decision Making library, is an example of entropy method applied to multi-criteria support system, and is often used by data scientists and researchers to solve MCDM (Multi-Criteria Decision Making) problems

Standard Deviation Method: An objective approach that determines the weights of the criteria in terms of their standard deviations using the expression in these equations:

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{m}[r_{ij} - \overline{r_j}]^2}{m}}$$; i = 1, ..., m; j = 1, …, n

And:

$$w_j = \frac{\sigma_j}{\sum_{j-1}^{n} \sigma_j}$$

CHAPTER 3

METHODOLOGY

3.1. Overview

Chapter 3, Methodology will analyze the topic of this thesis in detail. The aim of this thesis is to develop a Decision Support System (DSS) to provide an option for users who want to go on holiday but do not know where to go yet. Therefore, the main focus of this thesis is to create a simple, yet efficient algorithm that will be able to handle user's preferences and return a sastifactory location for them.

This Chapter will contain two major sections with the former being User Requirement Analysis, where question about this thesis purpose, target audience, as well as functional and non-functional requirements will be explain. The latter section, called System Design, will break down the proposed system and provide context on choices made in designing database design and User Interfac, as well as the main algorithm used.

3.2. Thesis Procedure

This section will describe the steps taken during the development process of the thesis. For this project, the steps taken are:

Requirements analysis and design: In this step, I will analyse the possible input that user may need when using the system such as budget, wanted scenery, as well as activities needed in the place where they are travelling. From this, I will be able to choose what function will be needed for the website, focusing on the main algorithm, and branching out on supporting function such as place searching and viewing usage history. With the function ready, I will also know what my database will contain, and what attributes each table in my database need.

Choosing technology: For this project, React.js is chosen for building the front-end section of my website because of its large community and extensive documentation combining with ease of use. Node.js, together with Express.js will be my main choice to provide structure and tools for building a fast and efficient backend side of my project. MySQL, and popular choice for relational database, and my choice for the database with because of my need for a consistent and highly connected database.

Design and Implementing: In this step, both frontend and backend section of the project will begin to be built. For the frontend side, the design will focus on making a user-friendly interface, which emphasize accessibility. As for the backend side, the model of the database will first be implemented and the populated. Then, the main algorithm for finding the place will be prioritizing first, and after it is complete, other function will be considered.

Testing, Evaluation, and Optimizing: After the project is complete, this step will occur. In this step, the project will be test for bugs using test cases as well as user demo. The project will then be evaluated on aspects such as accuracy, user's satisfactory, as well as the website's security. And finally, any change make to the project will be implement, and the project will also be optimized for faster and smoother experience.

Deployment and Future Planning: Finally, the project will be deployed and make available for public use. The website will be maintained and any bug discover will be investigated and patched shortly after. This period is also the time for making document and recalling what challenge occur during development process as well as thinking of possible improvement for website.

3.3. User requirement analysis

3.3.1. Purpose:

The website main goal is to recommend users a location when they input all their criterion into the website. To be able to make it easier to imagine, this section will analyze user requirement of the project. User requirement analysis is a crucial aspect in building this project. Through careful consideration of what to be expected, the chances of encounter problems during development is greatly reduced.

3.3.2. Functional Requirement

Functional requirements describe what actions can the system do, or in other words, the features and functionality that the system is built for.

For this Decision Support System, the main functionality is to return a recommended place for the user to go on holiday, therefore, these are the functional requirement that the system must have:

- **Preference Input:** User must be able to provide the system with what they wish to have in the recommended location so that the system's algorithm is able to calculate and propose a suitable place.

- **Place Recommending:** The system must be able to see what the user want to have, and search through its database and find a suitable place to suggest to the user with all necessary justification. The system should return a list of places ranked by how similar they are to user's desired location.
- **Place Viewing:** The user should be able to look at the information of the recommended location, as well as search for any place they separately search for using the provided search bar in the system.
- **Place Reviewing:** Users are able to leave a review for a location once they confirm that they have gone to that place at least one, and others are able to view those review when looking through the recommended list.

3.3.3. Non-Functional Requirement

Non-functional requirements address how well the system do its function, rather than what the system is capable of doing. Non-functional requirements are crucial as them control the quality of the system, affecting the system overall succes.

In this system, non-function requirements that must be considered are:

- **Performance:** The system is expected to work smoothly. All pages when visited must be fully loaded within 3 seconds. And, when the user uses the system for recommendation, the result is expected to be returned within reasonable time and have strong compatibility with user's provided criterion.
- **Accessibility:** The website must be intuitive to use, meaning that first time users are able to navigate the website with no problem. All information provided to user must also be easy to read and are placed neatly on user's screen without overloading the page.
- **Availability / Reliability:** The website must be accessible at any time minus occasional shutdown for update and maintenance. Any problems occur when sudden shutdown of the system must be accounted for appropriately
- **Maintainability:** The website's code must be easy to read and maintain. Any bugs and fault occur must be easily identified and fixed within reasonable time. Adequate usage of comment is being employed so that the code is easy to understand and transferable.
- **Security:** All sensitive user information must be protected. The system should also contain no possible exploit, and any newly discovered exploit must be handled quickly and properly to prevent malicious action toward the system's integrity.

3.3.4. Use case diagram

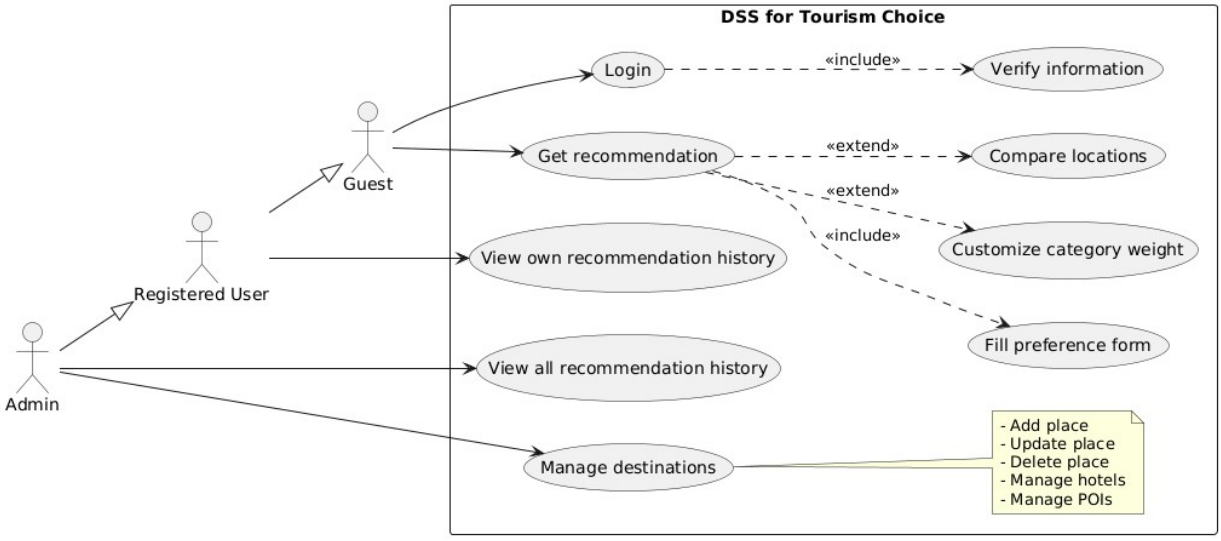

Figure 3.1. Use case diagram of DSS for tourism choice

Use Case Document:

UC ID	Use Case Name	Primary Actor	Description	Includes	Extends
UC-01	Login	Guest, Registered User, Admin	Allows users to log into the system	Verify information	-
UC-02	Verify information	System	Validates login credentials	-	-
UC-03	Get recommendation	Guest, Registered User	Generates destination recommendations based on user preferences	Fill preference form, Customize category weight	Compare locations

UC-04	Fill preference form	Guest, Registered User	User enters travel preferences	-	-
UC-05	Customize category weight	Guest, Registered User	User adjusts preference importance	-	-
UC-06	Compare locations	System	Compares destinations	-	-
UC-07	View own recommendation history	Registered User	Views personal recommendation history	-	-
UC-08	View all recommendation history	Admin	Views all system usage histories	-	-
UC-09	Manage destinations	Admin	Manages destination data	Add place, Update place, Delete place, Manage hotels, Manage POIs	-

Table 3.1: Use Case Document

Description:

This use case diagram illustrates all the interaction actors can have inside the DSS.

Users, both registered and unregistered, are able to access the main function of the system that is recommending through two actions, filling the preference form with all their needs, and then click the submit button and see the result. In the imputing preference steps,

users are able to make the system prioritize some criterion more than others through the extra action of customizing category's weight. All the weight of each category will start out at 1, meaning all criterions are treated as equals, and will be change from 1 to 10, with 10 being the most important.

Both types of users are also able to search for the name of any location inside the database through view city information. From there, any hotels, points of interest that are related to the city can also be view.

Verified users, after getting the recommended result, have three actions that they are able to take. First, they can compare the first-place location with other location inside the recommended list. Second, they will be able to favourite a location and save it for later viewing. Thirdly and finally, they are able to leave review of the location to share it with other users. Additionally, logged in users are also able to view their history tab, which includes their previous preference inputs, as well as the recommend lists.

Admins play a crucial role in the system. They manage not only the user data, but also in charge of adding, and updating location information so that the information is up to date.

3.4. Sequence Diagram

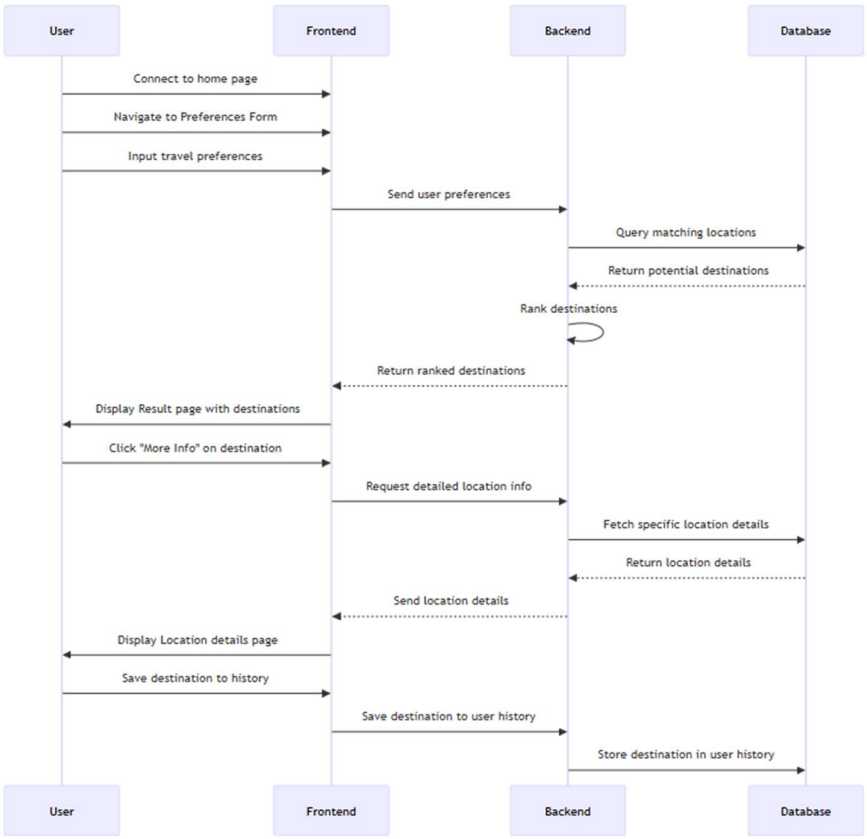

Figure 3.2. Decision Support System Sequence Diagram

The sequence diagram illustrates the complete workflow of the decision support system, showing how the full-stack system work to fetch and display data to users.

The process begins when the user connects to the home page and navigates to the preferences form, where they are able to input their travel preferences, which can include budget, food choices, etc. These preferences, will be passed to Backend server to process in form of JSON. Backend after receiving user input, will call database to fetch all locations and try to match each location with user preferences. After a list of top destination is formed, Backend will return data, also in form of JSON, back to Frontend server to display for user to see. From here, user is able to look more into each location through “More Information” button, or they can save it back into their history to view later if they are logged in.

3.5. System Design

This section provides the basic blueprint for building the website through breaking down its core objective and translate it into diagrams and model that helps illustrate the system for development to better understand and work with.

There will be three subsections in this sectio, each is used to focus on a different aspect of the system design. First, database design will show the models used inside the system, and explain what each attribute of each model as well as the relationship between them. Next, User Interface will explain the design choice of the page. Finally, Algorithm Design explain the main algorithm used by the system to propose an option to the user.

3.5.1. System Architecture:

In this project, I will follow the Monolith Architecture to develop my website.

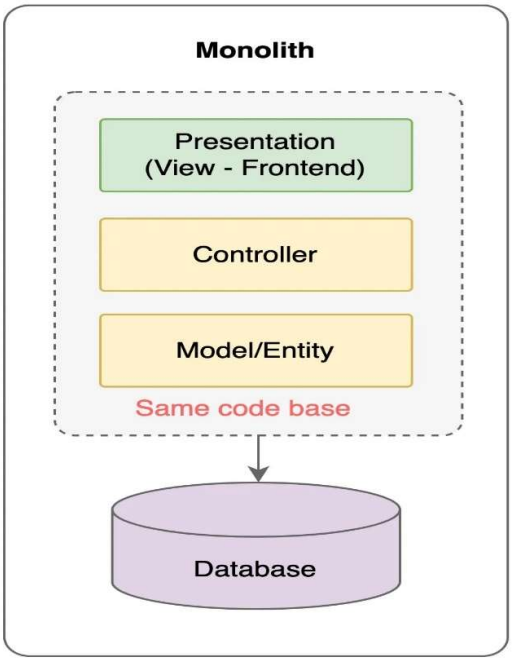

Figure 3.3. Simple illustration of Monolith Architecture

Monolith Architecture: is a software design methodology that combines all of an application's components into a single, inseparable unit [12]. Under this architecture, the user interface, business logic, and data access layers are all created, put into use, and maintained as one, unified unit.

Monolith Architecture exhibits several defining characteristics:

- **Single Codebase:** The program is simpler to manage and implement since all of its components are created and maintained in a single codebase.
- **Tight Coupling**: The architecture's components are closely linked, rely on one another, and frequently exchange resources and data directly.
- **Shared Memory:** Monolithic applications typically share the same memory space, allowing components to communicate efficiently without the need for network overhead.
- **Centralized Database:** Data storage is centralized within the application, typically using a single database instance for all data storage needs.
- **Layered Structure:** The structure of monolithic systems is frequently layered, with separate layers for data access, business logic, and presentation. This might result in dependencies across layers even while it separates issues.
- **Limited Scalability:** Because the entire application must be scaled at once, scaling a monolithic application can be difficult and frequently leads to inefficiencies and higher resource usage.

There are 6 components that are crucial to Monolith Architecture, those are:

1. **User Interface (UI):** Provides users with information and gathering feedback through the use of buttons, forms, and other interactive elements.
2. **Application Logic:** Contains the application's main functionality.
3. **Data Access Layer:** Manages interactions with the database or other data storage systems.
4. **Database:** Stores the application's data in a structured format. It can be relational, NoSQL, or another type of database, depending on the requirements of the application.
5. **External Dependencies:** Provides additional functionality or system integration such as third-party APIs communication, and authentications
6. **Middleware:** Manages cross-cutting issues like logging, security, or performance monitoring or help with communication between various application components may be included in monolithic systems.

Through the use of Monolith Architecture, I can take advantages of these benefits that the architecture provides:

- **Simplicity of development:** All source code is located in one place which can be quickly understood.
- **Simplicity of debugging:** The flow of a request can easily be followed and any issue can easily be located.
- **Simplicity of testing:** Only one service to test and without any dependencies makes everything clearer.

- **Simplicity of deployment:** Only one deployment unit should be deployed. Everything exists and changes in one place.
- **Simplicity of application evolution:** New features can be added easily and those are able to directly use the system's current database.

3.5.2. Design Pattern

To have a easier time managing and maintain the codebase, this project will follow MVC design pattern.

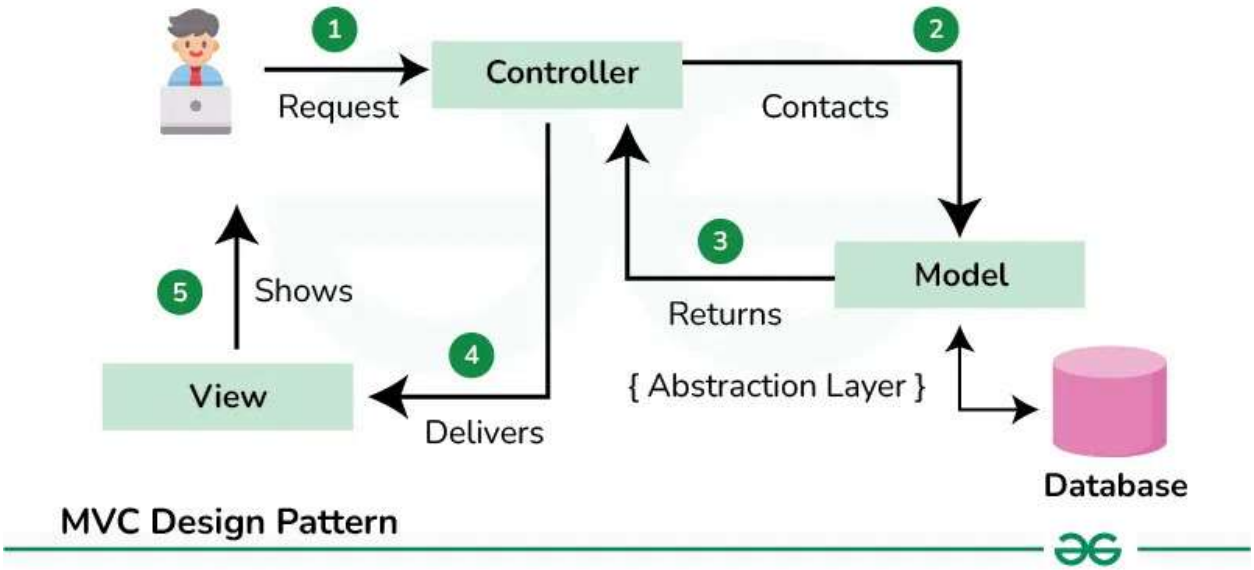

Figure 3.4. Illustration of MVC design pattern

MVC, or Model View Controller, like the name suggests, breaks an application into three parts: The Model, the View, and the Controller. MVC makes it easier for updating or fixing each component separately without breaking the whole project [13]. Not only does MVC helps keep the project organized, but it also improves the quality of the software.

In MVC, the Model component is responsible for managing the application's data, processing business rules, and responding to requests for information from both the View, and the Controller component. The View component displays the data from the Model to the user and sends user inputs to the Controller. The View component only interacts with Model components indirectly via the Controller component. Finally, the Controller component handles user input and updates the Model accordingly and updates the View to reflect changes in the

Model. Controller component contains application logic for handling user's authentication, as well as handling algorithm responsible for recommending to the user a suitable location.

3.5.3. Datadase design

a. Database model

For this project, 8 models are created to implement all features required from the user analysis.

Table	Description	PK	Attributes (excluding PK/FK)	Rules & Constraints	Relationships
User	Stores registered user account information.	id	username, email, hashed_password	username is unique; password length ≥ 6; valid email format	**User → Result:** Optional (1:M)
Place	Stores travel destination data used for recommendation	place_id	place_name, country, money_unit, avg_cost_per_day, place_description, place_img, tourist_density, tags	tags stored as JSON; soft delete enabled (paranoid)	• **Place → POI:** Optional (1:M) • **Place → Hotel:** Optional (1:M) • **Place → Weather_Conditions:** Mandatory (1:1) • **Place → Result:** Optional (1:M)
Hotel	Stores hotel information for a destination.	hotel_id	hotel_name, hotel_description, hotel_img, star_rating, address	star_rating is numeric	**Hotel → Place:** Mandatory (M:1)

Weather_Conditions	Stores weather data for a destination.	weather_id	month, avg_temp, humidity, weather_type	one weather record per place per month (logical constraint)	**Weather_Cons → Place:** Mandatory (M:1)
POI (Point of Interest)	Stores points of interest within a destination.	poi_id	poi_name, poi_description, poi_img, opening_hour, closing_hour, entry_fee, additional_fee, rating, tags, timezone	rating is numeric	• **POI → Place:** Mandatory (M:1) • **POI → POI_Activity:** Optional (1:M)
Activity	Stores activity types associated with POIs.	activity_id	activity_name, activity_description	activity_name is unique	**Activity → POI_Activity:** Optional (1:M)
POI_Activity	Links POIs with activities (junction table).	poi_act_id	_	resolves many-to-many relationship	• **POI_Activity → POI:** Mandatory (M:1) • **POI_Activity → Activity:** Mandatory (M:1)
Result	Stores recommendation results for users and places.	result_id	preferences (JSON)	preferences stored as JSON	**Result → User:** Optional (M:1) **Result → Place:** Mandatory (M:1)

Table 3.2: Use Case Detail

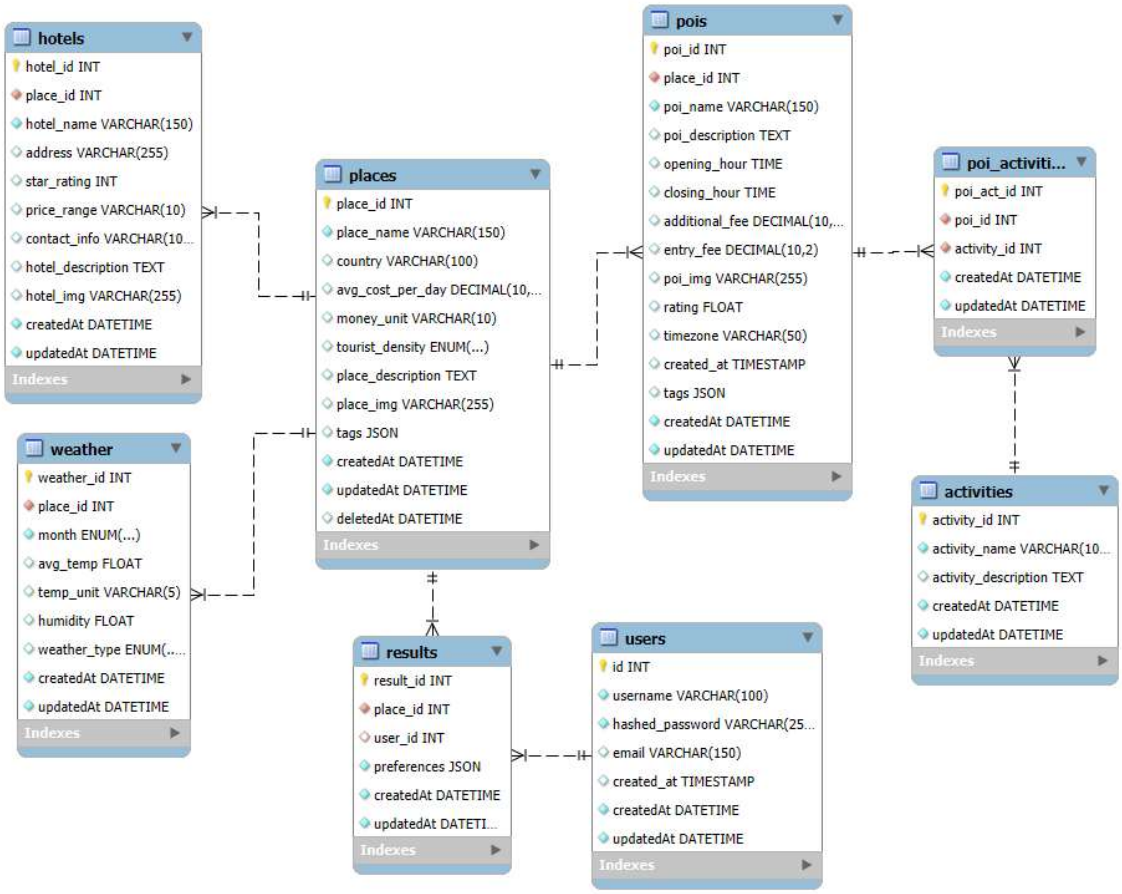

Figure 3.5. Models used in DSS for tourism choice

b. Relationship

In this website, the relationship between each models are as follow:

- **User – User_Preferences:** One to Many – Users are able to create a lot of recommendation request at the server, but each request only belongs to one user.
- **Place – Hotels:** One to Many – A location can have many hotels, ranging from cheap 2 to 3 star hotels, to luxurious ones with 4 and 5 stars. These hotels, however, only belongs to one place. Any other hotels either with the same name, or from the same company, but is located at a different location will be treated as a different entry to the database.
- **Place – POI:** One to Many – POIs, or points of interest, are noticeable location or activities that user can go to or do when visiting a place. Therefore, each place may have a lot of POIs, but each POI will only belong to that location.

- **POI – POI_Activities:** One to Many – this relationship illustrates all the activities that one can do inside a POI, thus, a singe POI will contain multiple POI_Activities, but each activitie will belong to only one POI
- **Place – Weather_Condition:** One to Many: This relationship helps illustrate the weather pattern of a location if the users are interested in it. Because each pattern belongs to a month, this relationship of place and weather becomes one to many so that information on each month is translated clearly inside the database for easier time of searching information

3.5.4. User Interface (UI) design

UI design focus on the look of the website in the eyes of a user. Good UI design can enhance user satisfaction, as well as increasing the accessibility of the website.

For this system, the main goal of UI design is to focus on creating a assessible website that is intuitive to use. The website is divided in to multiple page, with a separated page for information about the current user. The system has a search bar inside navigation bar so that user can search for the name of the location for viewing their information. The main form used a simple design so that there will be no confusion when used. Textbox, checkbox, and selection options, as well as slider are used to contain all user input, and a submit button is located at the bottom of the form for user to submit and get their recommended location.

3.5.5. Algorithm design

Scoring algorithm

To recommend to the user the suitable place depending on the preferences of the user, a point-based system will be used by the decision support system. After the point of each location have been calculated, the system will rank them based on the highest value, and the top location as well as a list of upward of five other places will be propose to the user if they choose to view alternative and compare the result with other location.

To handle this task, each location will be mapped with a score that is calculated by a scoring algorithm. For each criterion inside the preference form, the algorithm will use a difference logic for handling it. Each criterion in the preference sheet will be evaluate and score from zero to one hundred depending on how well each location's feature match the user's need

For budget criteria: each location will be scored based on how close the cost of going there is compared to the user's budget. Any location that has higher cost will have lower score compared to those that stays in budget. The higher the difference the between the cost and the

wanted budget is, the lower the score of that place will be. Depending on if normal or luxury trip option is chosen by user, a minimal cost value will also be employed, and any location that is under the cost will also has it score lowered accordingly.

- **For scenery criteria**: the system will grant full score for any location that have the wished scenery. As for the other, no score will be granted.
- **For temperature and weather condition:** the system will calculate this criteria score based on the when you want to go. From the provided date, the system will be able to check the pattern of temperature the place has, and closer the temperature is to the wanted one, the higher the score will be. Weather type will also have its own score, with any location having the right condition will have the full score, and other will
- **For activities criteria**: the system will search through the list of Point of Interest corresponding to each location and see if it has the needed activity and calculate the score of the location accordingly.

After the point of each criteria is calculated, they will be multiply with their respective weight value choosen by the user, and then got add together to form the final score of each location that will be used by the system when choosing where to recommend to the user.

Sorting and Choosing Algorithm:

To recommend only the top location for the user, the score of each location will be sorted first. To sort the score, the system use Javascript built-in sorting function: "Array.prototype.sort()". To get the correct sort, a comparison function is used. The comparision function will have the shorten form: "(a, b) => b – a" where a, and b are the element from the array that this function used to sort. Together, they form the full algorithm used to sort all the location before system recommending to user: "array.sort((a, b) => b – a)". This sorting function works by checking the value provided by the comparision function, if the value is a positive number, b element will be placed before a. If value is a negative number, the reverse will have happened and a will be placed before b. And, if the value is equals to 0, no change in placement will happened. This sorting algorithm achieves an average time complexity of O(n log n) [14].

```
function sort(array, compareFunction OPTIONAL)
  if compareFunction is provided and is not a function
    THROW TypeError
```

```
let length = array.length
if length < 2
   return array

let items = EMPTY LIST
for index FROM 0 TO length - 1
   if index exists in array
      add array[index] to items

function Compare(a, b)
   if compareFunction exists
      return ToNumber(compareFunction(a, b))
   else
      let A = ToString(a)
      let B = ToString(b)
      if A < B return -1
      if A > B return 1
      return 0

*Sort items using Compare

let writeIndex = 0
FOR EACH item IN items
   array[writeIndex] = item
   writeIndex++

WHILE writeIndex < length
   delete array[writeIndex]
   writeIndex++

return array
```

Table 3.3: sort() Pseudo Code

As for choosing the list of recommended list, the system will use the provided function: "Array.prototype.slice(0, n)" to return the list of item with n is configured by the the developer [15].

```
function slice(array, start OPTIONAL, end OPTIONAL)

   let length = ToLength(array.length)

   if start is undefined
      let startIndex = 0
   else
      let startIndex = ToInteger(start)
      if startIndex < 0
         startIndex = MAX(length + startIndex, 0)
      else
         startIndex = MIN(startIndex, length)

   if end is undefined
      let endIndex = length
   else
      let endIndex = ToInteger(end)
      if endIndex < 0
         endIndex = MAX(length + endIndex, 0)
      else
         endIndex = MIN(endIndex, length)

   let result = new empty array

   let resultIndex = 0
   let currentIndex = startIndex

   WHILE currentIndex < endIndex
      if currentIndex exists in array
         result[resultIndex] = array[currentIndex]
      // else leave hole (sparse array behavior)

      resultIndex++
      currentIndex++

   return result
```

Table 3.4: slice() Pseudo Code

CHAPTER 4

IMPLEMENT AND RESULTS

4.1. Implement

4.1.1. Overview

The implementation phase of the Decision Support System (DSS) for Tourism involves transforming the design of the system, as described in previous chapters, into a fully functioning, interactive website that can recommend travel destinations based on user preferences input. This chapter discusses the key technologies, tools, and methods used to develop the system, as well as the overall structure of the implementation.

The objective of the implementation phase is to create a working DSS that can generate accurate and personalized tourism recommendations based on the user's input. The system aims to take the preferences provided by users and use a point-based algorithm to generate tailored travel suggestions. These recommendations are based also on the weights assigned to each of the user's input categories, ensuring that the recommendations are personalized and align with the user's priorities.

The system is developed with a full-stack approach, using javascript as it main language for both Frontend and Backend. For frontend, React.js is used to build the dynamic and responsive user interface. React helps in rendering real-time data and seamlessly updating the UI when user preferences change. The server-side component of the application is powered by Express.js, a flexible and minimal web framework for Node.js, which handles routing, HTTP requests, and acts as the middleware between the frontend and the database. The data used inside the system is stored in MySQL, a relational database that is choosen due to its robustness, scalability, and ability to handle complex queries efficiently.

The recommendation system relies on a point-based algorithm. This algorithm processes the inputs, calculates scores for various destinations, and sorts the results according to the user's preferences. The system then queries the database for potential matches based on additional weighted scores that user can assign to each categories of the form.

Finally, the website also uses JWT, or JSON Web Token for user authentication.

4.1.2. Code Implementation

a. Project Structure

- **Frontend**: Frontend folder is structure so that it ensures modularity and scalability, maintaining readability of the code. Src folder is the source folder and main section of Frontend. Inside of it, components are folder for holding reusable components to be used across the project. Hooks folder contains custom React hooks foe handling numerous task inside the project. Utils folder holds reusable function that can be called without needing to be created again inside the file, and page contain the UI design of every page inside Decision Support System project. When Frontend is ran, main.jsx will be accessed first, before going into anyother files.

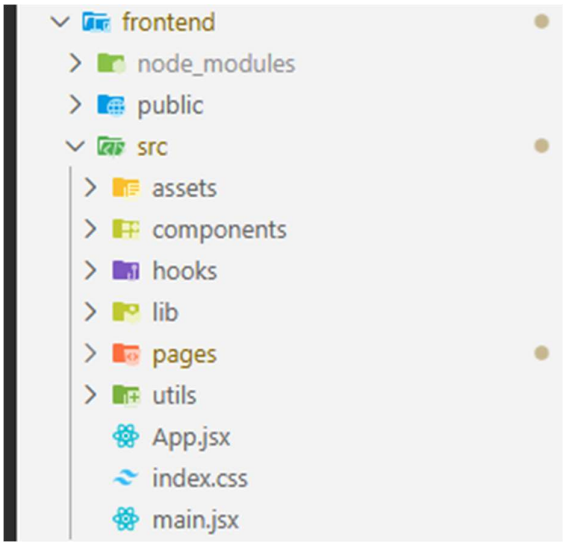

Figure 4.1. Frontend Folder Structure

- **Backend**: BE structure makes it easy to maintain, scale, and understand the backend application as it grows. Config folder is used to contain configuration files related to application's settings, such as connection setting for MySQL database. Controllers folder is responsible for handling the logic of various routes. Each controller typically defines the logic for processing HTTP requests. Models folder define the structure of the system's data. Model is used in combination with Sequelize ORM to write query and fetch data from server. he routes folder contains route handlers that connect HTTP requests to the appropriate controller functions. Each route defines a URL endpoint and which controller method should handle the request for that endpoint. And finally, the utils folder contains helper functions that are used across the application, which

include: helper functions, error handling, and authentication. The server.js file is the entry point for this BE server

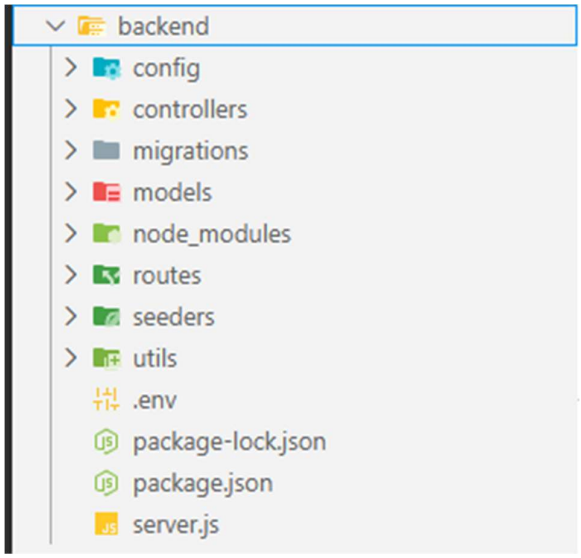

Figure 4.2 Backend Folder Structure

b. Recommendation Algorithm Implementation

This algorithm is contained inside createNewResult() function of BE server. This function is designed to take in and process user inputs, and calculated various scores for each destination inside the database, before calculating total by summing up all value and multiply with weight value, and then sort to get the list ranked by the highest score

Main algorithm:

```
function createNewResult(request, response):
  Try:
    # Extract relevant data from the request body
    Extract userBudget, totalTravelDays, user_scenery_requirement,
        user_activity_preference, user_weather_preference, travel_month
    If userBudget or totalTravelDays is not a valid number, set them to 0
    If travel_month is missing and user_weather_preference contains values,
      return an error response

    # Fetch all places from the database
    Place.findAll()

    # Initialize an empty list to store scores for each place
    Initialize empty list `scorings`

    # Iterate over each place to calculate its score
```

```
        For each place in places:
            Calculate budget_score using place.calculatedBudgetPoint() with userBudget
and totalTravelDays
            Calculate scenery_score using place.calculatedSceneryPoint() with
user_scenery_requirement
            Calculate activity_score using place.calculatedActivityPoint() with
user_activity_preference
            Calculate weather_score using place.calculatedWeatherPoint() with
user_weather_preference and travel_month


        # Sort the scorings list in descending order based on total_score
        Sort scorings by total_score in descending order

        # Send the response with the calculated scores and user preferences
        Send response with status "success" containing:
            - user_preferences (userBudget, totalTravelDays, user_scenery_requirement,
user_activity_preference, user_weather_preference, travel_month)
            - content (sorted scorings list)

    Catch error:
        # Handle any errors that occur during the execution
        Log error details (console.error)
        Return error message
```

Table 4.1 createNewResult() Pseudo Code

createNewResult() function is called when users submit the form containing their preferences. This function will take in six parameters, consisting of userBudget, totalTravelDays, user_scenery_requirement, user_activity_preferences, user_weather_preference, and travel_month to get an understanding of user preferences, an additional option of weight value is also included which will be used later for the calculating the score of each place. Place.findAll() function is then called to fetch the list of all destinations and their attributes. Using map() function, the system then iterates over each item inside the destination list, calling place.calculatedBudgetPoint(), calculatedSceneryPoint(), calculatedActivityPoint(), and calculatedWeatherPoint() to handle calculate the base score of each aspect, before multiplying each of them with their weight value , and finally sum them up to get the final scoring of each place. Each scoring will be stored inside an array, and will be sorted using javascript sort() function, before sending it back to the Frontend to process and display.

calculateBudgetPoint() function:

```
function calculatedBudgetPoint(userBudget, totalTravelDays):
   tolerance = 0.10          // 10% over-budget tolerance
   underTolerance = 0.50      // 50% under-budget tolerance
   bias = 0.05               // slight penalty/bonus shift

   // Default case if inputs missing
   if userBudget <= 0 OR totalTravelDays <= 0:
      return score = 0.5, totalBudgetNeeded = -1
   totalBudgetNeeded = avg_cost_per_day * totalTravelDays
   difference = absolute(userBudget - totalBudgetNeeded)
   percentageDiff = difference / totalBudgetNeeded

   // Case 1: User is over budget AND difference exceeds tolerance
   if userBudget < totalBudgetNeeded AND percentageDiff > tolerance:
      return score = 0, totalBudgetNeeded

   // Case 2: User is over budget but within tolerance
   if userBudget < totalBudgetNeeded:
      ratio = percentageDiff / tolerance
      score = 0.5 * (1 - (Math.pow(ratio, 2))
      score = score - (bias * 0.5)
      score = max(score, 0)
      return score (clamped 0–1), totalBudgetNeeded
   // Case 3: User is under budget
   // If far under budget → max score
   if percentageDiff > underTolerance:
      return score = 1, totalBudgetNeeded

   // Otherwise, Case 4: User is under budget but within 50%
   ratio = percentageDiff / underTolerance
   score = 0.5 + 0.5 * sqrt(ratio)
   score = min(score, 1)

   return score (clamped 0–1), totalBudgetNeeded
```

Table 4.2 calculatedBudgetPoint() Pseudo Code

calculatedBudgetPoint() takes in two parameters, userBudget and totalTravelDay to calculate how well a place do it the budget aspect on a scale of 0 to 1 with 1 is the best match. Inside the function, three constants are created, tolerance for overbudget, underTolerance for underbudget, as well as bias which will be used to make it so that the system reward places that are underbudget more than those that are overbudget. After checking the validity of the parameters, the system will calculate percentageDiff

of userbudget and needed budget. From here, depending on the percentageDiff, four cases will be occurred.

- Case 1, the place is overbudget and exceed the acceptable tolerance, the budget score in this case will be immediately set to 0, marking this place unsuitable for user preferences.
- Case 2, the place is overbudget but is within accepted range, the place's scoring will be calculated using a smooth quadratic curve, and a small penalty is applied to the final score.
- Case 3, if the needed budget is way smaller than the user budget (smaller than 50%), the scoring of that place will be immediately set to 1, or most suitable.
- Case 4, user have enough budget to affor going to the place, the place's scoring will also be calculated using a smooth quadratic curve, with a small reward applied to the final score at the end.

calculatedSceneryPoint() function:

```
function calculatedSceneryPoint(user_scenery_preferences):
   // 1. Handle missing or invalid user preferences
   if user_scenery_preferences is not a non-empty list:
      return 0.5

   // 2. Retrieve place tags
   placeTags = getDataValue("tags")

   // 3. Normalize user preferences and place tags to lowercase
   userPrefsLower = lowercase + trimmed version of each user preference
   tagsLower      = lowercase + trimmed version of each place tag

   // 4. Count unique matching tags
   matchedSet = empty set

   for each pref in userPrefsLower:
      if tagsLower contains pref:
         add pref to matchedSet

   matchesCount = size of matchedSet

   // 5. Determine score based on match count
   if matchesCount == 0:
      return 0

   if matchesCount == 1:
```

```
    return 0.5

// 6. Multiple matches → scale upward with bias
bias = 0.1
score = 0.5 + bias * (matchesCount - 1)

if score > 1:
   score = 1

// 7. Return final rounded score

return round(score to 3 decimal places)
```

Table 4.3 calculatedSceneryPoint() Pseudo Code

calculatedSceneryPoint() is used to compute the scoring of each location using user input as preference. Upon running, this function will first normalize the tags data of each place by converting them all to lower cases. Next, the function will count the unique tags that match user preferences, and store it inside an array. From here, depending on the length of the array, three behaviors of the function may occur.

- First, if the length of the array is 0, the place is deemed unsuitable to user, and the function will return the score of 0.
- Second, if the length of the array is exactly 1, the function will return the default score of 0.5, the same as when user have no scenery preferences
- Third and finally, for each additional tags that place matched, the score will be scale upward till it is capped at 1.

calculatedActivityPoint() function:

```
async function calculatedActivityPoint(user_activity_preference):
   // 1. Handle missing or empty user preferences
   if user_activity_preference is not a non-empty list:
      return score = 0.5,
           place_activity_tags = undefined

   // 2. Normalize user preferences
   prefsLower = for each preference:
              convert to string → lowercase → trim

   // 3. Query database for all POIs and activities at this place
   pois = query POI table where place_id = this.place_id
        and include:
            POI_Activity
```

```
            -> Activity (with activity_name field)

// 4. Extract all activity names from all POIs
allActivityNames = flatten list of:
   for each POI:
      for each POI_Activity:
         take the related Activity.activity_name
         convert to lowercase and trim
      if none, use empty list

// 5. Count unique matches between user prefs and activity names
matchedSet = empty set
for each pref in prefsLower:
   if pref exists in allActivityNames:
      add pref to matchedSet
matchesCount = size of matchedSet

// 6. Score based on match count
if matchesCount == 0:
   return score = 0,
        place_activity_tags = allActivityNames
if matchesCount == 1:
   return score = 0.5,
        place_activity_tags = allActivityNames
// More than one match → scale with bias
bias = 0.1
score = 0.5 + bias * (matchesCount - 1)
if score > 1:
   score = 1

// 7. Return final result
return score rounded to 2 decimals,
     place_activity_tags = allActivityNames
```

Table 4.4. calculatedActivityPoint() Pseudo Code

calculatedActivityPoint() evaluates how well a place matches the user's preferred activities, based on the POIs (Points of Interest) inside each place on the scale of 0 to 1. The function first takes a user wanted activity list and normalizes it by converting all activities to lowercase. Then, the database is called to fetch all POIs inside the location by matching their place_id foreign key with this location's place_id. From the POIs list, the same logic of calculatedSceneryPoint() function is used, where each POI's activity list will be looked into, and each unique activity will be taken out to store inside an array. By using the length of unique activities array, function will return the score of 0

for no match, 0.5 as default for 1 match, and 0.6 and above till capped at 1 for each additional match

calculatedWeatherPoint() function:

```
async function calculatedWeatherPoint(user_weather_preference, travel_month):
   // 1. Handle missing input
   if user_weather_preference is missing OR travel_month is missing:
      return score = 0.5,
            weather_data = null

   // 2. Fetch weather record for the place & month
   weather = query Weather_Conditions table where:
              place_id = this.place_id
              month = travel_month
   if weather does not exist:
      return score = 0,
            weather_data = null

   // 3. Extract user-preferred weather attributes
   userTemp = user_weather_preference.avg_temp
   userHumidity = user_weather_preference.humidity
   userType = user_weather_preference.weather_type

   // 4. Temperature scoring
   tempScore = 0.5
   if userTemp is not null:
      tempDiff = absolute(weather.avg_temp - userTemp)
      tempScore = 1 - (tempDiff / 20)       // 20°C tolerance
      tempScore = max(tempScore, 0)

   // 5. Humidity scoring
   humidityScore = 0.5
   if userHumidity is not null:
      humDiff = absolute(weather.humidity - userHumidity)
      humidityScore = 1 - (humDiff / 50)    // 50% tolerance
      humidityScore = max(humidityScore, 0)

   // 6. Weather type scoring
   typeScore = 0.5
   if userType exists:
      if lowercase(weather.weather_type) == lowercase(userType):
         typeScore = 1
      else:
         typeScore = 0

   // 7. Average scores into final score
   totalScore = (tempScore + humidityScore + typeScore) / 3
```

```
// 8. Return final score and weather info
return score = round(totalScore to 3 decimals),
      weather_data = {
          month: weather.month,
          avg_temp: weather.avg_temp,
          humidity: weather.humidity,
          weather_type: weather.weather_type
      }
```

Table 4.5. calculatedActivityPoint() Pseudo Code

calculatedActivityPoint() is used to calculate weather compatibility score between what weather the user wants, and what weather the place typically has in the selected travel month. The function starts by using place_id and travel_month parameters to fetch weather data stored inside the database that is related to the location at that month. Next, the function will try to calculate three scores related to the weather data: temperature, humidity, and weather type scoring. For temperature scoring, the difference between user's preferences and the actual temperature data of the location in user's travelling month. A 20% tolerance is applied and if the range exceeds that value, the score will be 0, and if the difference is in the accepted range, the score will become higher the closer the difference is, till the score is capped at 1. Humidity scoring also apply the same logic of temperature scoring, and will also calculate the difference between user's preferred humidity to return the scoring accordingly. As for weather type scoring, if the weather data show that it is the exact match to user preference, the score will be 1, otherwisem it will be 0. Finally, the function will use all 3 scoring to calculate and return the average which is rounded to 3 decimal places.

4.2. Results

4.2.1. HomePage / Landing Page

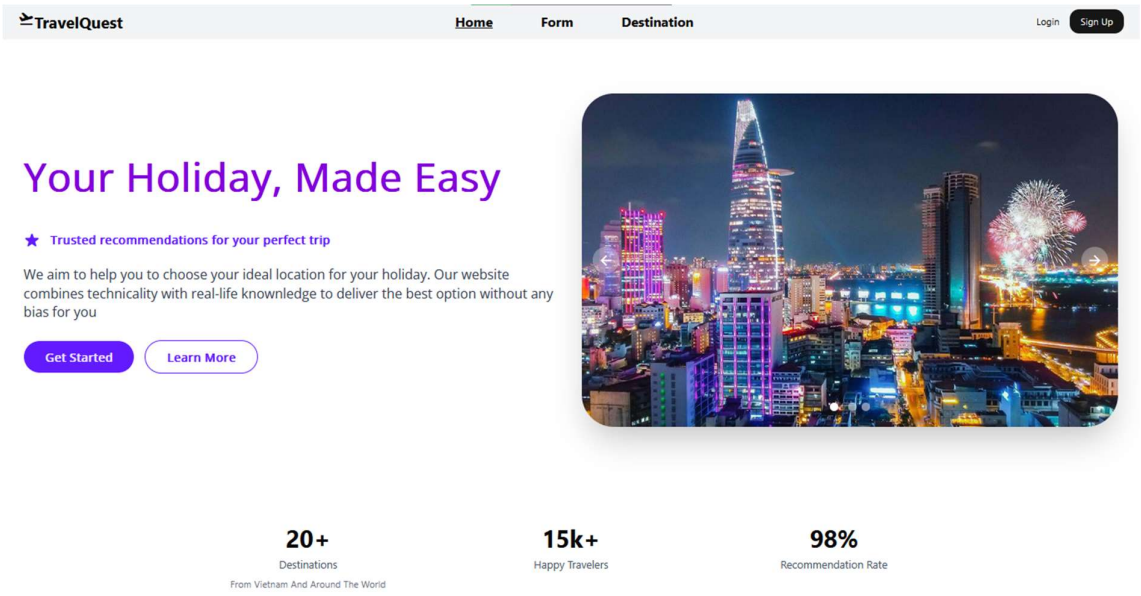

Figure 4.3 Landing Page

The Homepage is the start of the website as well as a default page to show up when user first connected to the website. The Homepage of the project focuses on simplicity, and efficiency. User should have no problem understanding where to start when first enter the website. There are three main components that make up the front page. Text Component, which include the slogan, and when as a short description of what does the page do. Two button components “Get Started”, and “Learn More” locates under the Text Component provides link when clicked on to redirect user to the preference form or QnA Page respectively. Finally, on the right side is a slideshow which showcases pictures of many locations inside the database. Each picture is shown for a few seconds before another one replaces it. The Navigation bar located on top of the website acts as easy way for user to move around the website. The navigation bar contains the name of my website, links to move to either homepage, form, or destination search page, and a sign-up button for user to register to the website.

4.2.2. User Form

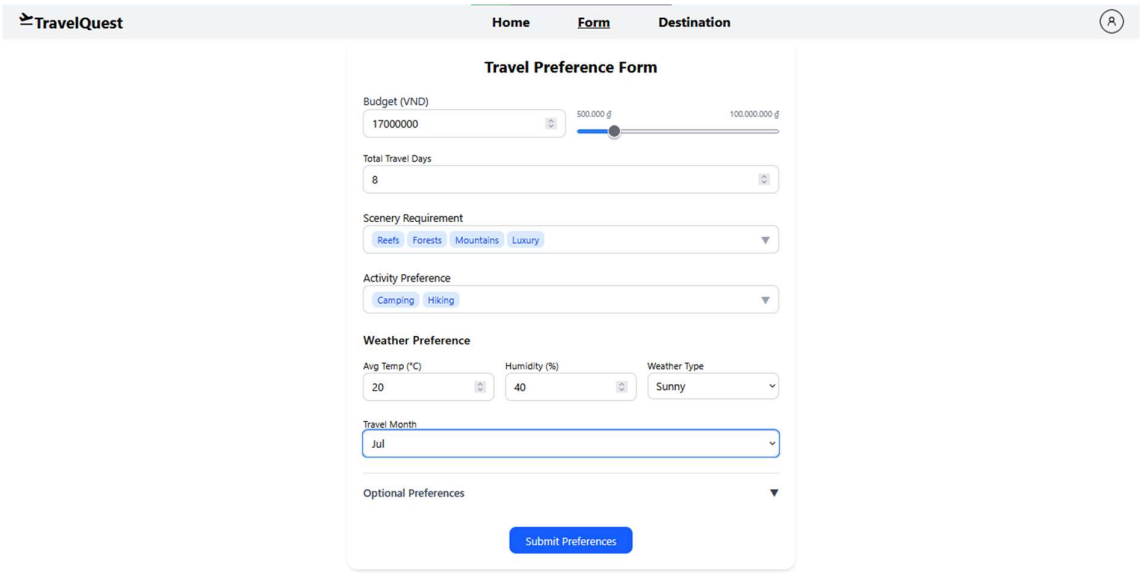

Figure 4.4.1 User Form Page

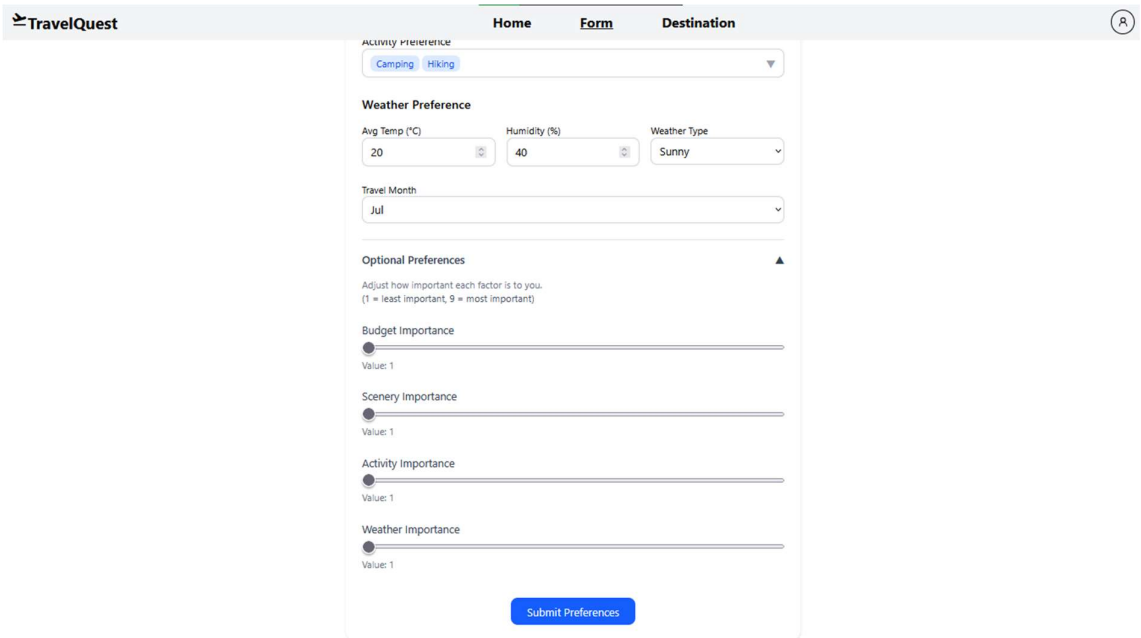

Figure 4.4.2 User Form Expanded

The User's Preference Form is where the users are able to input their needs using the provided criteria. For easier time using, each criteria is set to used a default value

so that user does not need to fill in all criteria before they are able to get their recommendation.

Initialy, Optional Preferences Tab is closed when user first accesses the page. The content of Weigth Value, which is set by default to all 3, can be access after the arrow button on the right of Optional Preferences is clicked. The Form is made from a combination of selection boxes, sliders, and text boxes. For the budget section, the text box is made to only accept numberical value, and an error will occur when a wrong value is input. Selection boxes for Tourist density, Activities preferences, and Weather preference, which can have multiple selection.

4.2.3. Result Page

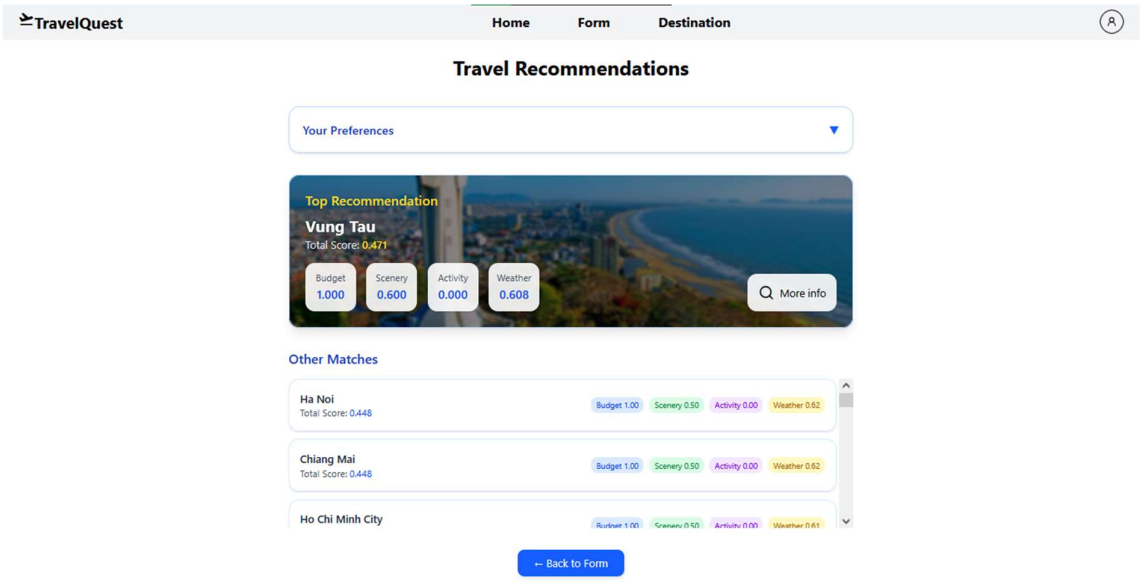

Figure 4.5.1. Result Page

Result page displays personalized travel results returned from Backend based on the user's submitted preferences. At the top of the page, a summary of user's preferences can be opened and collapsed, and is used to remind user of their input, and help user make a decision on the recommended place.

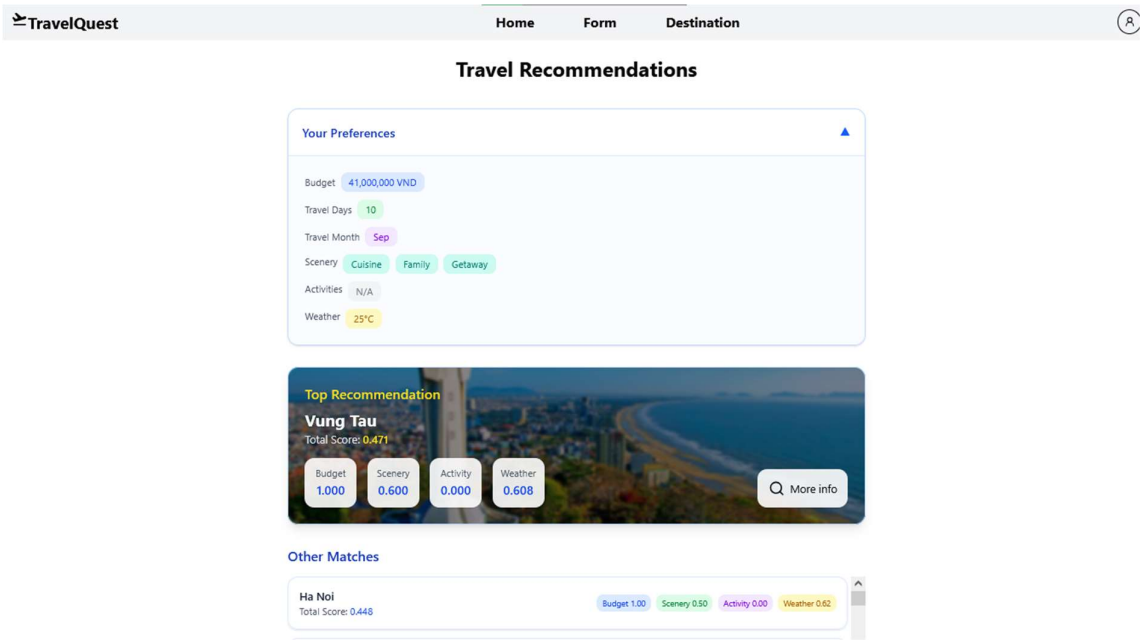

Figure 4.5.2. User Preferences Tab, now open

The destination with the highest score will have it own section. In this section, the total score represents how well the destination matches the user's preferences. Individual category is shown in small, rounded badges to explain how total score is calculated. And a "More info" button when clicked navigate user to the location information pages to show in depth information about the location if user is interested.

"Other Matches" section is used to list other top ranking destinations. Each place is also display their score, which can help user to compare the highest ranked location with the other, and to see which aspects the most suitable recommendation do well in.

4.2.4. Destination Search Page

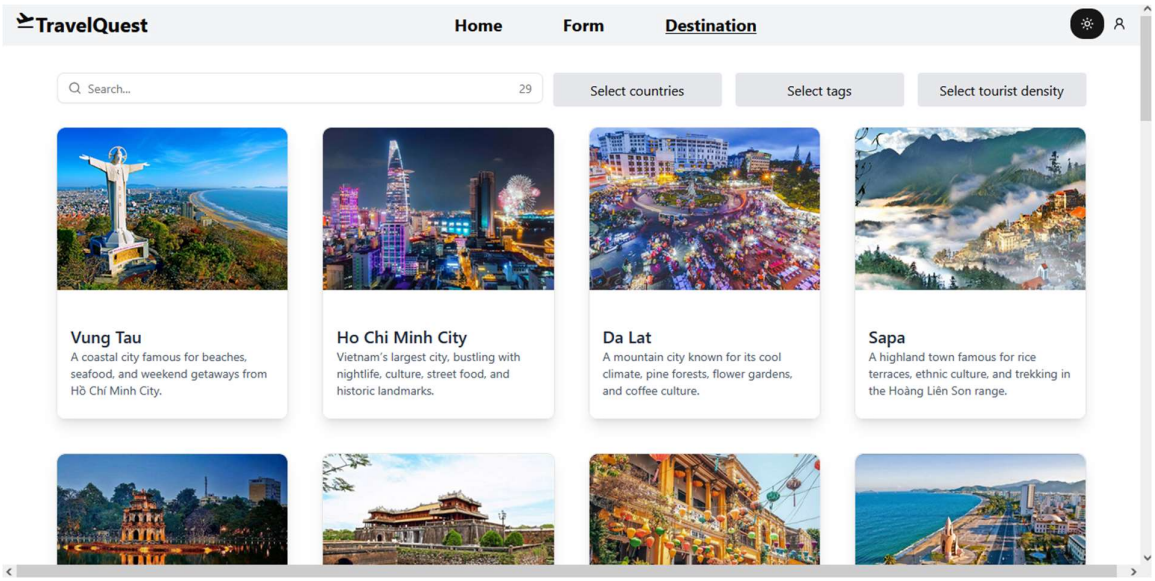

Figure 4.6.1. Destination Finder

Search Page allow users to search for a location using its name or a string of alphabet. Each location inside All Locations section is a Card Component. Each card component is created with React as a reusable component which include the location name, its picture, as well as a tag to see whether the location is popular, just added to the website, or is featured. When clicked on will show user to the location's page inside the website.

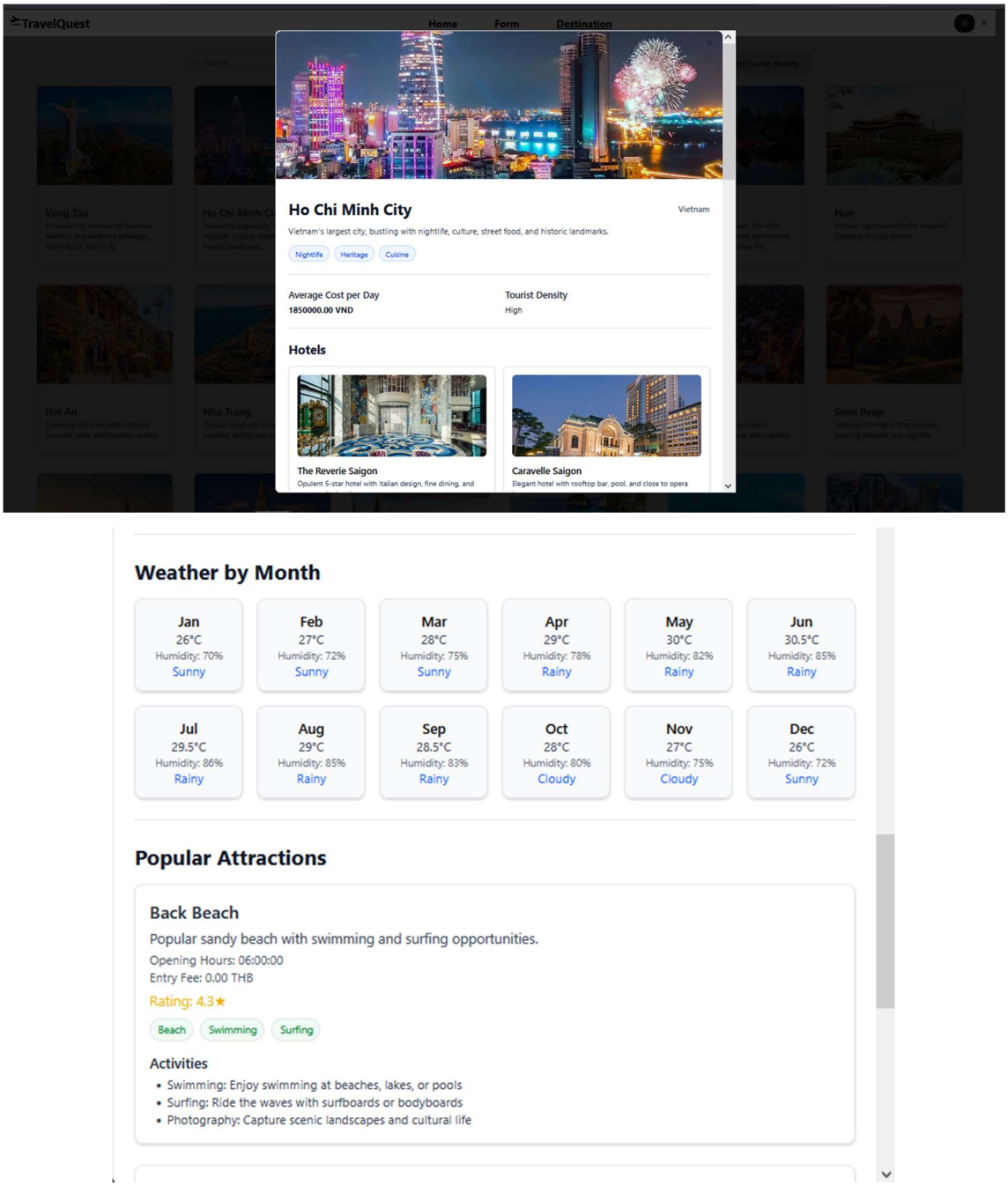

Figure 4.6.2.a & 4.6.2.b Detailed Location Popup

In this page, users are able to view some general information about the location such as its tourist density and average cost per day. From here, user will also be able to view information about any points of interrest or hotel related to that location via a link inside the page.

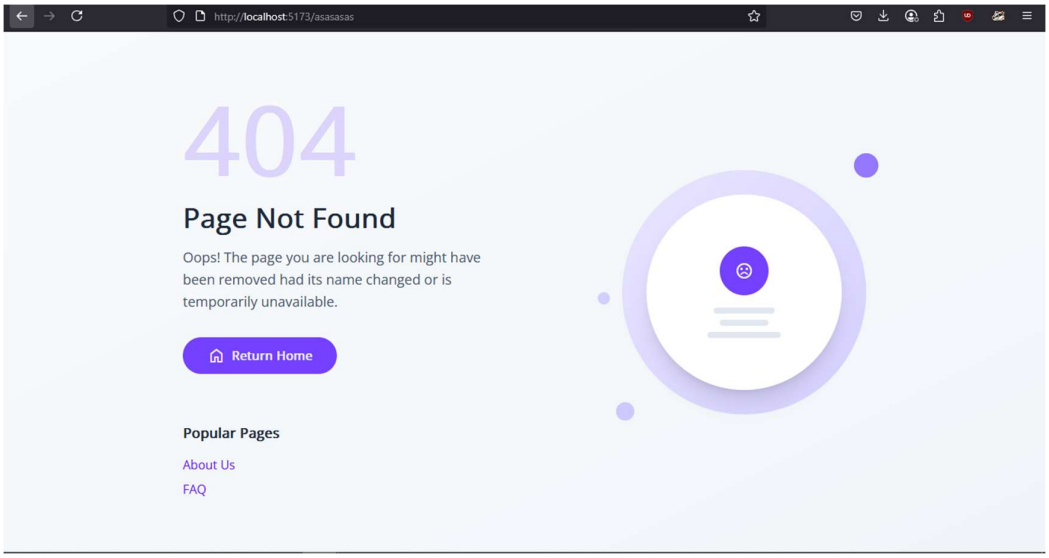

Figure 4.7. 404 Not Found Page

Not Found Page only be accessible if user has entered the wrong link, or have spelling mistakes inside their url. The design of this page centers around notifying the user of their error, therefore, the error code 404 is enlarged to take up a noticeable portion of the page, followed by an error description of "Page Not Found" and a quick description of the situation that the user encountered. The page also provides the user with a quick way to get back to Homepage through "Return Home" button. The page also suggests popular pages for user to visit such as About Us and FAQ or Frequently Asked Questions

CHAPTER 5

DISCUSSION AND EVALUATION

5.1. Discussion

The objective of this section is to go over all findings found during this project, as well as addression the research question explored in previous chapters, and the theoretical application this project may provides.

Decision Support System (DSS) is the project aims at assisting a group or an individual when choosing where to go next on their vacation, using their personalized preferences as a base for recommendation. The system also integrates a close-knit relationship between tables used in side the relational database with various attribute to keep up with users' complex requirement

The result of each usage from the user demonstrate the effectiveness of the point-based logic used by the system. Point-based algorithms offer a straightforward calculation (e.g., summing points for each place), which makes the process transparent and easy for user to understand. And, with the UI also provides a ranked list of suggestions, users can also able to see the alternatives to the highest ranked place, as well as see what category the top result is better in compared to the other.

During testing, it can be seen that accuracy in recommendations improved the more the user provided, which indicates that the logic used for recommendation is works as expected. Futhermore, over fifty percentage of users are reported to have feel satisfied with the system's output, proving the potential of such project when it is further improved on

The results from this study provide support for theories of decision support systems in complex, multi-criteria decision-making environments. Specifically, the study confirms that a point-based decision logic can offer a transparent and systematic method for guiding people in making decision in a complex topic, such as tourism. By quantifying preferences and systematically ranking options, DSS models can helps enhance decision-making processes by reducing the cognitive load on users and providing objective, data-driven and no bias suggestions.

5.2. Comparision

5.2.1. Compare with TravelMyne

This application will be compared with TravelMyne on these criterias: decision-making logic, UI/UX, and recommendation quality.

Both this project and TravelMyne calculate recommendations using a similar weighted scoring system. However, unlike this project, TravelMyne provides a longer list of categories to better fine tunes user preferences, as well as provide an option to do either quick search or detailed search

As for the UI/UX, both this project and TravelMyne choose to use a variety of fields and buttons to get the scope of user preferences. However, TravelMyne opts for the multi-page select option, which can sometime be more overwhelming than the single page designs this project chooses to use. The over-usage of words also makes it so that TravelMyne page feels crowded.

Finally, on the topic of recommendation quality: TravelMyne provides personalized recommendations based on the user's preferences and budget but lack of deep personalization because of the limited choice in each category, but make up by having a vastly superior dataset compared to the database of this project.

In conclusion, both systems have their strengths, and depending on the target audience, either platform could be preferable: TravelMyne for casual users and those looking for quick suggestions, while this DSS is for users who are seeking a more customized and detailed travel planning experience.

Feature	Thesis Project	TravelMyne
UI design	Web-based, single page form	Web-based, each parameter separated by page,
System Type	Logic-Based Decision Support System	Collaborative Filtering or Content-Based Recommendation System
Scoring Methodology	Point-based scoring system with weighted category	Point-based scoring system
Customizable Category	Budget. Scenery, Activies, Weather	Travel Time, Climate, Location, Region, Top Criteria, Nature, Culture, Activities
Data Source	Manual admin input, auto input with web crawling (to be implemented)	External sources, user input
Performance	Quick	Quick
Explain reasoning	Yes	No

Table 5.1 Comparision Table with TravelMyne

5.2.2. Compare with Wonderplan

Compared with Wonderplan, there are a few criteria to be covered – Personalization and Logic, User experience, and Recommendation quantity and quality.

For the first criteria, this Decision Support System project operates on a logic-based system, thus, it is easier for the user to not only change the importance of certain criteria, but also to see why one place is ranked higher than the other. Wonderplan, with the use of AI models, has proven to be difficult to fine-tune the logic of recommendation to each of the user's desires.

Next, anout the second criteria, this DSS used a variety of fields to collect the user preferences before applying the logic to get the ranked location list. This approach, while it is simple and straightforward to use, is still prone to user-side errors, which may cause unexpected behavior. Wonderplan, also provides the user with a clean and easy to navigate

UI with limited options but has better error checking in check filed, which helps mitigate errors from users.

Finally, with a limited set of options, Wonderwall recommendation quality can be lacking in this project at times. However, with a more complex dataset, Wonderwall has a wider range of recommendations in comparison with this thesis project.

To summarized, while this project offers a more personalized and dynamic decision-making process through its point-based multiple-criteria scoring system, Wonderwall excels in its simplicity and ease of use, making it more appealing to users looking for quick suggestions.

	Thesis Project	Wonderplan
Usage	Find a destination through ranking location	Create plan with predetermined destination with AI
Technology usage	Point-based scoring system with weighted category	LLM to help create plan
Customizable Category	Budget. Scenery, Activies, Weather	Budget, Activies, Travel Style (Alone, With Group)
Target Audience	People who want to try new location or those who don't know where to go	People already with location in mind
Explain reasoning	Yes	No
Subscription	No	Yes, must make an account to use

Table 5.2 Comparision Table with Wonderplan

5.2.3. Conclusion

In comparison with initial expectations, the Decision Support System (DSS) developed in this study met most of the goals, particularly in terms of personalized recommendations and performance. However, challenges with niche preferences and using mainly a simple algorithm suggest areas for future improvement. When compared with existing systems in tourism, this DSS demonstrates advantages in customization of user preferences and logic

handling, although it could benefit from having a better dataset as well as more category to fine tune user preferences even more. The system's use of a point-based decision logic in combination with weighted category sets it apart from review-based or user-data-driven approaches, making bias having a lesser effect on the final result of the system

If this system is to be further developed, future improvement should focus on refining the user interface, expanding the dataset, as well as incorporating method to update data to improve both the accuracy and relevance of the recommendations.

5.3. Evaluation

5.3.1. Effectiveness of the System

In this section, the project is asseted on how well it achieves the project goal, which is to recommend a suitable place as an option for user to choose on holiday. These are the category to be used to calculate the system effectiveness: Accuracy and Relevancy, and Performance.

About accuracy, the project had accurately provided result match with user's need. For all runs, all the parameters are correctly aligned with provided input thanks to the logic of point-based algorithm. The project will push location that is the most identical to user inpur, as in have the most tags from the list provided, having the budget not over the provided limit, as well as have all the wanted activities to the first of the ranked list and the least similar places will be placed last. The project also be able to adapts to the more complex user preferences through the use of multi-criteria decision support system as well as weight category scoring system.

For the performance, the project uses various techniques, including placeholder component as well as cached data to help improve the loading time of each website. The recommendation result also gets returned to user within seconds, which is acceptable.

5.3.2. Strength and Limitations

The system's use of a point-based algorithm combined with weighted categories allows it to generate highly personalized tourism recommendations tailored to individual preferences. This means users can receive suggestions that align closely with their interests, budget, and other criteria such as preferred activities or weather preferences. If a user inputs that they prefer budget-friendly, nature-based experiences, the system would prioritize destinations like national parks or beach towns with a range of affordable accommodation options. This personalization also helps to include new places for user to consider.

Moreover, the weighted category system gives users control over the degree to which each factor influences the recommendations, which allow for better fine tuning of result that better reflect user's priority. For instance, if tags category has higher weight value, the system will prioritize places includes the most tags from user lists if it found multiple places with the same score.

As for the system's weakness, the most important one is the over-simplication of user preferences when using current algorithm. While weighted category is serviceable, Travel choices are often influenced by factors that go beyond the categories included in the system, thus, a more complex algorithm that taken into account those variable is needed and should be look more into.

Another limit of the system is the Scope of Data. As there is only limited time to develop the system, the quality and quantity of the data is not the main factor of concern. Therefore, the dataset used inside the project may be incomplete, outdated, or biased. This limitation highlights the importance of having a way to regularly updating the system's data.

Finally, through generalization of users, the system may not always be able to fully account for the diverse needs of different types of travelers, such as those with disabilities, solo travelers, or those seeking very niche experience

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1. Conclusion

In conclusion, over the course of a few months, this Decision Support System (DSS) project has since been developed into a useful tool to assist users on choosing their holiday stay through the use of the point-based logic recommendation system with personalized criteria. While the algorithm can be simple at time, this DSS provides adequate results that are relevant to the user's preferences. By using the system, users can explore unfamiliar places that they might have not consider

On the UI/UX side, this DSS uses an intuitive design with simple buttons and input fields with minimal special effects which is helpful in guiding users when they first use the website.

This Decision Support System aims to further explore the topic of choosing a destination on holiday, a topic that has not been extensively researched, especially with the recent advancement in technology. Not only does this system can be used as an alternative way to plan one's holiday, but this system can also be used as a base to further implement advanced technology such as the AI system to improve both the system's performance as well as user's experience.

With such a short development time, there are still some limitations that must be considered moving forward. Firstly, the quality of the database used inside the project. Without proper automation for hourly update of the system, some factors, such as real-time availability as well as ongoing events might be inaccurate and not properly incorporated into the website. Secondly, because the system is built with users to personalize input, the system failed to properly handle cases where a detailed user preference is not provided. And finally, the usage of point-based system inside project might be too simple for certain cases and should be replaced with more advanced algorithm if the system is to be further developed

While the system successfully meets its objectives, further improvements can be made. If further improvement can be made to the project, this Decision Support System will be able to provide beyond its original objective and becomes a better tool that can be used by many people.

6.2. Future work

While the Decision Support System developed in this thesis has addressed the problem of recommending vacation destinations, there are still several areas left where further development could be made.

If the project is to be further improved, data accuracy should be the first thing to be considered. Real-time data should be gathered and processed automatically, with limited human intervention, and collected from multiple sources to improve the database's accuracy and the data's quality and quantity.

Furthermore, the system could try to explore an alternative way for users to interact with the application, such as a voice recognition system, or an image-based recommendation system. The algorithm used inside the system should and could also be further improved with more complex and advanced algorithms that aim to improve both the performance and accuracy of the system.

By addressing these areas for future development, the system has the potential to significantly affect how users choose their destination moving forward, making the task of planning for a vacation no longer seem so tedious.

Decision Support System is great tool that should be further ultilized. By using it, the quality of tourism decision-making is able to be greatly improved by systematically evaluating alternatives based on predefined criteria, eliminating the hindering bias causes. With DSS, no longer do users need to spend long hours deciding where to go from the ever-growing list of tourist destinations. Furthermore, through fine tuning preferences with priority, the system is able to perfectly adapt to the need of users, and able to provide the satisfied results to everyone. From a broader perspective, such DSS applications contribute to improving tourist satisfaction and enhancing decision support practices within the tourism industry.

And finally, from the thesis, it can be said that for dealing with multi-criteria, preference-based, and semi-structured destination for tourist, a Model-Driven Decision Support System is the contender for the most effective one with it mathematical and logic-based approach, suitable for such semi-structured decision problems.

REFERENCES

[1] World Tourism Barometer, UN Tourism, https://www.unwto.org/un-tourism-world-tourism-barometer-data

[2] Economic impact of Travel & Tourism, World Travel & Tourism Council, https://wttc.org/research/economic-impact

[3] Ferrari, J.R. (2010). Still procrastinating? The no regrets for getting it done. John Wiley & Sons.

[4] TravelMyne, http://www.travelmyne.com/

[5] Wonderplan, 2024 Wonderplan. A product of Vecro Tech LTD., https://wonderplan.ai/

[6] A Brief History of Decision Support Systems, D. J. Power, editor at DSSResources.COM, https://dssresources.com/history/dsshistory.html

[7] Decision Support Systems: A Summary, Problems, and Future Trends, M.C. ER, Department of Computer Science, St. Patrick's College, National University of Ireland, Maynooth, Co. Kildare, Ireland

[8] Shaofeng Liu*, Alex H.B. Duffy, Robert Ian Whitfield and Iain M. Boyle, Integration of decision support systems to improve decision support performance, 2008

[9] Multi-Criteria Decision Analysis for Strategic Decision Making, Gilberto Montibeller and Alberto Franco

[10] Sang M. Lee and Hyun B. Eom, Multiple-Criteria Decision Support Systems: The Powerful Tool for Attacking Complex, Unstructured Decisions, 1990

[11] ODU, G.O, Weighting Methods for Multi-Criteria Decision Making Technique, SSN 1119-8362

[12] Monolithic Architecture - System Design, GeeksforGeeks, https://www.geeksforgeeks.org/system-design/monolithic-architecture-system-design/

[13] MVC Framework Introduction, GeeksforGeeks, https://www.geeksforgeeks.org/software-engineering/mvc-framework-introduction/

[14] Array.prototype.sort(), Mozilla Developer, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

[15] Array.prototype.slice(), Mozilla Developer, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice

APPENDIX