KARNATAK LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

Department of Computer Science and Engineering



*Project Report on*

**Personalized Travel Recommendation System**

*Submitted in partial fulfillment of the requirement for the award of the degree of*

**Bachelor of Engineering**

**in**

**Computer Science and Engineering**

*Submitted by*

|  |  |
| --- | --- |
| Sandesh Hiremath | 2GI19CS132 |
| Sanket Katti | 2GI19CS134 |
| Umesh Bhati | 2GI19CS168 |
| Vivek Valagadde | 2GI19CS186 |

**Guide**

Prof. Sagar Pujar

**2022 – 2023**

KARNATAK LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

Department of Computer Science & Engineering

 A picture containing emblem, symbol, logo, circle

Description automatically generated

**CERTIFICATE**

Certified that the project entitled **PERSONALIZED TRAVEL RECOMMENDATION SYSTEM** carried out by **Mr. Sandesh Hiremath** (2GI19CS132), **Mr. Sanket Katti** (2GI19CS134), **Mr. Umesh Bhati** (2GI19CS168), **Mr. Vivek Valagadde** (2GI19CS186) students of KLS Gogte Institute of Technology, Belagavi, can be considered as a bonafide work for partial fulfillment for the award of **Bachelor of Engineering** in Computer Science & Engineering of the Visvesvaraya Technological University, Belagavi during the year 2022- 2023. It is certified that all corrections/suggestions indicated have been incorporated in the report. The project report has been approved as it satisfies the academic requirements prescribed for the said Degree.

**Guide     HOD   Principal**

**Prof. Sagar Pujar Dr. V. S. Rajpurohit Dr. J. K. Kittur**

**Date:**

**Final Viva-Voce**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name of the examiners** | **Date of Viva -voce** | **Signature** |
| **1.** |  |  |  |
| **2.** |  |  |  |

**DECLARATION BY THE STUDENTS**

We, **Mr. Sandesh Hiremath**, **Mr. Sanket Katti**, **Mr. Umesh Bhati**, **Mr. Vivek Valagadde**, hereby declare that the project report entitled **Personalized Travel Recommendation System**

submitted by us to KLS Gogte Institute of Technology, Belagavi, in partial fulfillment of the Degree of **Bachelor of Engineering** in Computer Science and Engineering is a record of the project carried out at us. This report is for the academic purpose.

We further declare that the report has not been submitted and will not be submitted, either in part or full, to any other institution and University for the award of any diploma or degree.

|  |  |  |
| --- | --- | --- |
| Name of the student | USN | Signature |
| Sandesh Hiremath | 2GI19CS132 |  |
| Sanket Katti | 2GI19CS134 |  |
| Umesh Bhati | 2GI19CS168 |  |
| Vivek Valagadde | 2GI19CS186 |  |

Place: Belagavi

Date:

**Acknowledgement**

We would like to express our deepest gratitude to our esteemed project guide, Dr. Manjula Ramannavar for their invaluable guidance and exceptional mentorship throughout the duration of our final year project.

We would like to thank all the people that worked along with us for the project for their patience and openness in creating an enjoyable working environment.

It is indeed with a great sense of pleasure and immense sense of gratitude that We acknowledge the help of these individuals.

We would like to thank our mentor Dr. Manjula Ramannavar Ma’am for her constant support and guidance throughout our project tenure.

We are extremely grateful to my CSE department staff members and friends who helped us in successful completion of this project.

**Abstract**

This project investigates the implementation of a local Kubernetes area with K3s on Raspberry Pi metal hardware with the aim of creating a scalable and cost-effective computing solution. Detailed hardware specifications and software configurations are described, including lightweight Kubernetes distribution K3s, along with network configuration considerations to ensure smooth communication between cluster nodes. In addition, the integration of Grafana and Prometheus is explored to monitor and visualize cluster metrics, providing real-time insights into performance and resource usage. Through rigorous testing and analysis, this project reveals the challenges and opportunities of deploying Kubernetes on resource-constrained IoT devices like the Raspberry Pi. The findings demonstrate the feasibility of such implementations and highlight their potential to revolutionize edge computing paradigms by providing decentralized and decentralized infrastructures capable of supporting various applications at the edge of the network. Finally, opportunities for future research and development are identified, including resource utilization optimization and scalability improvements that will further advance the edge computing capabilities of Raspberry Pi clusters.

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Content** | **Page No.** |
|  | i. | Declaration | iii |
|  | ii. | Acknowledgement | iv |
|  | iii. | Abstract | v |
|  | iv. | Table of contents | vi |
|  | v. | List of Tables | vii |
|  | vi. | List of Figures | vii |
|  | vii. | List of Abbreviations | ix |
| **1.** |  | **Chapter 1:** **Introduction, Objectives, Purpose, Scope, and Applicability.** | **1** |
|  | 1.1 | Introduction | 1 |
|  | 1.2 | Objectives | 3 |
|  | 1.3 | Purpose, Scope, and Applicability | 3 |
|  |  | 1.3.1: Purpose |  |
|  |  | 1.3.2: Scope |  |
|  |  | 1.3.3: Applicability |  |
| **2.** |  | **Chapter 2: Literature Survey** | **5** |
|  | 2.1 | Introduction | 5 |
|  | 2.2 | Overview of Edge Computing | 5 |
|  | 2.3 | Kubernetes and K3s in Edge Computing | 16 |
|  | 2.4 | Monitoring Solutions for Edge Environments | 16 |
| **3.** |  | **Chapter 3: Hardware and Software Setup** | **18** |
|  | 3.1 | Hardware Specifications of Raspberry Pi Cluster | 19 |
|  | 3.2 | Installation and Configuration of K3s | 19 |
|  | 3.3 | Networking Setup for Cluster Communication | 27 |
| **4.** |  | **Chapter 4: Deployment of Application** | **29** |
|  | 4.1 | Containerization of Node.js Application | 29 |
|  | 4.2 | Deployment of Application Pods on K3s Cluster |  |
| **5** |  | **Chapter 5: Monitoring Infrastructure Setup** | **30** |
|  | 5.1 | Introduction to Grafana and Prometheus | 30 |
|  | 5.2 | Installation and Configuration of Prometheus | 32 |
|  | 5.3 | Integration of Grafana with Prometheus for Visualization | 33 |
| **6** |  | **Chapter 6: Monitoring Application with Prometheus** | **36** |
|  | 6.1 | Instrumentation of Node.js Application for Prometheus | 36 |
|  | 6.2 | Configuration of Prometheus to Scrape Application Metrics | 37 |
| **7** |  | **Chapter 7: Experimentation and Results** | **43** |
|  | 7.1 | Performance Evaluation Metrics | 43 |
|  | 7.2 | Analysis of Cluster and Application Performance under Various Workloads |  |
| **8** |  | **Chapter 8: Discussion** | **46** |
|  | 8.1 | Challenges Encountered during Setup, Deployment, and Monitoring | 46 |
|  | 8.2 | Insights into Monitoring Edge Environments with Grafana, Prometheus, and Node.js | 47 |
| **9** |  | **Chapter 9: Application and Conclusion** | **53** |
|  | 9.1 | Application | 53 |
|  | 9.2 | Conclusion | 53 |
|  | 9.3 | Future Scope off the Work | 53 |
| **10** |  | **Reference** | **55** |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| 3.1 | Software requirements | 18 |
| 7.1 | Unit Test Result | 44 |
| 7.2 | Integration Test Result | 45 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| 1.1 | Tourist Arrival and Revenue Earned in India | 2 |
| 2.1 | RS in Machine Learning | 6 |
| 2.2 | Hybrid Method for TRS | 7 |
| 2.3 | Concept of Recommender System | 8 |
| 2.4 | Framework of TRS | 10 |
| 2.5 | System Architecture for proposed Personalized TRS | 11 |
| 2.6 | Cosine Similarity for Vector Comparison | 12 |
| 2.7 | Flow Chart of SVD Approach | 14 |
| 2.8 | System Design for Cosine Similarity and SVD based Approach | 15 |
| 3.1 | Use Case Diagram for TRS | 19 |
| 3.2 | Sequence Diagram for TRS | 21 |
| 3.3 | Activity Diagram for Registered User | 23 |
| 3.4 | Activity Diagram for New User | 24 |
| 3.5 | Class Diagram | 25 |
| 3.6 | State Diagram | 26 |
| 4.1 | Pert Chart | 29 |
| 5.1 | System Architecture | 30 |
| 5.2 | Interface Design | 32 |
| 8.1 | Confusion Matrix | 46 |
| 8.2 | Classification Report | 47 |
| 8.3 | Home Page | 47 |
| 8.4 | Services Page | 48 |
| 8.5 | Signup Page | 48 |
| 8.6 | Login Page | 49 |
| 8.7 | Input Recommendation Form | 49 |
| 8.8 | Post-Submission of Form | 50 |
| 8.9 | Recommendation Page 1 | 50 |
| 8.10 | Recommendation Page 2 | 51 |
| 8.11 | Ratings Page-1 | 51 |
| 8.12 | Ratings Page-2 | 52 |

**List of Abbreviations**

|  |  |
| --- | --- |
| **Abbreviation** | **Title** |
| K3s | Lightweight Kubernetes distribution |
| IoT | Internet of Things |
| CPU | Central Processing Unit |
| RAM | Random Access Memory |
| API | Application Programming Interface |
| HTTP | Hypertext Transfer Protocol |
| DNS | Domain Name System |
| YAML | Yet Another Markup Language |
| SSH | Secure Shell |
| CLI | Command Line Interface |
| GUI | Graphical User Interface |
| LAN | Local Area Network |
| WAN | Wide Area Network |

**Chapter 1**

**1. Introduction, Objectives, Purpose, Scope, and Applicability.**

**1.1 Introduction**

In recent years, the rise of IoT devices and the emergence of data-intensive applications have increased the need for distributed computing architectures that can process data closer to the source. The move to edge computing is to ease the burden on the cloud's core infrastructure, reduce latency, and improve privacy and security by processing data locally. However, deploying and managing raw computing environments presents unique challenges, including resource limitations, network connectivity issues, and the need for efficient deployment methods.

Kubernetes, a powerful containerization platform, has emerged as a key technology for managing distributed workloads in distributed environments. K3s, a lightweight Kubernetes distribution designed for resource-constrained environments, provides an easy way to deploy Kubernetes clusters on edge devices. Using K3 on Raspberry Pi devices is metal-free and a powerful solution for building edge computing infrastructures, enabling organizations to leverage automation and orchestration in distributed environments.

**1.2 Objectives**

This project aims to investigate the feasibility and effectiveness of deploying a local Kubernetes cluster using K3 on a Raspberry Pi device for edge computing applications. The main objectives are:

* **Setup and configuration:** Setting up a Kubernetes cluster on a Raspberry Pi device, including hardware requirements, software installation, and network configuration.
* **Application Deployment:** Deploy and deploy Node.js applications in a Kubernetes environment to demonstrate hosting global workloads on the network side.
* **Monitoring and Management:** Implement monitoring and management solutions such as Prometheus and Grafana to track team performance and ensure efficient resource utilization.
* **Performance Evaluation:** Run dynamic tests to evaluate the performance and scalability of Kubernetes clusters and applications deployed across multiple workloads.

**1.3 Purpose, scope, and applicability**

**1.3.1 Purpose**

The purpose of this project is to showcase the practical implementation of Kubernetes on edge devices that are limited in resources and offer a glimpse into the workings of edge computing. Addressing the complexities of edge infrastructure deployment, application deployment, and monitoring, this project provides stakeholders with the tools and knowledge to leverage edge computing as a transformative technology across a variety of use cases and industries.

**1.3.2 Scope**

The scope of the project includes setting up a Kubernetes cluster using K3s on Raspberry Pi hardware, deploying a Node.js application to the cluster and configuring the monitoring infrastructure with Prometheus and Grafana. In addition, the project includes performance evaluations to evaluate the scalability and the utilization of the resources of the cluster and the application used. Extensive documentation is maintained throughout the project, including detailed installation procedures, kits and test results. In addition, the project explores possible future extensions, such as the integration of advanced features of Kubernetes and optimization techniques to improve the capabilities of the deployed infrastructure

**1.3.3 Applicability**

The proposed system offers the following practical applications:

* **IoT Implementation:** Edge computing allows data generated by the Internet of Things to be efficiently processed at the edge of the network, reducing latency and bandwidth requirements when transferring data to centralized servers. Deploying Kubernetes on Raspberry Pi clusters provides a cost-effective solution for managing IoT workloads in distributed environments.
* **Edge Infrastructure Development:** The project is a hands-on exploration of edge infrastructure development and provides insight into the challenges and opportunities associated with deploying containerized applications on resource-constrained devices. Through a practical experiment, stakeholders will gain valuable information and knowledge about the principles and methods of edge calculation.
* **Research and Innovation:** As advanced computing advances, the need for research and innovation in the design, management and optimization of edge infrastructure increases. This project advances knowledge of edge computing by presenting a comprehensive study of the implementation of Kubernetes on Raspberry Pi clusters and its impact on the development and management of edge applications.

**Chapter 2**

**2. Literature Survey**

**2.1 Introduction**

A literature survey, also known as a literature review, is a survey of previously published scholarly resources such as books, journals, and articles on specific themes or concerns. It entails searching for and evaluating available material in your selected subject or issue area. It documents the state of the art in the subject or issue about which you are writing. Concerning the project, a literature survey was undertaken to gain a better understanding of the idea, for idea growth, and to learn about the limits inherent in the correct approaches.

**2.2 Summary of papers**

**Kunal Shah** have presented an overview of the field of recommender systems and describes the present generation of recommendation methods. Recommender systems or recommendation systems (RSs) are a subset of information filtering system and are software tools and techniques providing suggestions to the user according to their need. Many popular Ecommerce sites widely use RSs to recommend news, music, research articles, books, and product items. Recommendation systems use personal, implicit, and local information from the Internet. The study attempts to describe various limitations of recommendation methods and their advantages. An overview of different approaches to recommendations provides an overview of the different approaches that are commonly used in building recommender systems. Recommender systems are a type of information filtering system that predicts the preferences or interests of users and recommends items that are most likely to be of interest to them. The study begins with a brief introduction to recommender systems, followed by an explanation of the various approaches used in developing these systems.

The study first discusses the content-based approach, which recommends items based on the similarity of their attributes with those of items that the user has liked in the past. The collaborative filtering approach is then discussed, which recommends items based on the user's past behaviour and similarity with other users' behaviour. The paper also discusses hybrid systems, which combine the content and collaborative filtering approaches to overcome their individual limitations. The authors then introduce other approaches, such as knowledge based, demographic-based, and community-based filtering. Knowledge-based systems use expert knowledge to provide recommendations, while demographic-based systems recommend items based on the user's demographic information such as age, gender, or location. Community-based systems rely on the user's social network to provide recommendations. Finally, the study concludes with a discussion on the evaluation of recommender systems. The authors explain the various evaluation metrics used to measure the effectiveness of the recommendation system, such as precision, recall, and F1 score. The study also discusses the challenges and limitations of developing recommender systems, such as the cold-start problem and the sparsity of data.

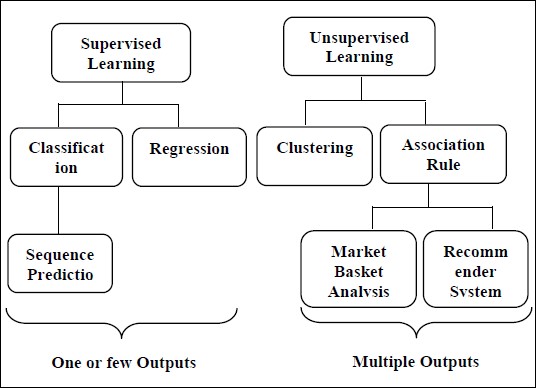


Figure 2.1: RS in Machine Learning

**Hela Masri** have proposed a hybrid tourism recommender system that combines collaborative filtering and content-based approaches to generate personalized recommendations for tourists. The study begins by discussing the importance of personalized recommendations in the tourism industry and the challenges of generating accurate and diverse recommendations. The proposed system consists of three main components: the user profile, the item profile, and the recommendation generation engine. The user profile includes information about the user's preferences and behaviour, while the item profile includes features, of the tourism destinations and attractions. The recommendation generation engine combines both collaborative filtering and content-based approaches to generate personalized recommendations for the user. The authors evaluate the performance of the proposed system using a dataset of tourism destinations and attractions in Tunisia. They compare the performance of the hybrid approach with pure content-based and collaborative filtering approaches and demonstrate that the hybrid approach outperforms the other two approaches in terms of accuracy and diversity of recommendations. Overall, the paper provides a novel approach to generating personalized recommendations in the tourism industry and highlights the importance of combining different techniques to achieve better accuracy and coverage. The proposed system has the potential to improve the tourism experience for travellers and help promote tourism destinations and attractions. It serves as a useful resource for researchers and practitioners in the field of tourism recommender systems and provides valuable insights into the challenges and opportunities in this area of research.

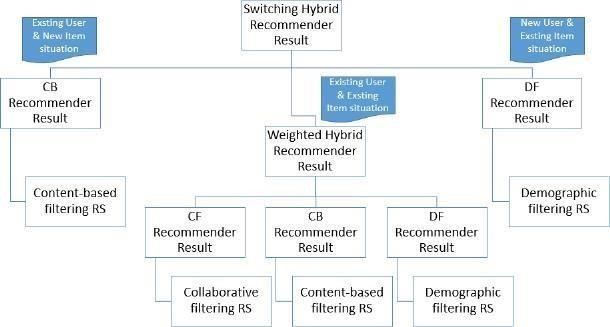


Figure 2.2: Hybrid Method for TRS

**Marwa Hussien Mohamed** have researched on to a survey of the challenges and solutions in the field of recommender systems. The study begins by introducing the concept of recommender systems and then increasing the only, importance in various applications, including e-commerce, social media, and entertainment. The study then discusses the challenges faced by recommender systems, including data sparsity, cold-start problems, scalability, and the challenge of incorporating contextual information. They also highlight the importance of accuracy, diversity, and novelty in recommendation generation. The study discusses various techniques and algorithms used in addressing these challenges, including matrix factorization, clustering, and deep learning. The study also highlights ethical considerations in recommender systems, including issue of bias, fairness and then study also highlights the ethical considerations in recommender systems, including issues of bias, fairness, and privacy. The study discusses various solutions to address these concerns, including the use of transparency and interpretability techniques and the incorporation of ethical considerations in the design and development of recommender systems. Overall, the study provides a comprehensive overview of the challenges and solutions in the field of recommender systems, highlighting the importance of addressing ethical concerns and the need for more accurate and diverse recommendation generation techniques. It serves as a valuable resource for researchers and practitioners in the field of recommender systems and provides useful insights into the current state of the field and future directions for research.

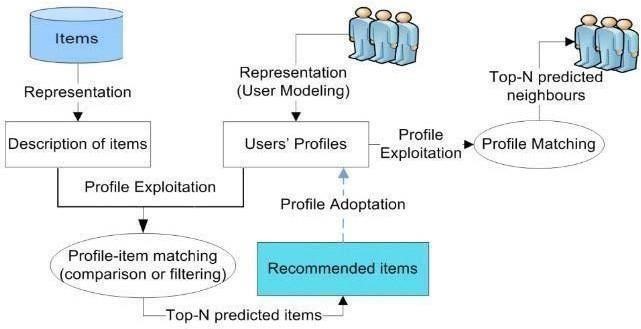


Figure 2.3: Concept of Recommender Systems

**Ivens Portugal** provided an overview of the use of machine learning algorithms in recommender systems. The study begins by introducing the concept of recommender systems and their increasing importance in various applications, including e-commerce, social media, and entertainment. The study discusses the challenges faced by traditional recommender systems and how machine learning techniques can be used to address these challenges. They highlight the importance of data pre-processing, feature selection, and model selection in achieving better accuracy and coverage in recommendations. The study discusses various machine learning algorithms used in recommender systems, including k-nearest neighbours, decision trees, support vector machines, and neural networks. The study discusses the evaluation of recommender networks. The study also discusses the evaluation of recommender systems and the various metrics used to measure their overall effectiveness, including precision, recall, and F1-score. The study highlights the importance of offline and online evaluations in ensuring the quality of recommendations and the need for continuous improvement and adaptation of recommender systems.

Overall, the study provides a comprehensive overview of the use of machine learning algorithms in recommender systems, highlighting the strengths and limitations of different techniques and algorithms. It serves as a valuable resource for researchers and practitioners in the field of recommender systems and provides useful insights into the current state of the field and future directions for research.

**Shuang Cangb** proposed a decision tree- based recommendation system for tourists. The authors begin by introducing the concept of recommender systems and their increasing importance in the tourism industry. The study highlights the challenges faced by tourists in planning their trips, including the abundance of information available and the difficulty in selecting the most suitable destinations and activities. The proposed system is a decision tree-based recommendation system that considers various factors, including the tourist's preferences, budget, and travel time, to generate personalized recommendations. The study discusses the methodology used in building the decision tree and the various metrics used to evaluate its effectiveness. And provide a case study of the proposed recommendation system in action, highlighting its usefulness in helping tourists plan their trips more efficiently. Overall, the study provides a practical and effective approach to addressing the challenges faced by tourists in planning their trips. The decision tree-based recommendation system proposed provides personalized recommendations that consider various factors, making it a valuable resource for tourists and tourism operators alike. The study serves as a valuable resource for researchers and practitioners in the field of recommender systems and provides useful insights into the application of decision tree-based techniques in tourism.

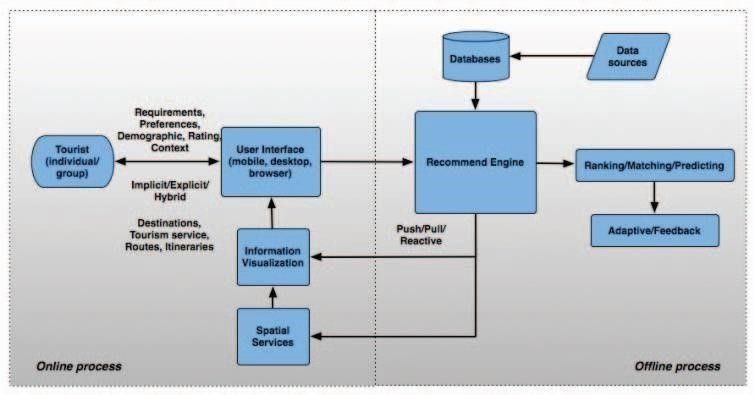


Figure 2.4: Framework of Travel Recommendation System (TRS)

**Prof. N.G Pardeshi** presented a personalized travel sequence recommendation system that uses travelogues data, community contributed data which includes images, heterogeneous metadata (e.g., Tags, geo-location, and date taken) associated with those images to recommend users the Point of Interest (POIs) to visit. They had constructed a Topical package space by collecting travelogues, community Contributed data. The system will analyse this synthetic data set to get the maximum Point of Interest between sources to destination of user travel route within specified distance. Unlike most existing travel recommendation approaches, our approach is not only personalized to users travel interest but also able to recommend a travel sequence rather individual Points of Interest (POIs). Travelogues data is the data collected from travel sites (e.g., www.hellotravel.com)that offer rich depictions about historic points and voyaging background composed by clients. Moreover, group contributed data with metadata (e.g., photos, POIs, date taken, tags and so on.) collected from social media sites record client’s day by day life and travel understanding. But can this information from various travelogue sites and group contributed sites used to recommend user a well-planned travel sequence such that maximum POIs other than individual POIs would be covered the current system would fail here. Hence the proposed system provides personalized travel sequence recommendation to users by analysing his or her Topical interests and recommends POIs the user should visit.

To obtain results such that maximum Point of Interest would be covered the system used Greedy algorithm so that it can get optimized results with maximum POIs.

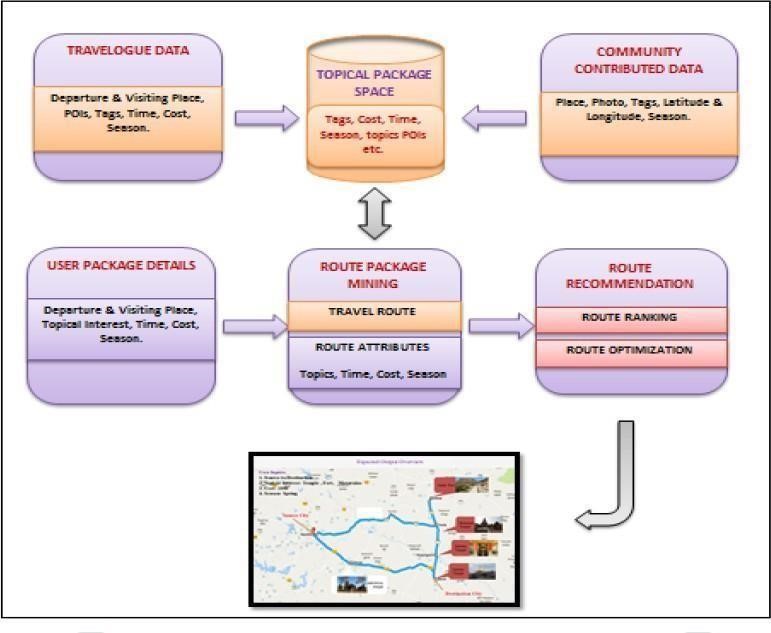


Figure 2.5: System Architecture for proposed Personalized TRS

**Shashank Jagtap** described an approach which offers generalized recommendations to every user using the Cosine Similarity algorithm. The dataset used exhibits a vast and distinct combination of tourist places. The Cosine Similarity algorithm predicts the most relevant tourist places using some important features of the dataset such as tourism category, minimum budget (per day) and the visa requirement. The amount of information is increasing day by day. To traverse this information in order to find relevant information is difficult. To automate the process of traversing this enormous information, we use machine learning algorithms. They are 3 main types: Supervised: learning a function that maps an input to an output based on example input-output pairs, Unsupervised: algorithms are left to their own devices to discover and present the interesting structure in the data without a supervisor and Reinforcement: algorithms are concerned with how software agents ought to take actions in an environment in order to maximize some notion of cumulative reward. We use these machine learning algorithms to find patterns and similarities between different items of a dataset. A recommendation system is an application of machine learning that plays a vital role in providing relevant recommendations, be it a movie recommendation or a product recommendation. In our daily life we depend on recommendations provided by our friends and family or general surveys. Similarly, recommendation systems are tools used to provide logical and rational product recommendations to users that might interest them by using some algorithms.

The Destination Recommendation System filters through enormous data and provides a highly relevant and cogent recommendation based on vital parameters of the tourist dataset. It uses Cosine Similarity algorithm to provide fast and reliable recommendations.

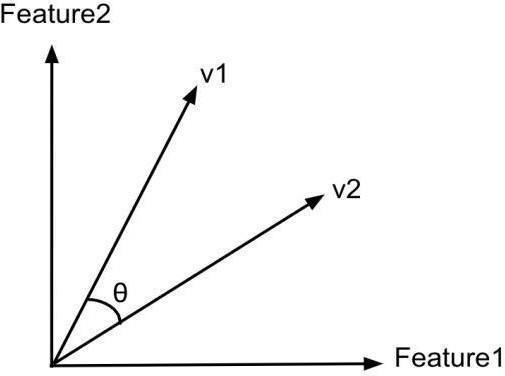


Figure 2.6: Cosine Similarity for Vector Comparison

**Xun Zhou** conducted a study on the use of Singular Value Decomposition (SVD) as a technique for incremental recommendation in recommender systems. The paper emphasizes the importance of incremental recommendation in adapting to the ever-changing user preferences. The authors propose SVD as an efficient method to address this challenge. The study provides an overview of SVD-based methods for recommender systems, covering both traditional and incremental approaches. The Study identifies limitations in traditional recommendation approaches and highlights the advantages of incremental approaches. By utilizing incremental SVD-based techniques, recommender systems can update recommendations in real-time as new data becomes available. This capability enables more accurate and up-to-date recommendations for users. To validate their proposal, the authors present an empirical study that compares two incremental SVD-based approaches: KNN-SVD and Batch-Incremental-SVD. The study evaluates the accuracy and efficiency of these methods and finds that the Batch-Incremental-SVD approach outperforms the KNN-SVD approach in terms of recommendation accuracy and computational efficiency.

In conclusion, the study asserts that incremental SVD-based approaches are effective for recommender systems, delivering accurate and efficient recommendations in real-time. The authors showcase the potential of the Batch Incremental-SVD approach as a promising technique for incremental recommendation. By adopting this approach, recommender systems can enhance their accuracy and efficiency, ultimately improving user satisfaction. In summary, Xun Zhou, et al. demonstrate the significance of incremental recommendation in adapting to evolving user preferences. They propose SVD as an effective method and present empirical evidence supporting the superiority of the Batch-Incremental-SVD approach over the KNN-SVD approach. Overall, the study provides valuable insights into the use of SVD in, paving the way for more accurate and efficient recommendations in real-time scenarios. .

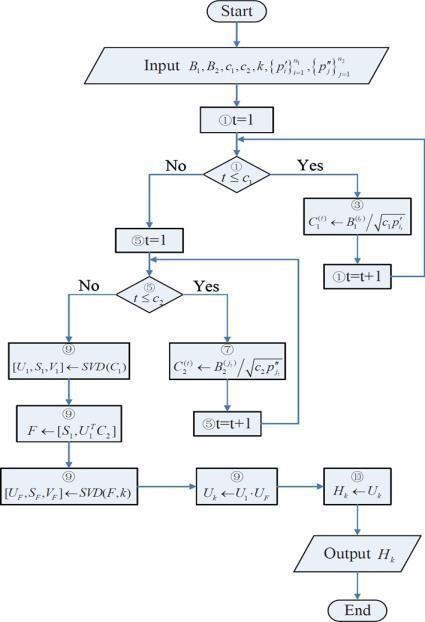


Figure 2.7: Flow Chart of SVD Approach

**Qilong Ba** proposed a clustering-based collaborative filtering (CF) recommendation system using the singular value decomposition (SVD) algorithm. The proposed system aims to address the issues of scalability and sparsity in traditional CF methods. The SVD algorithm is used to factorize the user-item matrix into low-rank matrices, which can be used to predict user- item ratings. However, SVD suffers from scalability issues due to the large size of the user-item matrix. To address scalability issues, the authors propose a clustering- based approach that partitions the user-item matrix into smaller clusters. They use the k-means clustering algorithm to cluster users and items separately, and then construct submatrices for each cluster. The SVD algorithm is then applied to each submatrix to obtain low-rank matrices for each cluster. Experiments were conducted on the Movie Lens dataset to evaluate the proposed method. Results show that the clustering based SVD algorithm outperforms traditional SVD and other CF methods in terms of accuracy and scalability.

**Z K Abdurahman** proposed a tourist places recommender system using two machine learning algorithms, Cosine Similarity and Singular Value Decomposition (SVD). The system aims to assist tourists in selecting tourist attractions in a particular city. The system works by first obtaining data from the Internet about the tourist places and then filtering the data to obtain relevant features such as the name of the attraction, location, category, and user ratings. The Cosine Similarity algorithm is used to determine the similarity between the features of different attractions. The Singular Value Decomposition algorithm is then used to decompose the user-item rating matrix into low-rank matrices, which can reduce the dimensionality of the data and identify latent factors that influence the user's preferences. The system also presents a user interface for the system, which allows users to input their preferences and receive recommendations based on their input. The system was evaluated using real-world data from the TripAdvisor website and the results showed that the proposed system outperformed other traditional recommendation methods such as User-Based Collaborative Filtering and Item-Based Collaborative Filtering.

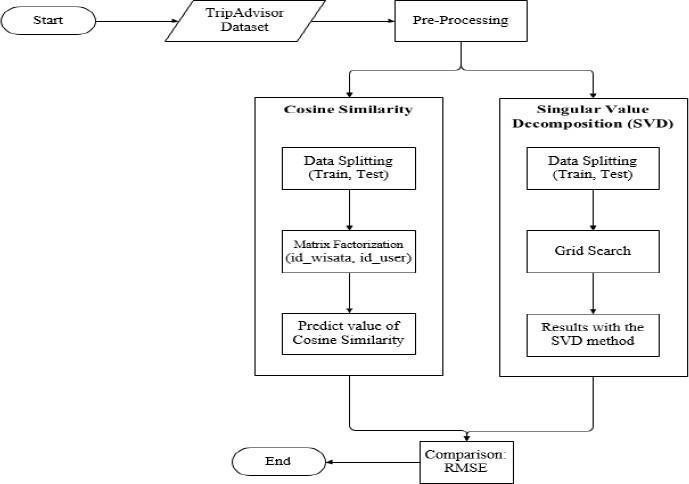


Figure 2.8: System Design for Cosine Similarity and SVD based Approach.

**2.3 Drawbacks of Existing Systems**

* People obtain suggestions for their personal tourism from their friends or travel agencies. The suggestions from friends are limited to those places they have visited before.
* The information from travel agencies (TripAdvisor, MakeMyTrip) is sometime biased since agents tend to recommend businesses they are associated with.
* The prevalence of the Internet provides the possibility for users to learn to plan their tourism by themselves via Internet. However, Internet information is too overwhelming and time consuming.

**2.4 Problem Statement**

**“**To design and develop a system that focuses on personalized travel recommendation.**”**

**Input:** Data Set from Government Tourism Websites of Karnataka.

**Output:** Recommend Tourist spots based on the user’s previous travel history, their preferences, and choices along with the current season.

**2.5** **Proposed System**

* The proposed system aims to provide personalized travel recommendations by collecting user preferences and utilizing machine learning algorithms.
* It aims to enhance the user experience by offering a user-friendly web application developed using Node.js, with features like login, signup, preference questionnaire, rating submission, and displaying recommended places.
* The system aims to leverage the cosine similarity algorithm to calculate similarity scores and recommend the top 10 places based on user preferences and the city database.
* It further incorporates the SVD algorithm to generate an anti-set of unvisited cities and compute average ratings, resulting in the recommendation of the top 6 places.
* The proposed system specifically focuses on the state of Karnataka, utilizing a city database that encompasses the region's attractions, activities, and accommodations.
* By integrating the machine learning code with the web application, the system aims to provide real-time and accurate recommendations, tailored to each user's preferences and ratings.

**Chapter 3**

**3. Requirement Engineering**

**3.1 Software and Hardware Tools Used**

**3.1.1 Software Tools used:**

|  |  |
| --- | --- |
| Python 3 | Python is an interpreter, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. |
| Pip | Pip is a package-management system written in Python used to install and manage software packages. |
| NumPy | NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. |
| Visual  Studio Code | Visual Studio Code, commonly known as VS Code, is a free and open- source code editor developed by Microsoft. It supports a wide range of programming languages and has a rich set of features that make it popular among developers. |
| Mongo DB | MongoDB is a popular open-source, NoSQL document-oriented database. It stores data in flexible, JSON-like documents, allowing for dynamic and agile schema design. |
| Node JS | Node JS is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser. |
| Mongo DB compass | MongoDB Compass is a graphical user interface (GUI) for MongoDB that allows users to visually explore their data, manipulate their schema, and perform various database operations. |

Table 3.1: Software Requirements

**3.1.2 Hardware Requirements:**

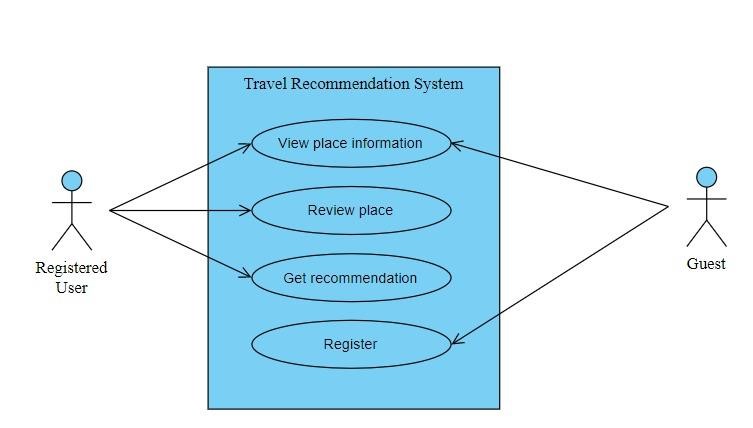
* The Pentium IV or higher
* 25 GB hard free drive space
* 8 GB RAM
* Standard Keyboard and Mouse
* VGA and High-Resolution Monitor

**3.2 Conceptual / Analysis Modeling**

**3.2.1 Use Case Diagram**

A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Figure 3.1 shows the use case diagram is a graphical depiction of user’s possible interactions with a system. This diagram shows various use cases the user and the system have.

Figure 3.1: Use Case Diagram for TRS

In this use case diagram:

* "Registered User" and "Guest" are two types of users.
* Guest needs to register himself in order it become a Registered user.

The system performs the following functions:

* **View Place Information:** Users can access detailed information about different places, images, ratings, and reviews, aiding in informed decision making.
* **Review Place:** Users can share their experiences and provide ratings for specific places, contributing to the overall rating system and helping others make informed choices.
* **Get Recommendation:** Registered users can receive personalized travel recommendations based on their preferences, leveraging machine learning algorithms for tailored suggestions.
* **Register:** Unregistered users can create an account to access additional features, such as saving preferences, submitting reviews, and receiving personalized recommendations.

**3.2.2 Sequence Diagram**

Sequence diagrams can be used to model complex interactions between different components of a system, helping developers to understand the sequence of events that occur during the execution of a particular use case or this scenario.

Figure 3.2 shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between these objects. It shows the interaction between the user and the system to fetch the predictions.

A picture containing text, screenshot, parallel, diagram

Description automatically generated

Figure 3.2: Sequence Diagram for TRS

* **User logs in:**

The user initiates the login process by providing their credentials. The system verifies the user's credentials to authenticate their identity.

* **Display questionnaire for preferences:**

Upon successful login, the system presents a questionnaire to the user for filling their travel preferences. The questionnaire includes various parameters like age, season, travel group type, and other factors.

* **User fills the questionnaire:**

The user fills in the questionnaire by providing their preferences and requirements. They input values for each parameter based on their personal choices.

* **Store values and execute machine learning algorithms:**

The system stores the user's preference values in the database for future reference. Machine learning algorithms, such as cosine similarity and SVD, are executed using the stored preference values and existing data to generate personalized recommendations.

* **Display recommended places:**

The system presents the recommended places to the user based on their filled preferences and the executed machine learning algorithms. The user can view the recommended places on the user interface.

* **User reviews a place:**

The user selects a specific place from the recommended list and proceeds to provide a review. They share their experience, rate the place.

* **Store review in the database:**

The system stores the user's review, rating, and feedback in the database, associating it with the corresponding place and user.

**3.2.3 Activity Diagram**

Activity diagrams can be used to depict the financial and operating activities of network elements in the System Development Process. A process map displays the total control flow. An action diagram obviously presents a movement of exercises or stream of control in a system like a flowchart or a data stream outline.

Activity Diagrams shown in figure 3.3.1 and 3.3.2, visually presents a series of actions or control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

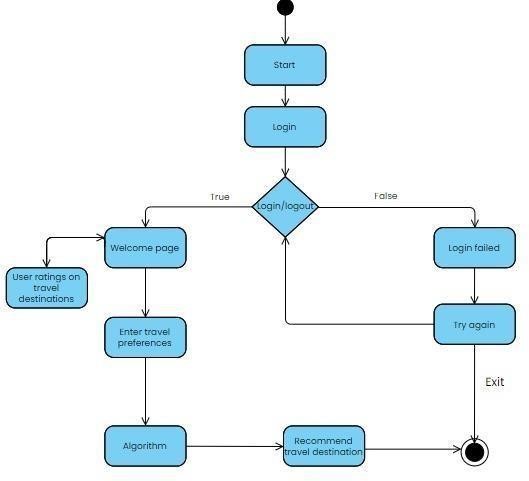


Figure 3.3: Activity Diagram for Registered User

For a registered user, after logging in, they are directed to a welcome page where they can choose to enter ratings for travel destinations or input their travel preferences. The system uses an algorithm to analyse the ratings and preferences of the registered user, generating personalized recommendations for travel places. These recommendations are then displayed to the registered user, providing them with tailored suggestions for their next adventure.

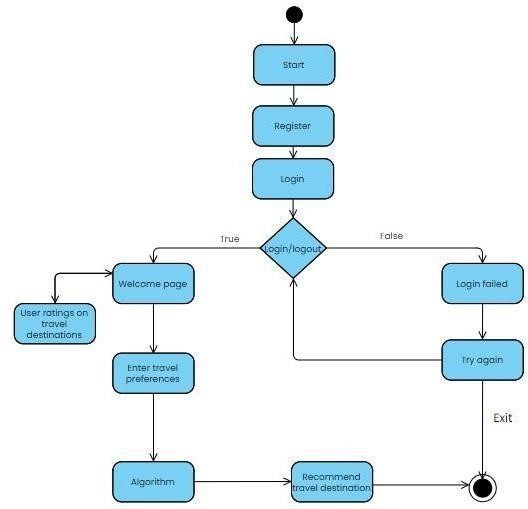


Figure 3.4: Activity Diagram for New User

Upon registering, the user can log in and is directed to a welcome page. From there, they have the option to enter ratings for travel destinations or input their travel preferences. The system employs an algorithm to analyse the ratings and preferences of the new user, generating personalized recommendations for travel places. These recommendations are then displayed to the new user, providing them with tailored suggestions for their upcoming adventures.

**3.2.4 Class Diagram**

Class Diagram in the Unified Modelling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

Figure 3.4 represents a class depict a static view of the model, describing the attributes and behaviour the system has.

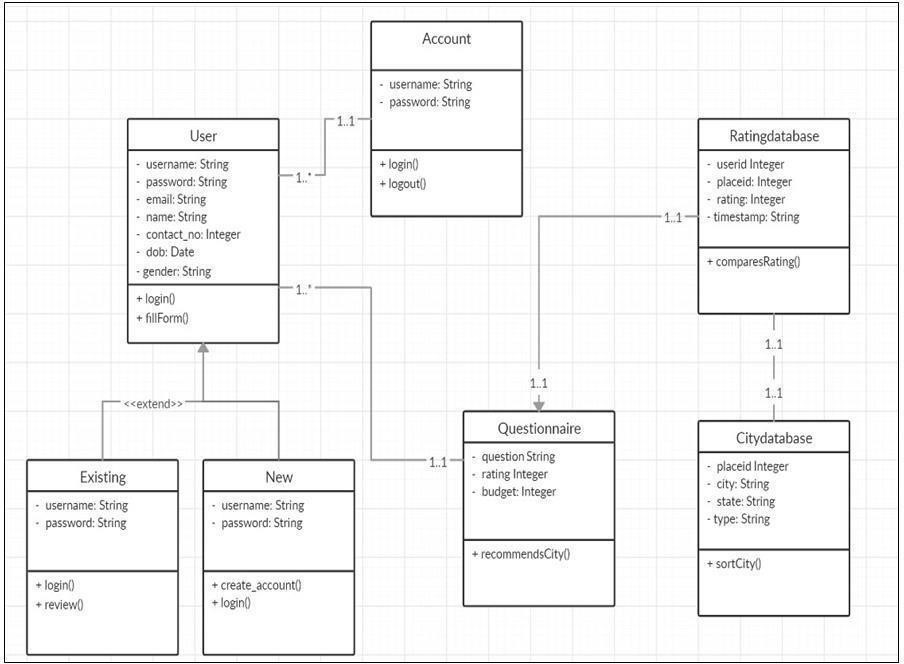


Figure 3.5: Class Diagram

The User class, which represents system users and handles authentication and registration. The Existing and New classes extend the User class and represent existing and newly added place data in the city database, respectively. The Account class manages user accounts, the Questionnaire class captures user preferences, and the City Database and Ratings Database classes handle place and rating data. These classes work together to support user management, preference collection, and the recommendation of personalized travel destinations based on user preferences and the available city database information.

**3.2.5 State Diagram**

A state diagram is a type of diagram used to describe the behaviour of a system. It depicts the various states that a system can be in, and the transitions between those states. State diagrams are often used to model the behaviour of software systems, control systems and communication protocols.

A picture containing text, screenshot, diagram, line

Description automatically generated

Figure 3.6: State Diagram

* **Idle State (Welcome Page):** Represents the initial state of the system when the user first accesses the application. It displays the welcome page or landing page.
* **Authentication State:** Represents the state where the user provides their credentials for authentication, such as username and password, to log into the system.
* **Travel Data State:** Represents the state where the user enters their travel preferences or fills out a questionnaire regarding their desired travel experience, including factors like age, season, and travel group type.
* **Algorithm State:** Represents the state where the machine learning algorithm is executed to calculate the similarity scores or recommendations based on the user's preferences and the available data in the city database.
* **Result State:** Represents the state where the system presents the recommended travel destinations or results to the user based on the executed algorithm. This state may include displaying the top recommended places or providing additional information about each recommended destination.

**3.3 Software Requirements Specification**

A software requirement specification (SRS) is a document that outlines the functions and performance standards for a piece of software. Additionally, it outlines the functionality required for the product to meet the needs of both business and user stakeholders. Functional requirements specify what a system should be able to do through computations, technical details, data manipulation and processing, and other specialized functions. User requirements define how a facility, piece of machinery, or process should operate in relation to the output needed, the conditions in which it should be used, and the product being manufactured.

**3.3.1 Functional Requirements**

* **Data Pre-processing:** Cleaning, transforming, and reducing data to convert raw data in a useful manner.
* **Training:** Learning how to perform the required task based on the inputs given through the dataset.
* **Forecasting:** Making predictions based on the user preferences and user history.

**3.3.2 Non-Functional Requirements**

* **Accuracy:** The system will provide unbiased result with higher accuracy.
* **User Friendly:** The software interface should be easy to use. It should be easy to understand or deal with.
* **Portability:** The system must be possible to run on many systems without making a lot of changes.
* **Reliability:** The system must produce fast and accurate results.
* **Performance:** The accuracy, efficiency, and speed of all the available features should be good.

**3.3.3 Domain requirements**

* The system should be able to collect and maintain the user's preferences, travel history, budget, and other relevant information to provide personalized recommendations.
* A comprehensive and up-to-date database of travel destinations with relevant information such as climate, attractions, accommodation, and reviews are required.
* The system should have an efficient and effective recommendation algorithm that considers user preferences, past behaviour, and other relevant factors to suggest suitable travel destinations.
* A user-friendly interface that enables users to input their preferences, view recommended destinations, and customize recommendations based on their needs is necessary.
* The system should have a mechanism for collecting feedback and ratings from users on recommended destinations to continuously improve the recommendation algorithm.

**Chapter 4**

**4. Project Planning**

**4.1 Pert Chart**

* A PERT (Program Evaluation and Review Technique) chart is a project management tool used to plan, schedule, and manage complex projects.
* It is a graphical representation of a project's tasks, milestones, and critical path. Each task is represented as a node, with arrows indicating dependencies between tasks.
* PERT charts are useful for identifying the critical path of a project, which is the sequence of tasks that must be completed on time to ensure that the project is finished on schedule.
* PERT charts help project managers to track progress, identify potential delays and adjust schedules accordingly.
* They are commonly used in industries such as construction, engineering, and software development where projects have many interdependent tasks and a tight deadline.

A picture containing text, handwriting, font, white

Description automatically generated

Figure 4.1: Pert Chart

**Chapter 5**

**5. System Design**

**5.1 System Architecture**



Figure 5.1: System Architecture

**User Interface - Web Browser**: This component provides the user interface for the application. It allows users to input their travel preferences, view recommended travel places, and modify their input parameters. The web browser communicates with the front-end and back-end components to exchange data and retrieve recommendations.

**Web App - Front End** (HTML, CSS, JS): This component is responsible for rendering the user interface of the application in the web browser. It consists of HTML, CSS, and JavaScript code that defines the structure, layout, and behaviour of the application's user interface. The front-end communicates with the back-end component to retrieve data and generate recommendations.

**Web app - back-end** (node.js, application logic, data pre-processing): This component is responsible for processing user input, retrieving data from the database, and generating recommendations using the recommendation engine. It is built using Node.js and includes the application logic and data pre-processing modules described in the architecture. The back end communicates with the front-end and recommendation engine to exchange data and generate recommendations.

**Recommendation Engine (SVD, dataset, ML model):** This component is the core of the recommendation system. It includes the SVD algorithm, which is applied to the ratings dataset to generate predicted ratings for new items. It also includes the similarity calculation module, which calculates the similarity score between the user's profile and each city in the dataset using cosine similarity. The recommendation engine combines the similarity score and predicted rating to generate personalized recommendations for the user.

**Database (storing user data, travel data, recommendation):** This component is responsible for storing user data, travel data, and recommendations. It includes a database management system such as MongoDB, which is used to store and retrieve data efficiently. The database stores the user's travel history, preferences, and input parameters, as well as the dataset of cities and ratings. It also stores the recommended travel places for each user, allowing them to revisit their recommendations later.

**5.2 Interface Design**

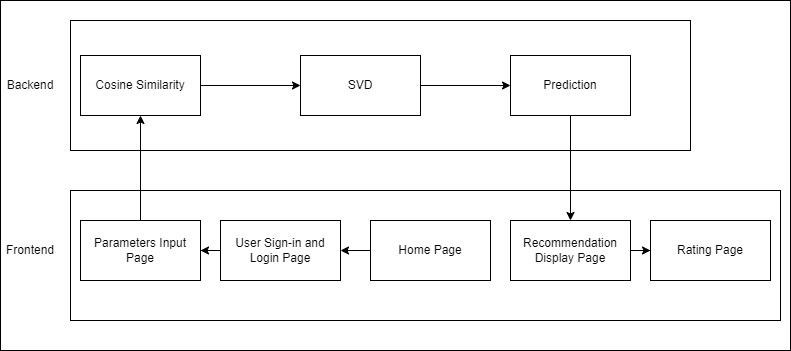


Figure 5.2: Interface Design for TRS

* **Home Page:** The home page serves as the landing page of the web app and provides an overview of the best places to travel. It could include visually appealing images and brief descriptions of popular travel destinations, enticing users to explore more. The interface design should create an engaging and visually appealing layout that encourages users to continue their journey on the web app.
* **User Sign-in and Login Page:** The sign-in and login page is where users can create a new account or log in to their existing account. The interface design should provide a simple and secure way for users to enter their credentials, such as username and password, with clear instructions and error handling for invalid inputs. It should also include options for login or password recovery, if applicable, to enhance user convenience and security.
* **The Parameters Input Page:** The parameters input page is where users can provide initial parameters to the web app to receive personalized travel recommendations. The interface design should include an intuitive and easy-to- use form or input controls for users to enter their preferences, such as destination preferences, travel dates, budget, and other relevant information. It should also provide clear instructions and guidance on how to input the parameters, and visually highlight any mandatory fields or important information.
* **Recommendation Display Page:** The recommendation display page is where the users can view the personalized travel recommendations based on their input parameters. The interface design should present the recommendations in a visually appealing and organized manner, with relevant information such as destination images, descriptions, ratings, and pricing.
* **Rating Page:** The rating page is a feature on a website or app that allows users to give feedback by assigning a numerical or qualitative score, typically out of 5 stars. This page is valuable in collecting user feedback which can be used to improve the quality of recommendation.

**5.3 Data Structure Design**

The data structure design is a method that starts with the specification of the program (what the program does) and leads to the detailed program design expressed in the form of pseudo-code. The data structure used in this project:

* **Data Frame:** A two-dimensional, size-mutable, possibly heterogeneous tabular data structure with labelled axes (rows and columns) is employed, called a Pandas Data Frame. Pandas is used to import a data set from a file into a Data Frame.
* **Dictionary:** A dictionary is a general-purpose data structure for storing a group of objects. A dictionary is an ordered or unordered list of key-element pairs, where keys are used to locate elements in the list.
* **CSV Files:**
* **Cities.csv:** The csv file includes city-specific information such as the city ID, city name, and additional attributes like location and attractions.
* **Preferences.csv:** The csv file captures user preferences, including the user ID, age category, preferred season for travel, preferred travel group type, and any other relevant preferences.
* **Ratings.csv:** The csv file stores user ratings and reviews for specific cities, recording the user ID, city ID, rating, and review text.
* **MongoDB:** For authentication purposes, MongoDB is used. The "Users" collection in MongoDB stores user-related information such as the unique user ID, username, hashed password, and potentially other fields like email or role.

* **Array:** Array is a data structure consisting of a collection of elements, each identified by at least one array index or key. Array is used to store the predicted values of trained model to evaluate performance.
* **List**: Lists are used to store multiple items in a single variable. This is used to store multiple attribute values which are features in dataset.
* **Tuple:** Tuples are used to store multiple items in a single variable, and it can have any number of items that they may be of different types (integer, float, list, string, etc.). A tuple can also be created without using parentheses.

**5.4 Algorithm Design**

**Ensemble Model Algorithm:**

PURPOSE: Providing Personalized Travel Recommendation for a user.

INPUT: User preferences like Interests, Season, Type, Budget, etc.

OUTPUT: Recommended Places**.**

**Algorithm:**

**Step 1:** Load the dataset.

* Load the dataset that contains the relevant information for cities, user preferences, and ratings.
* This dataset serves as the basis for generating recommendations.

**Step 2:** Feature Extraction from the loaded dataset

* Extract the relevant features from the loaded dataset.
* These features may include attributes like city names, user preferences, and ratings.

**Step 3:** Obtaining predictions from Cosine Similarity

* Apply the cosine similarity algorithm to calculate the similarity scores between user preferences and city features.
* These similarity scores help in identifying cities that closely match the user's preferences.

**Step 4:** Model the Training Set for SVD

* Prepare the training set for the Singular Value Decomposition (SVD) algorithm.
* This involves organizing the data, such as user preferences and corresponding ratings, in a suitable format for training the SVD model.

**Step 5:** Obtain a test set for the application of the algorithm.

* Set aside a test dataset to evaluate the performance of the algorithm.
* This test dataset consists of user preferences for which recommendations will be generated.

**Step 6:** Application of the algorithm to the test dataset

* Apply the SVD algorithm to the test dataset to compute the average ratings for the anti-set of cities.
* The anti-set represents the cities that the user has not visited, based on their rated places.
* This step helps in generating personalized recommendations based on the user's preferences and past ratings.

**Step 7:** Display of results

* Display the recommended places to the user based on the results obtained from the algorithm.
* These recommendations consider the similarity scores from cosine similarity and the average ratings from SVD.

**Chapter 6**

**6. Implementation**

**6.1 Implementation Approaches**

**Cosine similarity**: It is a mathematical measure that determines the similarity between two non-zero vectors in a high-dimensional space. It calculates the cosine of the angle between the two vectors, resulting in a value between -1 and 1. A value of 1 indicates that the vectors are identical, a value of 0 indicates that they are orthogonal (i.e., unrelated), and a value of -1 indicates that they are diametrically opposed.

In our travel recommendation system, cosine similarity is used to calculate the similarity score between the user's profile and each city in the dataset. The user's profile is represented as a vector, where each element represents a feature such as interests, preferred season of travel, and age. Similarly, each city in the dataset is also represented as a vector of features. The similarity score between the user's profile and a city is calculated as the cosine of the angle between their respective feature vectors. The higher the cosine similarity score, the more similar the city is to the user's profile. This allows us to recommend travel places that match the user's interests and preferences.

**Singular Value Decomposition (SVD)**: It is a linear algebra technique used to decompose a matrix into three constituent parts: a left singular matrix, a diagonal matrix of singular values, and a right singular matrix. SVD is used in a wide range of applications, including image processing, text analysis, and recommendation systems. Using SVD, the system can predict a user's rating for an item by calculating the dot product of the user's feature vector and the item's feature vector. This prediction is refined by considering the biases of the user and place, such as the tendency of a user to rate places highly or to increase its popularity.

In our travel recommendation system ratings matrix represents the ratings given by users to different travel places. However, the ratings matrix is often sparse and high-dimensional, making it difficult to make accurate predictions for new items. SVD decomposes the ratings matrix into three matrices: a user matrix, a rating matrix, and a travel place matrix. These matrices represent the latent factors that influence the user's rating for a travel place. By reducing the dimensionality of the ratings matrix using SVD, we can find these latent factors and make more accurate predictions for new items. This allows us to recommend travel places that the user has not rated yet but may be interested in based on their previous ratings and preferences.

**6.2 Coding Details**

from surprise import \*

import os

import csv

import sys

import re

from surprise import Dataset

from surprise import Reader

from surprise import accuracy

import numpy as np

import itertools

import pandas as pd

import difflib

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

from collections import defaultdict

user\_params\_keylist=['UserID','Type','Company','Seasons','Age']

user\_params\_valuelist=None

with open('C:/Users/bhargav/Desktop/TraverCity-v6/TraverCity-using Node/py

file/sample\_data/questionnaire.csv', 'r') as file:

reader = csv.reader(file)

last\_row = None

for row in reader:

user\_params\_valuelist = row

user\_params\_valuelist=user\_params\_valuelist[:-1]

user\_params = dict(zip(user\_params\_keylist,user\_params\_valuelist))

travel\_data = pd.read\_csv('C:/Users/bhargav/Desktop/TraverCity-v6/TraverCityusing Node/py file/sample\_data/CitiesData.csv')

selected\_features = ['Type','Seasons','Company','Age',]

combined\_features = travel\_data['Type']+' '+travel\_data['Seasons']+'

'+travel\_data['Company']+' '+travel\_data['Age']

vectorizer = TfidfVectorizer()

feature\_vectors = vectorizer.fit\_transform(combined\_features)

user\_features = ' '.join([user\_params[feature] for feature in selected\_features])

user\_feature\_vector = vectorizer.transform([user\_features])

similarity = cosine\_similarity(feature\_vectors)

similarities = cosine\_similarity(feature\_vectors, user\_feature\_vector)

place\_similarity = [(travel\_data.iloc[idx]['PlaceId'], similarities[idx][0]) for idx

in range(len(similarities))]

place\_similarity = sorted(place\_similarity, key=lambda x: x[1], reverse=True)

class datarel:

citiesId\_to\_name = {}

name\_to\_citiesId = {}

ratingsPath = r'C:/Users/bhargav/Desktop/TraverCity-v6/TraverCity-using

Node/py file/sample\_data/ratings.csv'

citiesPath = r'C:/Users/bhargav/Desktop/TraverCity-v6/TraverCity-using

Node/py file/sample\_data/CitiesData.csv'

def loadCitiesLatest(self):

ratingsDataset = 0

self.citiesId\_to\_name = {}

self.name\_to\_citiesId = {}

reader = Reader(line\_format='user item rating timestamp', sep=',',

skip\_lines=1)

ratingsDataset = Dataset.load\_from\_file(self.ratingsPath, reader=reader)

with open(self.citiesPath, newline='', encoding='ISO-8859-1') as csvfile:

citiesReader = csv.reader(csvfile)

next(citiesReader) #Skip header

for row in citiesReader:

placeId = int(row[0])

city = row[1]

self.citiesId\_to\_name[placeId] = city

self.name\_to\_citiesId[city] = placeId

return ratingsDataset

def assignedrating(self, user):

userRatings = []

hitUser = False

with open(self.ratingsPath, newline='') as csvfile:

ratingReader = csv.reader(csvfile)

next(ratingReader)

for row in ratingReader:

#userID = int(row[0])

userID = row[0]

if (user == userID):

placeId = int(row[1])

rating = float(row[2])

userRatings.append((placeId, rating))

hitUser = True

if (hitUser and (user != userID)):

break

return userRatings

def getcity(self, placeId):

if placeId in self.citiesId\_to\_name:

return self.citiesId\_to\_name[placeId]

else:

return ""

def getplaceId(self, city):

if city in self.name\_to\_citiesId:

return self.name\_to\_citiesId[city]

else:

return 0

def TestSet(testSubject, trainset):

fill = trainset.global\_mean

try:

u = trainset.to\_inner\_uid(str(testSubject))

except ValueError:

anti\_testset = [(str(testSubject), trainset.to\_raw\_iid(i), fill)

for i in trainset.all\_items()]

return anti\_testset

else:

user\_items = set([j for (j, \_) in trainset.ur[u]])

anti\_testset = [(trainset.to\_raw\_uid(u), trainset.to\_raw\_iid(i), fill) for

i in trainset.all\_items() if i not in user\_items]

return anti\_testset

def Get():

testSubject=user\_params['UserID']

ml = datarel()

data = ml.loadCitiesLatest()

trainSet = data.build\_full\_trainset()

algo = SVD()

algo.fit(trainSet)

testSet = TestSet(testSubject, trainSet)

predictions = algo.test(testSet)

recommendations = []

for userID, placeId, actualRating, estimatedRating, \_ in predictions:

intplaceId = int(placeId)

recommendations.append((intplaceId, estimatedRating))

recommendations.sort(key=lambda x: x[1], reverse=True)

similarity\_dict = {place\_id: sim\_score for place\_id, sim\_score in

place\_similarity}

product\_list = [(place\_id, similarity\_dict.get(place\_id, 0) \* est\_rating) for

place\_id, est\_rating in recommendations]

product\_list.sort(key=lambda x: x[1], reverse=True)

final\_recommendation=[]

for ratings in product\_list[:6]:

print(ml.getcity(ratings[0]))

final\_recommendation.append(ml.getcity(ratings[0]))

return None

Get()

**Chapter 7**

**7. Testing**

**7.1 Testing Approach**

Software testing is a process of determining whether the actual software product meets the expected requirements and ensuring that the software product is free of defects. It entails the use of manual or automated tools to execute software/system components to evaluate one or more properties of interest. The goal of software testing is to find mistakes, gaps, or missing requirements in comparison to the actual requirements. Software testing is important because faults or problems in software can be found early and fixed before the software product is delivered. A thoroughly tested software product provides dependability, security, and excellent performance, which saves time, money, and increases customer satisfaction. Performance analysis is a process of evaluating a project's success or failure based on various parameters. It involves studying the performance of a specific scenario and comparing it to the intended objective. When it comes to vulnerability assessment tools, performance analysis entails measuring the tool's time cost, accuracy, and comparing its performance with that of existing applications.

**7.1.1 Unit Testing**

Unit testing is a method for testing software that looks at the smallest testable pieces of code, called units, which are tested for correct operation. By doing unit testing, on can verify that each part of the code, including helper functions that may not be exposed to the user, works correctly and as intended. It checks the correctness of individual Parameter in the dataset. It is implemented on data inputs and outputs of the applied model to verify that the model is working.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Test Case Description** | **Expected Result** | **Actual Result** | **Pass/ Fail** |
| 1 | User input parameters with complete data | Mapping the values to the respective keys  {Interest: Nature, Season:  Winter,} | Mapping the values to the respective keys  {Interest: Nature,  Season: Winter,} | Pass |
| 2 | User input parameters with missing data | Correctly identify the invalid input parameters and return an appropriate error message | TypeError: Cannot read properties of undefined | Fail |
| 3 | Test User login | Authenticate and Login with {email:  "test@example.com", password: "password"} | Successfully logged in | Pass |
| 4 | Cosine similarity calculation | Calculating similarity scores of all the places with respect to the user’s preferences | Tuples with place-id and their respective similarity scores | Pass |
| 5 | Singular Value Decomposition calculation | Get the predicted ratings for unrated places | Matrix containing predicted ratings for unrated places | Pass |
| 6 | Obtaining Ratings from user for visited places | Capturing user ratings | Storing user ratings in ratings.csv file along with his user id | Pass |

Table 7.1: Unit Test Result

**7.1.2 Integration testing**

Integration testing is checking whether the different components work with each other within the machine learning pipeline and produces accurate result. Integration testing is a type of software testing that tests the interactions between different components or modules of a software system to ensure that they function correctly when integrated together.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **Test Data (Input**  **Parameters)** | **Expected**  **Result** | **Actual Result** | **Pass/ Fail** |
| 1 | Integration of Cosine Similarity with SVD | Recommend  cities with high similarity and high ratings | Providing personalized recommendation with  top 6 cities from cities database | Pass |
| 2 | Interests:  Adventure,  Nature, Solo  Traveller,  Summer, Young | Recommend  cities with good rating based on the user's input parameters. | Skandagiri,  Chikmagaluru,  Dandeli | Pass |
| 3 | Interests: Beach, Summer, Solo,  Young | Recommend  cities with good rating based on the user's input parameters | Gokarna,  Murudeshwara,  Skandagiri | Fail |

Table 7.2: Integration Test Result

**Chapter 8**

**8. Results Discussion and Performance Analysis**

**8.1 Results**

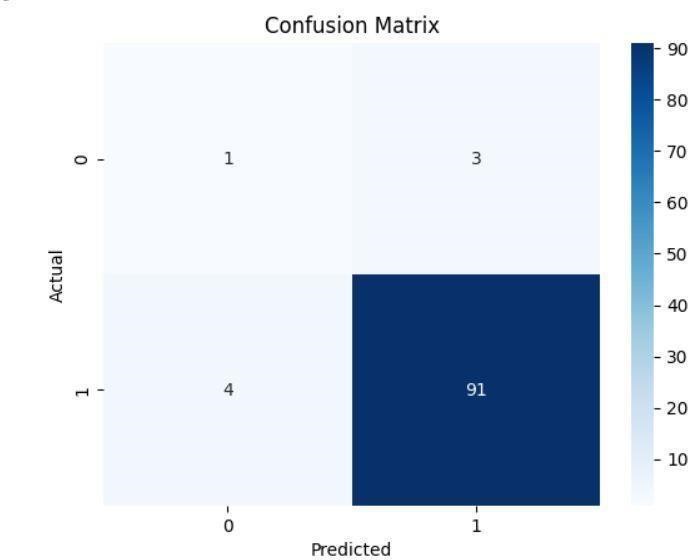


Figure 8.1: Confusion Matrix

Analysing the values in the confusion matrix, you can assess the accuracy and effectiveness of the recommendation system. Metrics such as precision, recall, and accuracy can be derived from the confusion matrix to quantify the system's performance. This evaluation helps identify areas of improvement and refine the recommendation algorithms to provide more accurate and relevant travel recommendations to the users.

A picture containing text, receipt, screenshot, font

Description automatically generated

Figure 8.2: Classification Report

Examining the classification report, you can gain a comprehensive understanding of the system's performance for different classes or categories. It allows you to identify areas where the system excels or requires improvement, enabling you to fine-tune the recommendation algorithms and enhance the overall accuracy and relevance of the travel recommendations.

**8.2 Snapshots**

A screenshot of a website

Description automatically generated with low confidence

Figure 8.3: Home Page

A screenshot of a website

Description automatically generated with medium confidence

Figure 8.4: Services Page

A screenshot of a computer screen

Description automatically generated with low confidence

Figure 8.5: Signup Page

A screenshot of a login page

Description automatically generated with low confidence

Figure 8.6: Login Page

A screenshot of a computer

Description automatically generated with medium confidence

Figure 8.7: Input Recommendation Form

A picture containing text, screenshot, design

Description automatically generated

Figure 8.8: Post-Submission of Form

A picture containing text, screenshot, sky, cloud

Description automatically generated

Figure 8.9: Recommendation Page-1

A picture containing text, website, web page, multimedia software

Description automatically generated

Figure 8.10: Recommendation Page-2

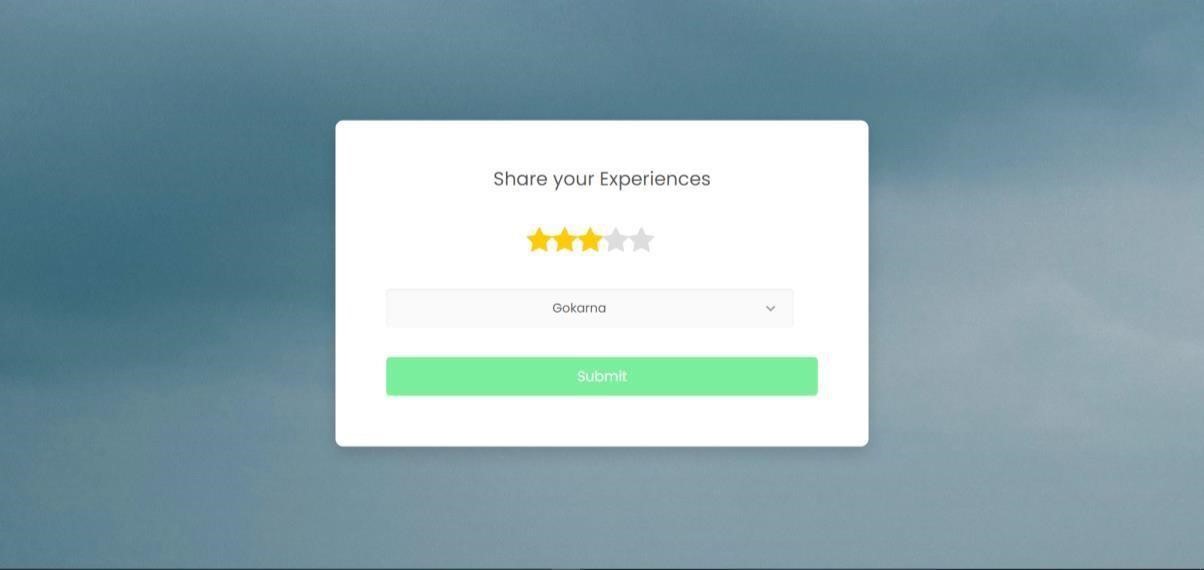


Figure 8.11: Ratings Page-1

A screenshot of a computer

Description automatically generated with low confidence

Figure 8.12: Ratings Page-2

**Chapter 9**

**9. Applications and Conclusion**

**9.1 Applications**

* Travel recommendation systems can help individuals plan personalized travel itineraries based on their preferences, budget, and travel history.
* Travel recommendation systems can be used by tourism boards and travel companies to market destinations, attractions, and experiences to potential customers.
* Travel recommendation systems can help travel companies engage with their customers by providing personalized recommendations based on their preferences and past behaviours. By creating a more personalized and engaging experience, companies can build customer loyalty and increase repeat business.
* It can provide valuable insights into customer behaviour and preferences, which can be used to optimize business operations and marketing strategies.

**9.2 Conclusion**

* The travel recommendation system developed in this project can effectively provide personalized travel recommendations to users based on their interests, preferences, and travel history.
* The system is built on a combination of cosine similarity and SVD, which provides accurate recommendations for users and helps overcome the cold start problem.
* The system has a modular architecture, which allows for easy integration with other systems and future enhancements.
* Overall, the travel recommendation system has the potential to enhance the travel experience for users by providing them with personalized and relevant travel recommendations.

**9.3 Future Scope of the Work**

* **Integration with social media:** We can integrate the system with social media platforms like Instagram and Facebook to allow users to share their travel experiences and recommendations with their friends and followers. This could help to expand the system's reach and improve its accuracy by incorporating more personalized data.
* **Mobile application development:** We can develop a mobile application to make the system more accessible to users on the go. The application could include features like real-time recommendations based on location, user reviews and ratings, and personalized travel itineraries.

* **Integration with booking platforms:** We can integrate the system with popular travel booking platforms like Expedia or Booking.com to enable users to make reservations directly from the recommendation system. This could streamline the travel booking process and provide users with a more comprehensive travel planning experience.
* **Expansion to other domains:** We can consider expanding the system to recommend other domains like restaurants, activities, or events based on user interests and preferences. This could help to make the system more versatile and provide users with a more comprehensive travel planning experience.

**References**

1. Subramaniyaswamy V, Vijayakumar V, Logesh R, and Indragandhi V (2015), “Intelligent travel recommendation system by mining attributes from community contributed photos”
2. Vyshnavi Garipelly, Padma Teja Adusumalli, Priyanka Singh, Travel

Recommendation System Using Content and Collaborative Filtering - A Hybrid Approach.

1. Siddharth S. Bhatkande, Roopa G. Hubballi2, “Travel Prediction Based on

Decision Tree Algorithm Using Data Mining Techniques”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 5, May 2016.

1. Kumar, Seth, Gupta, & Shubham (2020). Sentiment-enhanced content-based system for online recommendations and rating prediction. International Journal of Gaming and Computer-Mediated Simulations.
2. Thiengburanathum, Pree, Shuang Cang, and Hongnian Yu. "A decision tree- based recommendation system for tourists." In 2015 21st International Conference on Automation and Computing (ICAC), pp. 1-7. IEEE, 2015.
3. Shashank Jagtap, Sayali Borate, Tourist Destination Recommendation System using Cosine Similarity,2022
4. Siddharth S. Bhatkande, Roopa G. Hubballi2, “Travel Prediction Based on

Decision Tree Algorithm Using Data Mining Techniques”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 5, May 2016.

1. Xun Zhou, Jing He, Guangyan Huang, Yanchun (2019), “SVD-based incremental approaches for recommender systems”, Journal of Computer and System Sciences.
2. Sahukar S. Bhat, Roopa M, “Travel Prediction Based on Decision Tree Algorithm

Using Data Mining Techniques”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 7, Issue 5, May 2019.

1. Zing He, Guangyan Huang, Yanchun (2020), “Clustering Collaborative Filtering

Recommendation System Based on SVD Algorithm”, Journal of Computer and System Sciences.

1. BPS-Statistics Indonesia, "Domestic Tourism Statistics 2019", BPS RI / BPS Statistics Indonesia, 2019.
2. Utami, D. D., Sinaga, E. K., Desiria, M. K., Febriani, N., Prayitno, R. A.,

Department, T., Tinggi, S., & Bandung, P. (2020). “Potential of Smart Tourism Destination in Bandung City”, TEST Engineering and Management.

1. Xie, K. L., Chen, C., & Wu, S. (2016). “Online Consumer Review Factors Affecting Offline Hotel Popularity: Evidence from Trip advisor”, Journal of Travel and Tourism Marketing.
2. Vagliano, I., Monti, D., & Morisio, M. (2017). SemRevRec: “A recommender system based on user reviews and linked data”, CEUR Workshop Proceedings
3. Al-Ghuribi, S. M., & Mohd Noah, S. A. (2019). “Multi-Criteria Review-Based Recommender System-The State of the Art”, IEEE Access
4. Zhao, Q., Zhang, Y., Ma, J., & Duan, Q. (2019). “Factored Item Similarity and

Bayesian Personalized Ranking for Recommendation with Implicit Feedback”, Arabian Journal for Science and Engineering.

1. Hug, N. (2020). “Surprise: A Python library for recommender systems”, Journal of Open-Source Software.
2. Gutflaish, E., Kontorovich, A., Sabato, S., Biller, O., & Sofer, O. (2019).

“Temporal anomaly detection: Calibrating the surprise”, 33rd AAAI Conference on Artificial Intelligence

1. Sageri Fikri Ramadhan, ZK Abdurahman Baizal and Rita Rismala.(2020).

“Lodging Recommendations Using the SparkML Engine ALS and Surprise SVD”, Jurnal Media Informatika Budidarma.

1. Kumar, A., Seth, S., Gupta, S., & Shubham. (2020). “Sentiment-enhanced content-based system for online recommendations and rating prediction”, International Journal of Gaming and Computer-Mediated Simulations.
2. Joy, J., & Renumol, V. G. (2020). “Comparison of generic similarity measures in E-learning content recommender system in cold-start condition”, 2020 IEEE Bombay Section Signature Conference, IBSSC 2020.
3. Protasiewicz, J., Pedrycz, W., Kozłowski, M., Dadas, S., Stanisławek, T., Kopacz, A., & Gałȩzewska, M. (2016). “A recommender system of reviewers and experts in reviewing problems”, Knowledge-Based Systems.