Academia International College

Affiliated to:

Tribhuvan University

Institute of Science and Technology

Project final defense on

Trekking Recommendation System

Submitted to

Academia International College

Gwarko, Lalitpur

Department of IT

Submitted by:

Ajit Maharjan (26475/077)

`Nischal Shrestha (26499/077)

Nirajan Thapa (26498/077)

Under the supervision Of

Mr. Bishwas Mathema

August 2024

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

The project work report entitled 'Trekking Recommendation System" submitted by Ajit Maharjan, Nischal Shrestha and Nirajan Thapa of Academia International College, is prepared under my supervision as per the procedure and format requirements laid by the Faculty of B.Sc. CSIT, Tribhuvan University, as partial fulfillment of the requirements for Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT). We, therefore, recommend the project work report for evaluation.

.....

Mr. Bishwas Mathema
Academia International
College B.Sc. CSIT

Tribhuvan University

Faculty of Computer Science Information Technology

Academia International College

Letter of Approval

This is to certify that this project prepared by Ajit Maharjan, Nischal Shrestha and Nirajan Thapa entitled "Trek Recommendation System" in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

…….. Program Coordinator Academia International College	…….. Project Supervisor Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner

ACKNOWLEDGEMENT

We would like to extend our sincere gratitude to everyone who has supported and guided us throughout the development of this project. First and foremost, we express our heartfelt thanks to our project supervisor, **Mr. Bishwas Mathema**, for his invaluable expertise, encouragement, and constructive feedback. His guidance significantly deepened our understanding of the subject matter and played a crucial role in the successful completion of this research.

We are equally grateful to CSIT Coordinator, Mr. Santosh Rijal, for his motivational support and insightful suggestions, which helped us effectively plan and manage the project's progress. Our appreciation also goes to all the faculty members and non-teaching staff, who provided an ideal environment for carrying out this project seamlessly.
We owe a special debt of gratitude to our institution, Academia International College, for offering the necessary resources and a conducive setting for learning and exploration.
We also extend warm thanks to our friends and peers for their valuable insights, discussions, and encouragement, which significantly contributed to shaping this project.
Lastly, we express our profound appreciation to our families for their unwavering support and motivation, which have been our greatest source of strength throughout this journey.
Thank you all for being an indispensable part of this endeavor.

ABSTRACT

Nepal, renowned for its diverse terrain and breathtaking landscapes, offers some of the most sought-after trekking experiences in the world, including eight of the fourteen 8,000-meter peaks. However, the abundance of trekking routes, each with varying difficulty, cost, and duration, can make selecting an ideal trek overwhelming for travelers. Traditional resources like guidebooks and blogs often provide inconsistent or outdated information, further complicating the decision-making process.

This project introduces the Trekking Recommendation System, a digital solution designed to simplify trek selection by offering personalized recommendations based on user inputs such as budget and available time. The system utilizes a comprehensive dataset of Nepal's trekking destinations, providing detailed insights into trek difficulty, altitude, accommodations, and the best travel seasons. Key features include user-friendly interfaces for login and trek booking, as well as tailored suggestions to enhance the decision-making process.

While the system ensures ease of use and reliable recommendations, its accuracy depends on the quality of the dataset and currently lacks real-time user feedback and integrated payment functionalities. Despite these limitations, the Trekking Recommendation System promises to transform the trekking planning experience, making it more accessible, efficient, and user-centric for adventurers exploring Nepal.

Keywords: Cosine-Similarity Algorithm, Min-Max Scalar, Confusion Matrix, Booking

Table of Contents

List of Figures:

List of Tables

List of Abbreviation

CSS: Cascading Style Sheet

DFD: Data Flow Diagram

ERD: Entity Relation Diagram

FN: False Negative

FP: False Positive

HTML: Hyper Text Markup Language

SVD: Singular Value Decomposition

TN: True Negative

TP: True Positive

TRS: Travel Recommendation System

Chapter 1 : Introduction

1.1 Introduction

Nepal is home to eight of the world's fourteen 8,000-meter peaks, including the Mount Everest. Nepal is world-renowned for its breathtaking landscapes, diverse terrain, and numerous trekking routes. Trekking in Nepal provides a unique opportunity to experience some of the most beautiful landscapes, from the snow-capped mountainous regions to the subtropical forests. While trekking in Nepal offers incredible opportunities for exploration, it also presents several challenges, especially for those unfamiliar with the region. With such a broad range of trekking routes, each with its own level of difficulty, climate, and cost, planning the ideal trek can often be overwhelming.

Traditional sources of trekking information in Nepal, such as guidebooks, blogs, or word-of-mouth recommendations, can often be outdated, inconsistent, or incomplete. To address these challenges and ensure a safer and more enjoyable trekking experience **Trekking Recommendation System** aims to provide trekkers with personalized suggestions on trekking destinations, based on their budgets and number of days they want to travel. The system will recommend on the basis rich dataset of Nepal’s trekking destinations, considering factors like cost and time while giving information about difficulty, accommodations, max altitude and best time to travel.

1.2 Problem Statement

Trekking in Nepal offers a wide array of destinations, but choosing the right trek can be overwhelming due to the vast variety of options available. Many trekkers struggle to identify the most suitable routes based on their specific needs, such as budget, available time, and desired experience. Existing resources, including trekking guides and blogs, do not provide a personalized approach to help trekkers make these decisions easily.

A major challenge is that trekkers often cannot find destinations that match both their budget and the number of days they have available for the trek. For example, some trekkers may be looking for a budget-friendly trek that can be completed in just a few days, while others may have a larger budget and more time to explore longer, more challenging trails.

The problem of selecting a suitable trek in Nepal arises due to the following factors:

i. Overwhelming Options: Nepal offers a wide variety of trekking routes, each with unique features, durations, and costs, making it difficult for travelers to choose the best option.
ii. Limited Guidance for Personal Preferences: Existing recommendations often fail to consider individual constraints such as budget, time, and difficulty preferences, leading to generic and less personalized suggestions.
iii. Time-Consuming Research: Travelers spend significant time comparing costs, durations, and other details, which can be frustrating and inefficient.
iv. Lack of User-Centric Tools: There is a lack of digital tools or models that provide trekking recommendations based on specific user inputs.

1.3 Objectives

The primary objective of this report is to develop a trekking recommendation model that transforms the process of selecting treks in Nepal into a seamless and personalized experience. The specific objectives are:

i. To create a simple system that recommends trekking destinations in Nepal based on user inputs like budget and available time.
ii. To provide clear and accurate information about different treks, including trip grade, altitude, accommodations and best travel time.
iii. To help travelers make informed decisions by offering personalized suggestions that match their preferences.
iv. To make the process of booking a recommended trek easier.

1.4 Scope and Limitations

The scope of Trekking Recommendation includes:

i. The system includes user-friendly features like login and signup functionalities, ensuring secure access.
ii. User can input preferred cost and number of days for trekking in the recommendation system.
iii. Recommendation system displays accurate information like difficulty level, altitude, accommodations and best travel time based on those input.
iv. Booking one of the recommended results.

The limitations of Trekking Recommendation include:

i. Data Dependency: The accuracy of the recommendations depends on the quality and completeness of the trekking dataset. Any outdated or incomplete data may affect the results.
ii. Limited Scope: The system considers cost and time as the primary inputs, it may not fully account for other subjective preferences such as difficulty level, crowd levels, or personal fitness levels. Booking can only be done of the recommended treks.
iii. No Real-Time User Feedback system: The system does not adapt based on user feedback or ratings, which could determine the quality of recommendations.
iv. No Payment system: There is no payment system in the project.

1.5 Development Methodology

Agile methodology, known for its iterative approach and flexibility, can be highly effective in the development of a trekking recommendation system for Nepal. The iterative approach allows us to tackle complex problems in smaller, more manageable chunks, which is essential for a project that may involve various scopes, changing data sources, and frequent updates due to changes in objective and resources.

To ensure high quality and minimal risk, agile emphasizes automated testing and constant validation. We write unit tests and integration tests to ensure that the trekking recommendation system works as intended. For example, tests can validate that the recommendation engine correctly filters treks based on cost and trek duration. It ensures that validation and verification of log in and signup are tested. Agile development allows quick changes, if necessary, based on feedback from supervisors or other users.

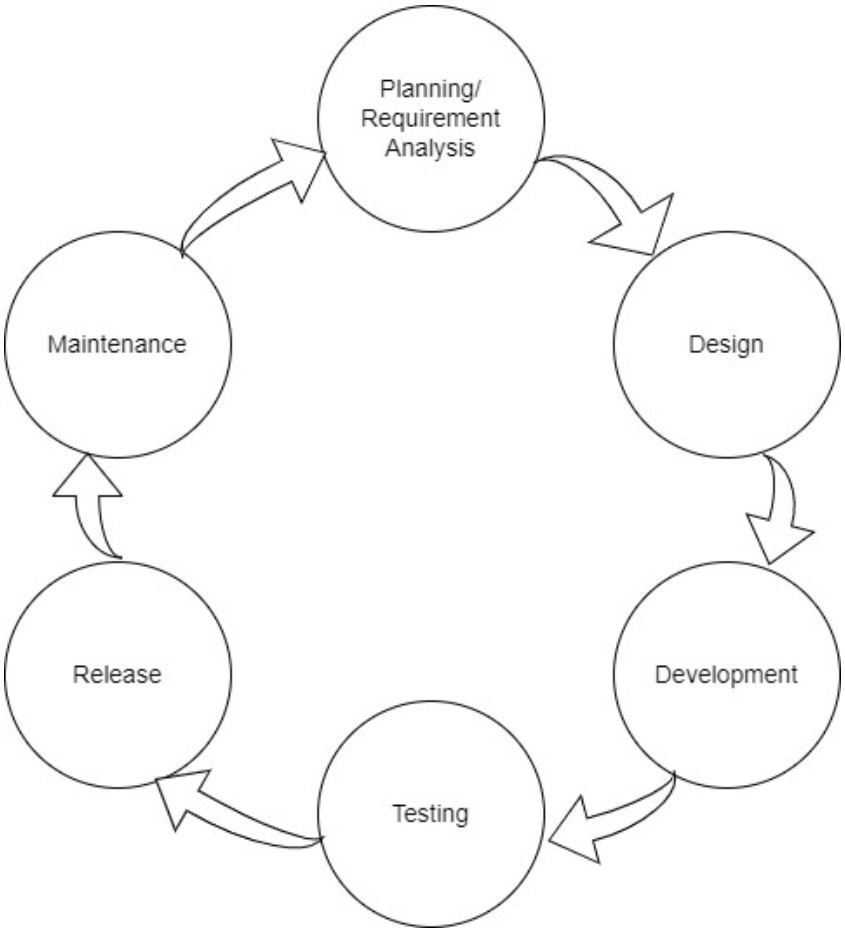


Figure 1.1: Agile Development Model

1.6 Report Organization

The Trekking Recommendation System, a tool created to assist trekkers in choosing trekking locations in Nepal depending on their time and financial constraints, is methodically documented in this study. The following chapters make up the report's structure:

Chapter 1: Overview

This chapter provides an overview of the project, including the system's background, motivation, and importance. It outlines the project goals, the problem statement, the system's scope, the constraints faced, and the development process technique.

Chapter 2: Background study and Literature review

This chapter offers a thorough examination of the basic ideas, jargon, and technological advancements associated with trekking advice. It examines current research papers, algorithms, and trekking suggestion systems, pointing out their advantages and disadvantages while demonstrating the necessity of the suggested method.

Chapter 3: System Analysis The following topics are covered when this chapter dives into the analytical phase: Requirement analysis includes both non-functional (such system performance and usability) and functional (like budget and time inputs, trek suggestions) requirements. Feasibility Study: Evaluation of the system's operational, financial, technological, and schedule viability. System modeling is the process of representing a system using diagrams, such as data flow diagrams (DFDs), entity-relationship diagrams (ERDs), and use case diagrams.

Chapter 4: System Design

The Trekking Recommendation System design is the main topic of this chapter, which covers:

Database Design: The trek database's structure, which includes variables like difficulty, cost, and time. Interface Design: Easy-to-use interfaces for entering data and viewing suggestions.

Algorithm Design: A thorough description of the recommendation-generating cosine similarity algorithm.

System Workflow: Diagrams and flowcharts that show how user input is converted into suggestions.

Chapter 5: Execution and Evaluation

This chapter details the system's implementation and assessment: Implementation: A breakdown of modules including preprocessing, recommendation logic, and web interfaces, as well as specifics about the tools and technologies utilized (such as Python, Django, and Pandas).

Testing: Unit testing, system testing, and result analysis are examples of test cases used to validate the system and guarantee the precision and dependability of recommendations.

Chapter 6: Conclusion and Suggestions for the Future

The Trekking Recommendation System's accomplishments and outcomes are detailed in this chapter. It emphasizes how the system makes journey planning easier and makes suggestions for possible future improvements, including adding real-time data updates, user feedback, or language support.

References

This section lists all the academic papers, books, and online resources referenced during the development and documentation of the project.

Appendices

This section includes supplementary materials, such as screenshots of the system's interface, code snippets of core algorithms, and a log of supervisor meetings and project milestones.

Chapter 2 : Background Study and Literature Review

2.1 Background Study

Trekking in Nepal has a deep-rooted history that goes back to the early 1900s when the country first welcomed foreign visitors. In those days, trekking served as a practical way for traders and locals to transport goods and supplies along mountain trails connecting villages. It wasn't until the 1950s that trekking started attracting foreign adventurers, drawn by the breathtaking beauty of the Himalayas and the opportunity to experience the rich culture of remote mountain communities. [1]

Global platforms like AllTrails, Komoot, and Hiking Project cater to general hiking and trekking enthusiasts, they lack the region-specific customization needed for Nepal's unique trekking landscape. These platforms typically offer basic route details and user reviews but fail to provide the level of personalization and detailed guidance required for treks in Nepal.

Need for a Nepal-Specific Trekking Recommendation System

A trekking recommendation system tailored for Nepal can fill the gaps in existing solutions by:

i. Focusing on Local Data: Leveraging a rich dataset of Nepal's trekking destinations to provide detailed and accurate information.
ii. Personalized User Experience: Allowing trekkers to input preferences such as budget and available time to receive tailored recommendations.
iii. Ease of Use: Incorporating user-friendly features like login/signup functionalities and secure access to personal recommendations.
iv. Booking Integration: Enabling users to directly book recommended treks, making the process convenient and efficient.

2.2 Literature Review

In 2022, Theriana Ayu et al. proposed a Travel Recommendation System (TRS) that used the Cosine Similarity Algorithm combined with content-based filtering. This method excels in delivering accurate recommendations by taking into account various factors that influence user preferences. However, a notable limitation of this approach is its inability to address difference in the rating scales used by different users, which may affect the overall fairness and effectiveness of the recommendations. [2]

Xun Zhou et al., in 2019, introduced an innovative incremental algorithm for recommendation systems based on Singular Value Decomposition (SVD). Their approach integrated the Incremental SVD algorithm with the Approximating Singular Value Decomposition (ApproSVD) algorithm to overcome challenges related to scalability in large datasets. While this combined method successfully addressed some of the scalability concerns, it has its own drawbacks. Constructing an SVD model remains a time-consuming process, and the ApproSVD algorithm struggles with efficiently handling the continuous growth of massive datasets, which can limit its applicability in real-time or rapidly evolving scenarios. [3]

In 2020, Qilong Ba et al. proposed a clustering-based collaborative filtering recommendation system that leveraged the SVD algorithm. Their strategy sought to enhance the accuracy of traditional collaborative filtering methods by incorporating clustering algorithms alongside SVD. This combination aimed to improve the system's ability to identify and recommend relevant content based on user preferences. However, the effectiveness of this approach can be undermined by the presence of biased opinions within the clusters, as such biases may skew the formation of clusters and, consequently, the recommendations generated by the system. [4]

Chapter 3 : System Analysis

3.1 System Analysis

System analysis involves studying the existing systems, identifying user requirements, and determining how the proposed system can address the identified issues. The aim is to design a system that meets user needs effectively while ensuring feasibility and scalability.

3.1.1 Requirement Analysis

The requirement analysis phase identifies the functional and non-functional requirements of the Trekking Recommendation System to ensure it meets user expectations.

i. Functional Requirement

a. User Authentication: Enable users to sign up, log in, and securely access their profiles.
b. User Input: Allow users to specify preferences such as budget and number of days
c. Recommendation Generation: Provide personalized trekking recommendations based on user inputs.
d. Detailed Information: Display trek details such as difficulty, maximum altitude, accommodation types, and best travel time.
e. Booking Feature: Allow users to book treks directly through the system.

The figure below represents a use case diagram, which provides a graphical illustration of the functional requirements of the application. A use case diagram is an essential tool in system analysis and design, as it visually demonstrates how various elements of the system interact with one another. By showing the relationships between actors (users or external systems) and the specific use cases (functionalities or actions) they are involved in, the diagram helps to clarify the scope and boundaries of the system. It is particularly useful in the early stages of development to identify, analyze, and organize system requirements.

Figure 3.1: Use case Diagram

ii. Non-Functional Requirement

a. Performance: The system should provide recommendations quickly, even with large datasets.
b. Scalability: Support a growing number of users and trekking destinations.
c. Reliability: Ensure accurate and up-to-date information is available for all trekking routes.
d. Usability: Offer an intuitive and user-friendly interface.
e. Security: Protect user data with secure authentication and data encryption.

3.1.2 Feasibility Analysis

Feasibility analysis evaluates the practicality of developing and implementing the Trekking Recommendation System by examining its technical, operational, economic, and schedule-related aspects.

i. Technical:

Technology Availability: The system can be developed using readily available tools and technologies such as Python, Django for backend development, basic HTML, CSS and JavaScript for frontend development and SQLite for database management.

Recommendation Algorithms: To deliver personalized trekking suggestions, Cosine Similarity-based Recommendation Algorithm is used. This algorithm ensures that user preferences such as budget and number of days are effectively matched with trekking routes in the database.

ii. Operational

Operational feasibility focuses on how well the proposed Trekking Recommendation System meets user needs, aligns with real-world requirements, and ensures smooth functionality in practice. The system's design emphasizes usability, adaptability, and effective problem-solving, ensuring it is operationally sound. The system offers significant time-saving benefits by reducing the effort needed to compare treks. The system is web-based, ensuring it can be accessed from any device with an internet connection.

iii. Economic

The Trekking Recommendation System is designed to be cost-effective, leveraging free or low-cost tools and resources for development and deployment. The system's implementation and maintenance incur minimal to no significant financial burden due to the strategic use of open-source technologies and existing infrastructure.

Development Costs

The development process uses freely available tools and technologies, eliminating the need for expensive proprietary software or licenses:

Programming Languages and Frameworks:

a. Python (Open-source) for backend development.
b. Django or Flask (Free frameworks) for web application development.
c. HTML, CSS, and JavaScript (Free) for frontend development.

Database Management:

a. SQLite us Free and open-source relational database management system.

Recommendation Algorithm Implementation: Algorithms like cosine similarity can be implemented using free Python libraries such as NumPy, pandas, and scikit-learn.

iv. Schedule

Schedule for project is:

ID	Title	Start Date	End Date	Duration
1	Initiation and Planning	3-Aug	11-Aug	8
2	Proposal Creation	12-Aug	21-Aug	9
3	Front-End development	21-Aug	3-Sep	13
4	Recommendation system creation	4-Sep	20-Oct	46
5	Booking system	20-Oct	28-Nov	39
6	Testing	28-Nov	20-Dec	22
7	Documentation	15-Aug	20-Dec	127

Figure 3.2: Project Schedule

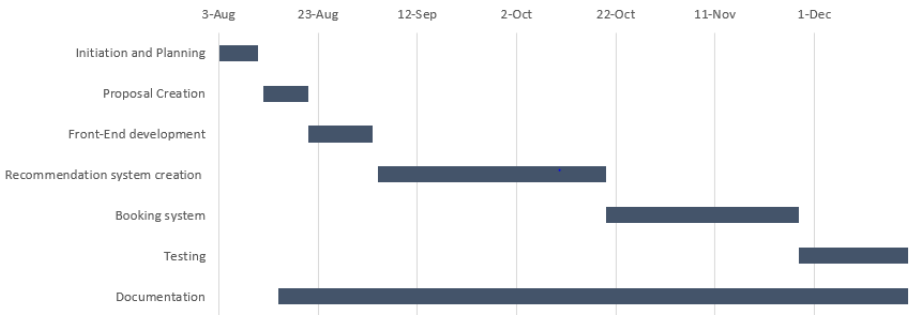


Figure 3.3: Gantt Chart for project schedule

3.1.3 Structured Analysis

i. ER Diagrams

The ER diagram captures the relationships between various entities involved in the trekking recommendation system.

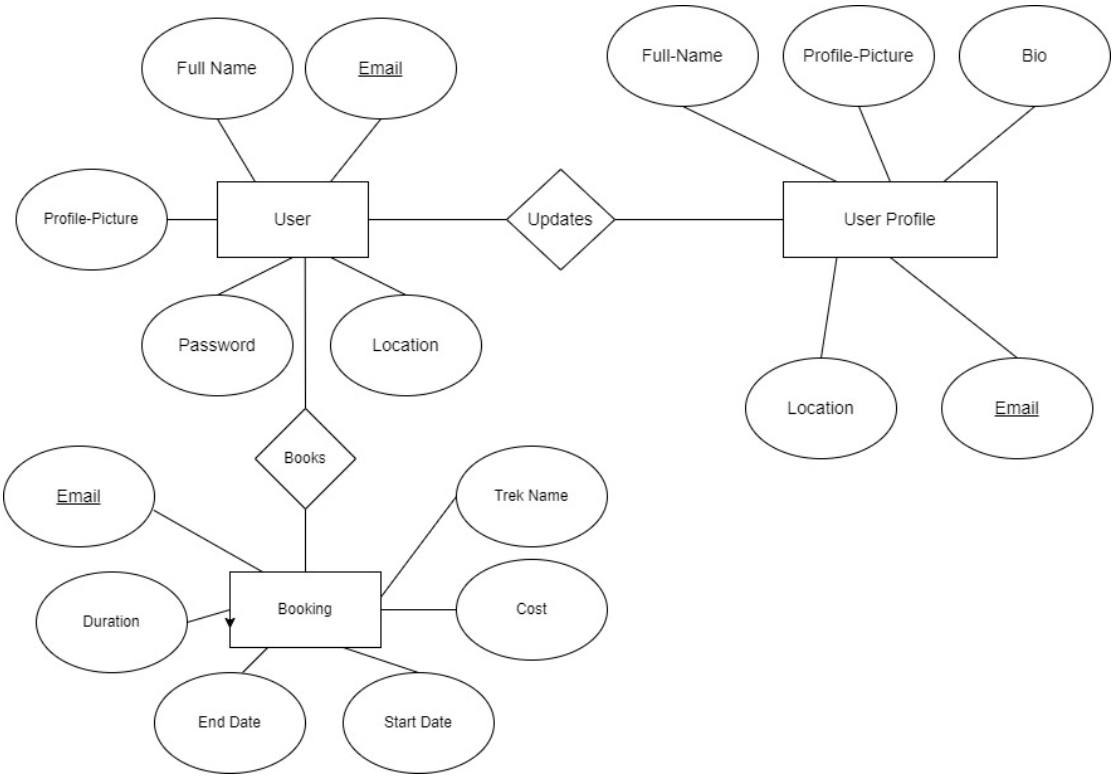


Figure 3.4: ER Diagram

The diagram appears to depict an Entity-Relationship (ER) model for a system involving users, user profiles, and bookings. Here's a brief description:

i. User Entity:

a. Attributes: Full Name, Email, Password, Location, Profile Picture.

b. Relationships:

- Updates: Links to the "User Profile" entity for updating user details.
- Books: Represents a booking relationship with the "Booking" entity.

ii. User Profile Entity:

a. Attributes: Full Name, Email, Profile Picture, Bio, Location.

iii. Booking Entity:

a. Attributes: Trek Name, Cost, Duration, Start Date, End Date.

b. Relationship with the "User" entity includes "Email" as a key connector.

.

ii. DFD Models

DFD Level 0: This diagram provides a high-level view of the system and its interaction with external entities.

The diagram illustrates the flow of interactions within a "Trekking Recommendation" system. It consists of three primary components: the User, the Trekking Recommendation module, and the Trek Data repository.

User Interaction: The user initiates communication by sending a request, which could involve logging in, signing up, or seeking trekking recommendations. This request is directed to the Trekking Recommendation system.

Trekking Recommendation Module: This central component processes the user's request. If the request involves data retrieval or operations such as booking a trek, it interacts with the Trek Data repository to fetch or update the necessary information. Once the operation is complete, it sends the corresponding response back to the user.

Trek Data Repository: This component serves as the database, storing all trek-related information. It responds to data requests or booking operations from the Trekking Recommendation system, enabling the module to fulfill user requests effectively.

Figure 3.5: Data Flow Diagram level 0

DFD level 1: This expands on the context diagram, showing internal processes within the system.

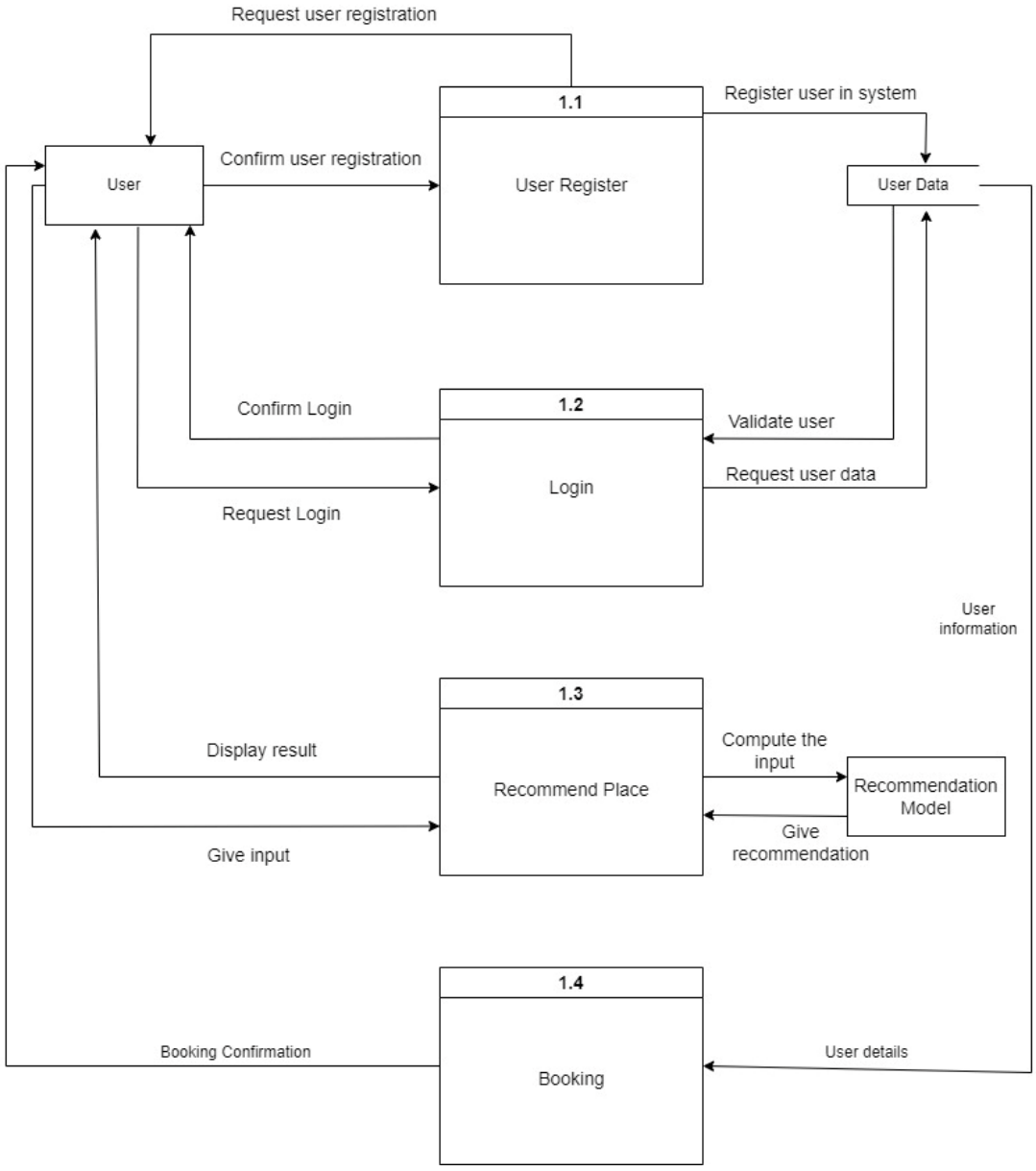


Figure 3.6: Data Flow Diagram level 1

Chapter 4 : System Design

4.1 Design

4.1.1 System Design

The system design outlines the architecture, data flow, and components necessary to implement the Trekking Recommendation System effectively. The design leverages a user-centric approach to ensure accurate recommendations and seamless interaction.

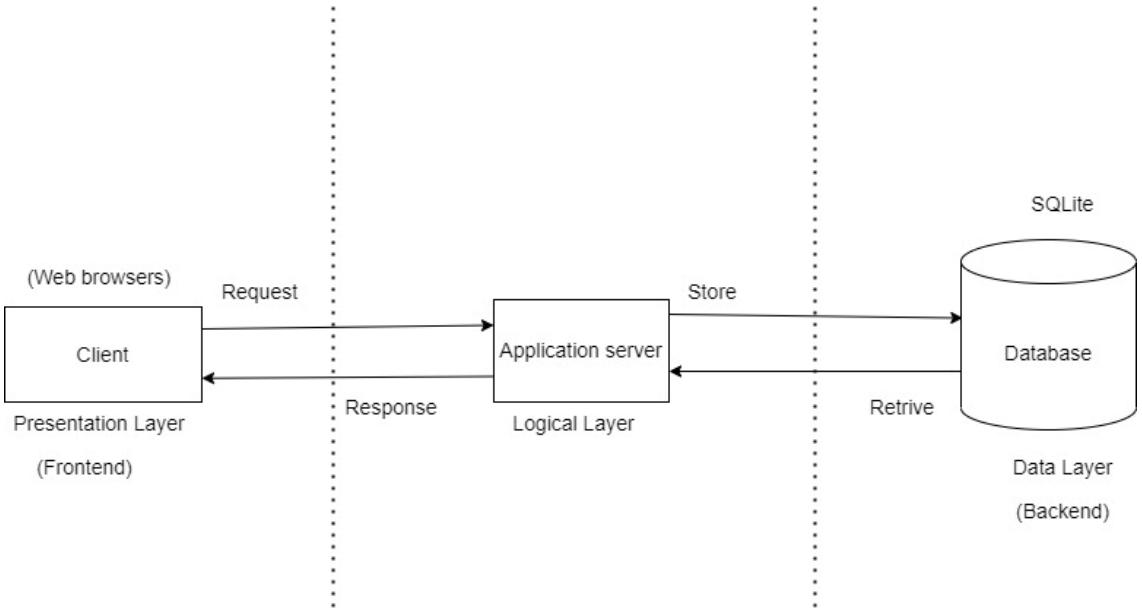


Figure 4.1: System Design

Presentation Layer (Frontend):

i. Represented by the Client (e.g., web browsers).
ii. Handles the user interface and user interactions.
iii. It sends requests to the Application Server (Logical Layer) and receives responses.
iv. Example: When a user submits a form in a browser, the request goes through this layer.

Logical Layer (Application Server):

i. The core of the application that processes requests from the client and interacts with the database.
ii. Contains the business logic of the application.
iii. Stores data received from the Presentation Layer into the database and retrieves data from the database when needed.

iv. Example: A Django server or any backend framework acting as the Application Server.

Data Layer (Backend):

i. Represented by the Database (e.g., SQLite).
ii. Stores, organizes, and retrieves data for the application.
iii. Handles data persistence and ensures that stored information can be retrieved when requested by the Logical Layer.

4.1.2 System Flowchart

This flowchart represents a system for a trekking application:

Start: Users decide whether they are already registered.

i. If not registered, they proceed to Sign Up.
ii. If registered, they Log In.

User Roles:

i. If the user is an admin, they access the Admin Dashboard, where they can Manage User Profiles and Bookings.
ii. If the user is not an admin, they navigate to the Home Page and then the Recommendation Page, where they can enter cost and date to Recommend Treks or proceed to Book a Trek.

Logout: Users can log out at any point.

Stop: The process ends.

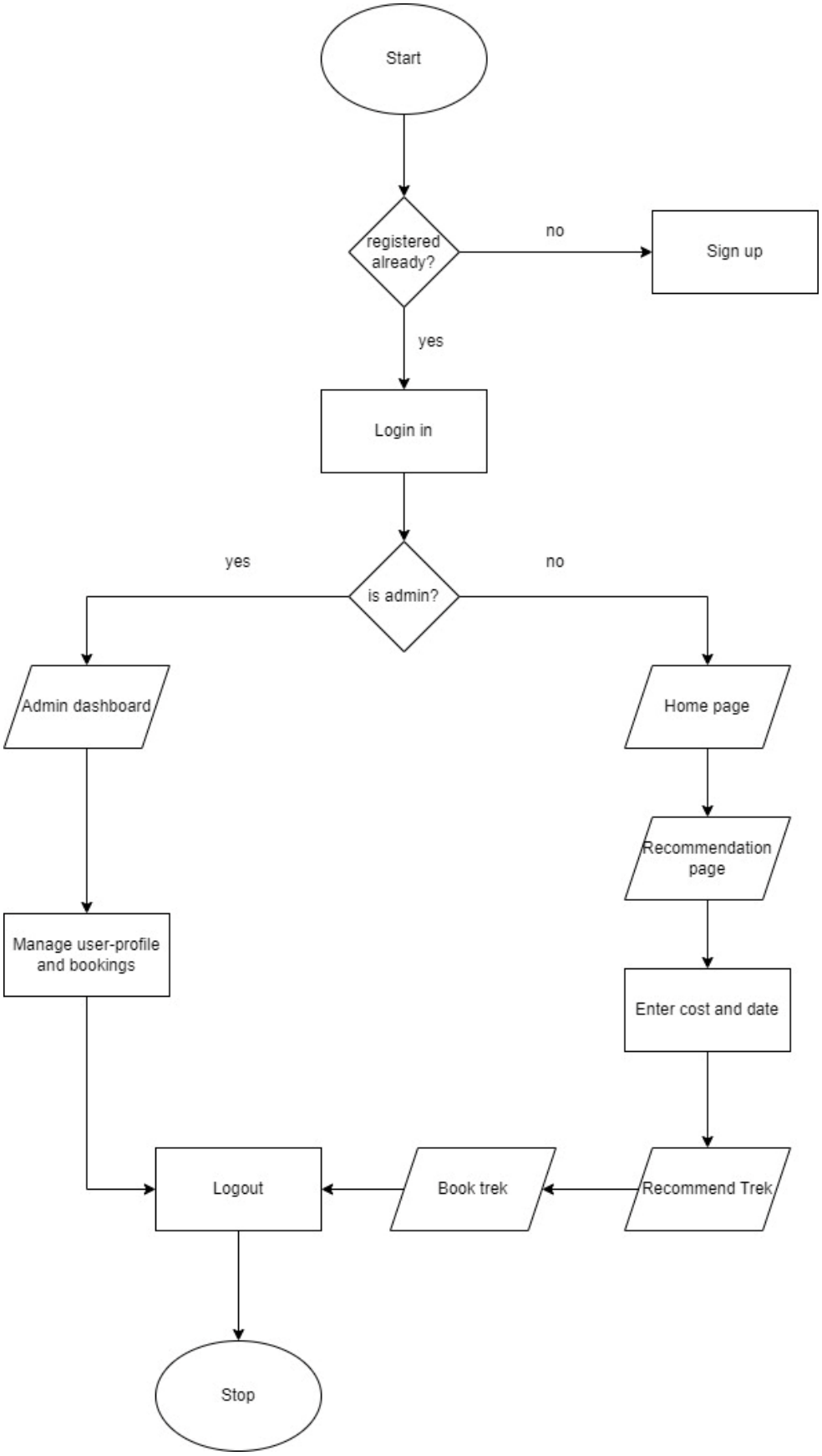


Figure 4.2: System-flow Design

4.1.3 Database Design

The database design for the Trekking Recommendation System (Trekking RS) is structured to handle user information, trek data, booking details, and system interactions effectively. The design ensures data integrity, scalability, and performance.

The diagram represents a database structure for managing users, their profiles, and bookings. The Users table includes basic user details like Full Name, Password, Email, and Location. It is connected to the User Profile table, which provides additional information, such as Bio, User's Picture, and a list of the user's bookings. The Booking table contains details about the user's trekking bookings, including Email (to link to the user), Trek Name, Cost, User Details, Start Date, and End Date. This design facilitates efficient management of user data, profiles, and trekking reservations.

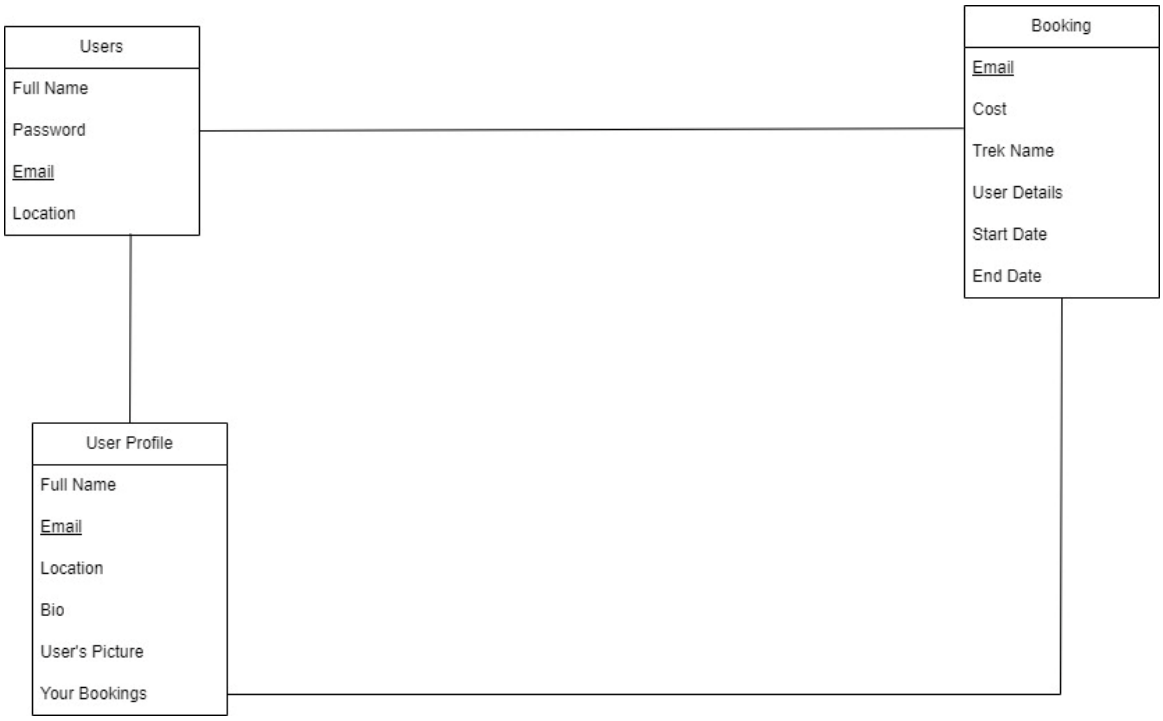


Figure 4.3: Database Design

4.1.4 Forms and Interface Design

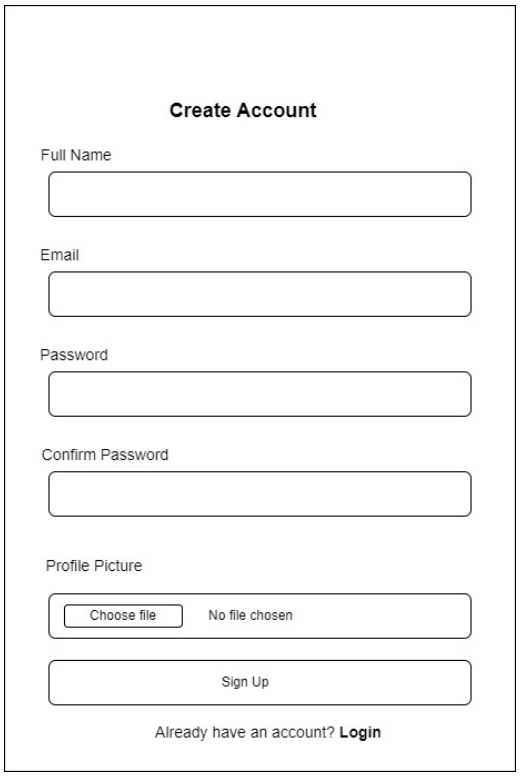


Figure 4.4: Signup Design

Figure 4.5: Login Design

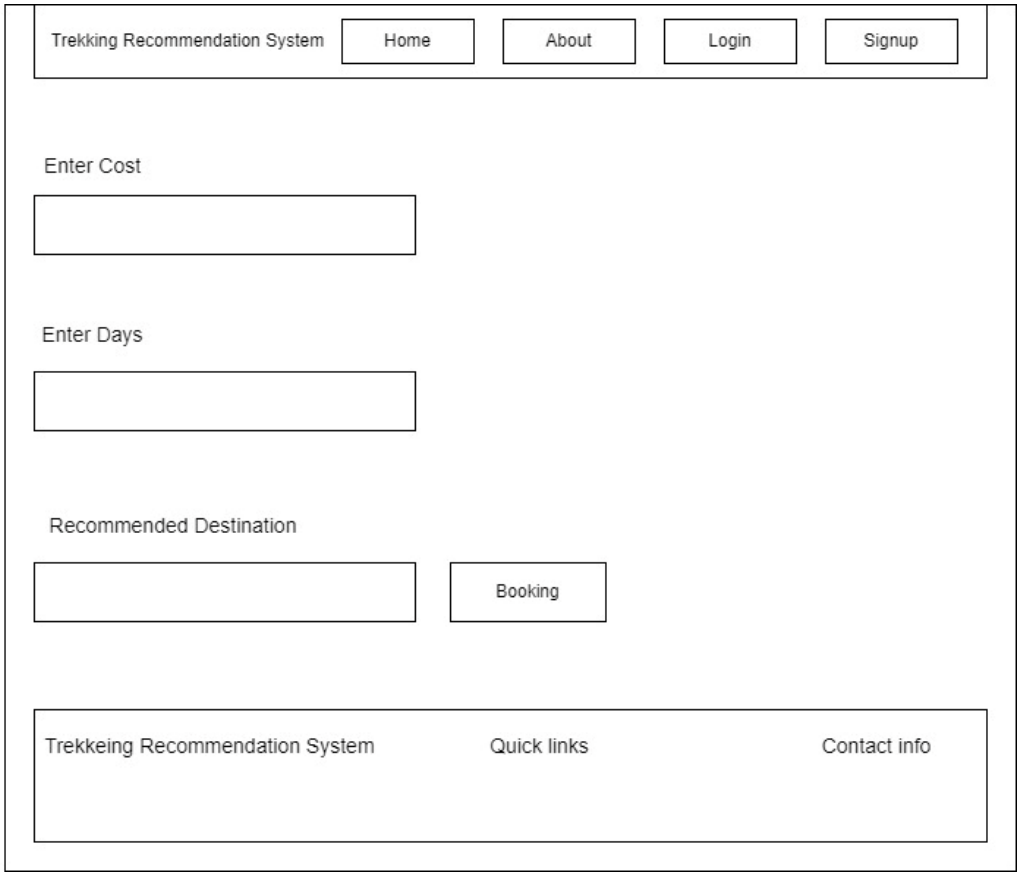


Figure 4.6: Recommendation Page Design

4.2 Algorithm Details

4.2.1 Cosine Similarity with Min-Max Scaling

Cosine similarity is a metric used to measure the similarity between two non-zero vectors in a multi-dimensional space. It calculates the cosine of the angle between the two vectors, which gives a value between -1 and 1. In the context of recommendation systems, cosine similarity is widely used because it focuses on the orientation of the vectors rather than their magnitude. This makes it especially useful when comparing data where the scale or size might vary. [5]

The formula for cosine similarity is:

$$\text{Cosine Similarity} = (\|A\| \times \|B\|)/A \cdot B$$

Where:

i. A and B are the vectors being compared.
ii. A·B is the dot product of the vectors.
iii. ‖A‖ and ‖B‖ are the magnitudes the vectors.

Role of Min-Max Scaling

Raw data often contains values that are on vastly different scales. For example, in a trekking recommendation system, cost may range from a few thousand to hundreds of thousands, while the number of days might range from 1 to 30. If these values are not scaled, larger values will dominate the similarity calculation, skewing the results.

Min-Max Scaling is a normalization technique that transforms features to a fixed range, typically [0, 1]. This ensures that all features contribute equally to the similarity measure.

The formula for Min-Max Scaling is:

$$\text{Normalized Value} = (\text{Original Value} - \text{Min Value})/(\text{Max Value} - \text{Min Value})$$

Where:

i. Value is the original value of the feature.
ii. Min and Max are the minimum and maximum values of the feature in the dataset.

4.2.2 Evaluation Matrix

A confusion matrix is a tabular representation of the performance of a classification model. It evaluates how well the recommended treks align with the ground truth (user's preferred treks).

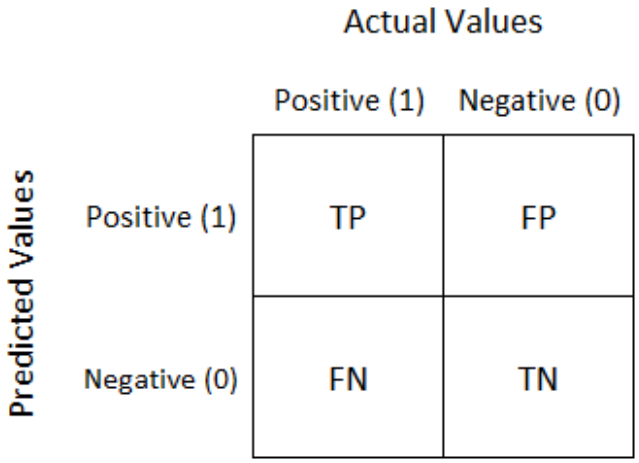


Figure 4.7: Confusion Matrix

Definitions:

True Positive (TP): A recommended trek that the user finds suitable or relevant.

True Negative (TN): A trek that was not recommended and the user agrees it is not suitable.

False Positive (FP): A recommended trek that the user does not find suitable or relevant.

False Negative (FN): A trek that was not recommended but the user would have found suitable.

Evaluation Metrics

Common metrics used for evaluating cosine similarity in a recommendation system include:

Precision: Measures the proportion of recommended items that are relevant.

Formula:

Precision=True Positives/(True Positives + False Positives)

Recall: Measures the proportion of relevant items that are recommended.

Formula:

Recall=True Positives (True Positives+ False Negatives)

F1-Score: Combines precision and recall into a single metric.

Formula:

F1-Score=2·(Precision· Recall)/(Precision+ Recall)

Accuracy: Measures the proportion of correctly identified items (both relevant and non-relevant).

Formula:

Accuracy=(True Positives+ True Negatives)/Total Items

Chapter 5 : Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

Programming Language:

i. Python: Selected for its simplicity and rich ecosystem of libraries for data processing, machine learning, and model evaluation.

ii. Html , CSS and JavaScript : for frontend development

Libraries:

i. Pandas: For data manipulation and preprocessing. Used to clean and structure the trekking dataset.

ii. NumPy: For efficient numerical operations and array handling.

iii. scikit-learn:

iv. MinMaxScaler: To normalize data for consistent feature scaling.

v. cosine_similarity: For calculating the similarity between treks based on user preferences.

vi. confusion_matrix and accuracy_score: For evaluation metrics.

vii. Pickle: For saving and loading the precomputed similarity model and scaling objects.

IDE/Code Editor:

i. Jupyter Notebook and VS Code: For iterative development, visualization, and debugging.

Hardware Requirements:

i. A standard computer or laptop with at least 4 GB RAM for moderate-sized datasets.

ii. Higher RAM and processing power may be needed for larger datasets.

Dataset:

i. Trekking data, including fields like cost, time (days), difficulty, altitude, accommodations, and best travel time.

	A	B	C	D	E	F	G	H
1		Trek	Cost	Time	Trip_Grade	Max _Altitude	Accomodation	Best _Travel_Time
2	0	Everest Ba	NPR 1,20,0	16 Days	Moderate	5545 m	Hotel/Guesthouse	March - May & Sept - Dec
3	1	Everest Ba	NPR 60,00(	14 Days	Moderate	5545 m	Hotel/Guesthouse	March - May & Sept - Dec
4	2	Everest Ba	NPR 3,80,0	12 Days	Moderate	5545 m	Hotel/Guesthouse	March - May & Sept - Dec
5	3	Everest Ba	NPR 2,63,6	11 Days	Moderate	5545 m	Hotel/Guesthouse	March - May & Sept - Dec
6	4	Everest Ba	NPR 1,20,0	20 Days	Moderate	5545 m	Hotel/Guesthouse	March - May & Sept - Dec

Figure 5.1: Dataset for Trekking

5.1.2 Implementation Details of Modules

The User Registration Module is responsible for onboarding new users by collecting essential information such as name, email, and password. This module validates user inputs to ensure data accuracy and uses secure password hashing techniques to store sensitive information. By maintaining a secure database, it prevents unauthorized access and duplicates during registration.

The User Login Module focuses on authenticating registered users. It verifies credentials by comparing the entered email and password against securely stored records in the database. Passwords are encrypted for safety, ensuring user data is protected from breaches. This module ensures that only authorized users can access personalized features like recommendations and bookings.

The Recommendations Module serves as the core functionality of the system, offering personalized trek suggestions based on user-provided inputs such as budget and available time. By utilizing techniques like data normalization (using MinMaxScaler) and cosine similarity, the module evaluates the most suitable treks from a dataset. The results are tailored to match user preferences, providing details such as trek duration, cost, difficulty level, and altitude.

Booking Module streamlines the process of reserving a trek. Users can select a recommended trek and proceed with booking through an intuitive interface. This module handles reservation details, confirmation, and updates to the database, ensuring a smooth and hassle-free experience.

Calculations for cosine similarity:

Minimum Trek Cost = NRs, 50000, Duration = 7 days

Trek A: Cost = NRs 60000, Duration = 9 days

Trek B: Cost = NRs 100000, Duration = 10 days

Trek C: Cost = NRs 150000, Duration = 15 days

The user gives their preferences:

User Input: Cost = NRs 100000, Duration = 10 days

Step 1: Min-Max Scaling

Min-Max scaling normalizes the data to a range between 0 and 1. The formula for min-max scaling is:

Normalized Value = (Original Value−Min Value)/(Max Value −Min Value)

Step 2: Calculate Min-Max Scaled Values

For Cost:

Min cost = NRs 50,000

Max cost = NRs 150,000

Now, calculate the scaled values for each trek and the user input:

Scaled Cost=(Cost−50000)/ (150000−50000)

For Duration:

Min duration = 7 days

Max duration = 15 days

Now, calculate the scaled values for each trek and the user input:

Scaled Duration=(Duration−7)/ (15−7)

Let's now perform the scaling calculations.

Minimum Trek: Scaled Cost = 0.0

Trek A: Scaled Cost = 0.1

Trek B: Scaled Cost = 0.5

Trek C: Scaled Cost = 1.0

User Input: Scaled Cost = 0.5

Scaled Values for Duration:

Minimum Trek: Scaled Duration = 0.0

Trek A: Scaled Duration = 0.25

Trek B: Scaled Duration = 0.375

Trek C: Scaled Duration = 1.0

User Input: Scaled Duration = 0.375

Step 3: Cosine Similarity Calculation:

Now, we can calculate the cosine similarity between the user input and each trek's scaled vector using the following formula:

$$\text{Cosine Similarity} = \frac{U_{\text{cost}} \cdot T_{\text{cost}} + U_{\text{days}} \cdot T_{\text{days}}}{\sqrt{U_{\text{cost}}^2 + U_{\text{days}}^2} \times \sqrt{T_{\text{cost}}^2 + T_{\text{days}}^2}}$$

Where:

- U_{cost}=0.5 and U_{days}=0.375 are cost and days for user input
- T_{cost} and T_{days} are Trek cost and days

We calculate cosine similarity for each trek's scaled vector.

For Trek A (Scaled: Cost = 0.1, Duration = 0.25)

Trek A Cosine Similarity = 0.854

For Trek B (Scaled: Cost = 0.5, Duration = 0.375)

Trek B Cosine Similarity = 1.0

For Trek C: Cosine Similarity = 0.990

Trek C (Scaled: Cost = 1.0, Duration = 1.0)

Step 4: Result

Trek B has the highest cosine similarity of 1.0, meaning it is the most similar to the user's input (Cost: NRs 100,000, Duration: 10 days).

Trek C is also a very close match with a cosine similarity of 0.990.

Trek A has a slightly lower similarity (0.854), indicating it is less similar to the user's preferences.

Thus, Trek B is the best match based on the cosine similarity calculation for the given user input.

5.2 Testing

5.2.1 Test Cases for Unit Testing

Unit tests will be conducted on individual functions or modules in the system. For example, Login and sign-up testing for validation and testing the algorithm with different input combinations (e.g., varying budget and time preferences).

Test case Id	Test case description	Steps to perform	Expected Outcome	Actual Outcome	Status
TC-01	User signup with valid input	Full Name: Ajit Maharjan Email: ajit123@gmail.com Password: 123	signup successfully	As expected	Pass
TC-02	User sign with invalid input	Full Name: user Email: User123 Password: 12345	Signup failed error message	As expected	Pass
TC-03	User Login with valid input	Provide mail: ajit123@gmail.com password: 123	login up Successfully	As expected	Pass
TC-04	Login user with invalid input	Email: user Password: 12345	Login Failed	As expected	Pass

Table 1: Unit Test for Login and signup

Test case Id	Test case	Expected Output	Actual Output	Remarks
TC-01	User inputs budget: NPR 60,000, time: 10 days	Treks that fall within the user's budget and time constraints.	System recommends 3-4 treks matching the user's preferences.	Output is as expected.
TC-02	User inputs budget: NPR 300,000, time: 20 days	Long-duration, high-cost treks suitable for experienced trekkers.	Relevant treks displayed, including additional details.	Matches user expectations.
TC-03	Invalid input (e.g., negative budget)	Error message or request for valid input.	Displays a validation message prompting for valid inputs.	User-friendly handling.

Table 2: Unit test for Recommendation System

5.2.2 Test Cases for System Testing

The system will be tested in an environment that mimics the production setup. System testing will focus on:

i. User Journey Tests: From logging in, inputting trekking preferences, receiving recommendations, and completing a booking.

ii. Performance Testing: Ensuring the system can handle multiple users at once without lag or crashes.

iii. Data Integrity Tests: Ensuring that the trek data (difficulty, altitude, cost, etc.) is consistent across different user requests and recommendations.

Test Case ID	Test Case Name	Preconditions	Test Steps	Expected Result
TC-01	Home Page Layout on Desktop	User accesses the system on a desktop browser.	1. Open the system on a desktop screen. 2. Verify alignment of content and navigation menu.	Layout is properly aligned with readable text and images.
TC-02	Responsiveness of Booking Section	User is browsing the booking section on mobile.	1. Open booking page on mobile. 2. Check form fields and buttons for mobile usability.	Input fields and buttons are legible, tappable, and displayed correctly.

Table 3: Testing for Responsiveness

Test Case ID	Test Case Name	Preconditions	Test Steps	Expected Result
TC-01	Simplicity of Trek Recommendation Process	User is on the trek recommendation page.	1. Input cost and duration preferences. 2.Click "Recommend Trek". 3.Verify recommendation details.	Trek options are relevant, clear, and easy to understand.
TC-02	Book a Recommended Trek	User has selected a trek to book.	1. Click "Book Now". 2. Fill booking details. 3. Submit and verify confirmation message.	Booking is successfully processed, and confirmation appears.
TC-03	Navigation Between Pages	User is navigating through the system.	1. Click various navigation links (Home, About Us, etc.). 2. Ensure smooth page transition.	Pages transition smoothly without errors.

Table 4: Test case for Usability

5.3 Result Analysis

The Trekking Recommendation System was designed to provide personalized trekking suggestions based on user inputs such as budget and duration. This chapter presents an analysis of the results obtained during system testing, evaluates the effectiveness of the recommendation engine, and discusses user feedback. The analysis highlights the system's strengths, identifies limitations, and suggests areas for improvement. The resulted Recommendations can be booked by the logged in user and is saved in their user profile.

Due to imbalanced data set model is incorrectly classifying all negatives as positives. So even though accuracy is 96%, precision, recall, and F1-score are 0.

```
User 0 Recommendations: [233 307 381  85 158 232 306 380  84 228]
Confusion Matrix for User 0:
[[370  10]
 [  3   0]]
Accuracy for User 0: 96.61%
Precision for User 0: 0.00%
Recall for User 0: 0.00%
F1-score for User 0: 0.00%

User 1 Recommendations: [156 304  82 378 230   4  84 380 306 232]
Confusion Matrix for User 1:
[[371  10]
 [  2   0]]
Accuracy for User 1: 96.87%
Precision for User 1: 0.00%
Recall for User 1: 0.00%
F1-score for User 1: 0.00%
```

Figure 5.1: Result Analysis

Chapter 6 : Conclusion and Future Recommendations

6.1 Conclusion

In conclusion, the trekking recommendation system based on Nepal, utilizing cosine similarity to recommend treks, has proven to be an effective tool for matching users with appropriate trekking options based on their budget and time constraints. By considering the cost and duration of each trek, the system can accurately match users to the most suitable treks that align with their preferences, ultimately enhancing the user's experience. The use of cosine similarity ensures that the recommendations are based on both objective and subjective data, making the suggestions personalized and reliable.

The integration of the booking feature into the system further streamlines the user experience by allowing trekkers to book their recommended trek directly, eliminating the need for external booking steps and making the process seamless. This end-to-end solution facilitates trekkers in making informed decisions while ensuring convenience, especially for those unfamiliar with the vast trekking options in Nepal.

6.2 Future Recommendations

Future Recommendations can be as follows:

I. Incorporate User Reviews and Ratings: To further personalize the recommendations, it would be beneficial to include user reviews and ratings. These can provide additional context on the trekking experience.

II. Incorporating Real-time Data: Introducing real-time data regarding weather conditions, trail closures, and availability could make the recommendation system even more dynamic, ensuring trekkers receive up-to-date suggestions that align with current conditions in Nepal.

III. Mobile Application Development: To increase accessibility, developing a mobile application could allow trekkers to use the system on-the-go. This would enable trekkers to access detailed information about treks, track their bookings, and receive recommendations directly from their mobile devices.

IV. Integrate Social Media or Travel Communities: By allowing trekkers to share their experiences or even collaborate with others, the system could provide a community-driven aspect to its recommendations. This social feature could include discussion forums, group treks, and social media integrations to enrich the experience.

REFERENCES

[1] N. T. Planner, "Nepal Trekking Planner," [Online]. Available: https://www.nepaltrekkingplanner.com/blogs/the-history-of-trekking-in-nepal-how-it-all-began.

[2] A. M. A. A. U. A. P. V. K. Abhay Sankar K, "Intelligent Travel Recommendation System," ijraset, 16 06 2023. [Online]. Available: https://www.ijraset.com/research-paper/intelligent-travel-recommendation-system.

[3] X. Zhou, "www.researchgate.net," [Online]. Available: https://www.researchgate.net/publication/269728260_SVD-based_incremental_approaches_for_recommender_systems.

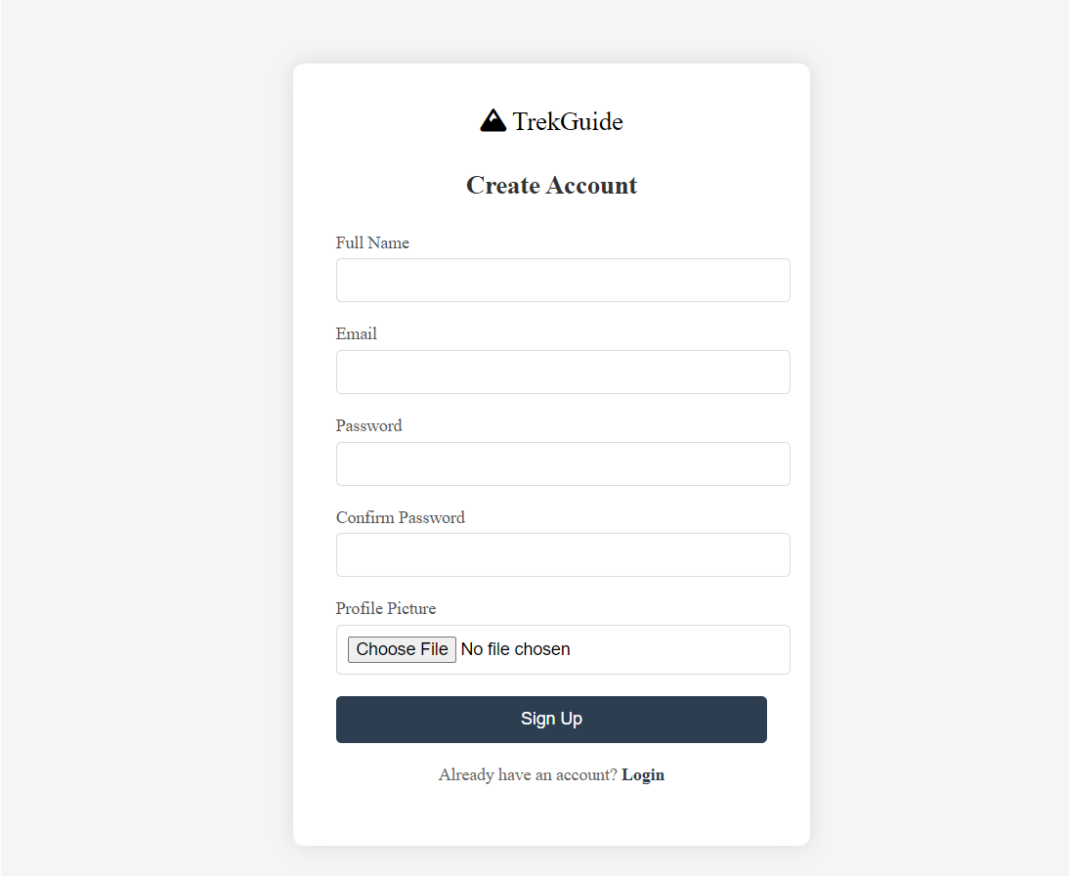
[4] J. &. R. V. G. Joy, 2020. [Online]. Available: https://doi.org/10.1109/IBSSC51096.2020.9332162.

[5] builtin, "builtin," [Online]. Available: https://builtin.com/machine-learning/cosine-similarity.

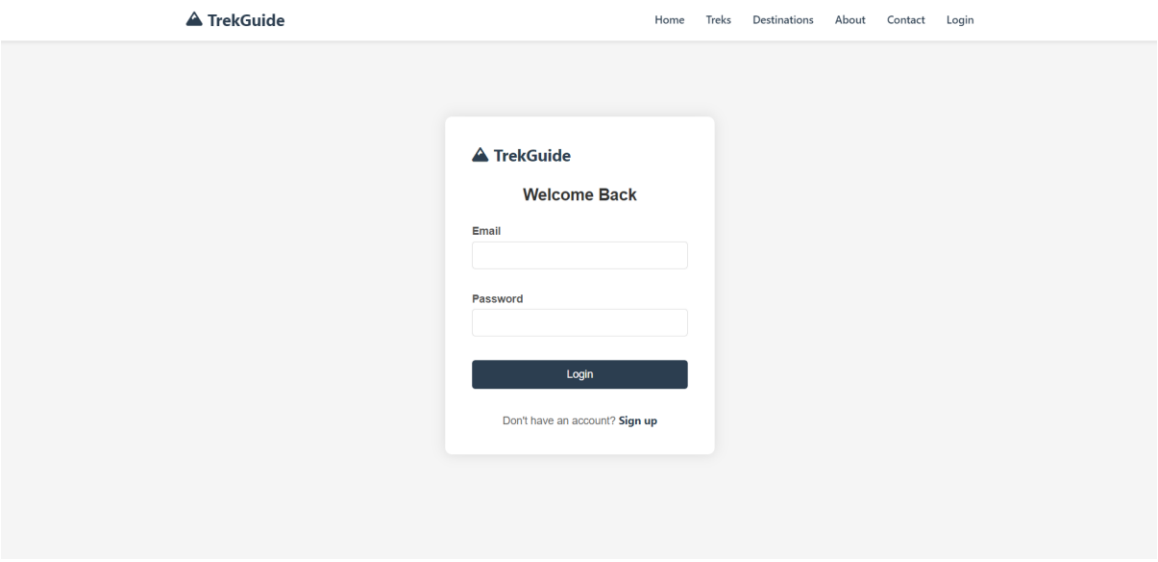
[6] Turing, "turing," [Online]. Available: https://www.turing.com/kb/content-based-filtering-in-recommender-systems.

[7] S. P. R. Asaithambi, "mdpi," 2023. [Online]. Available: https://www.mdpi.com/2227-7080/11/1/28.

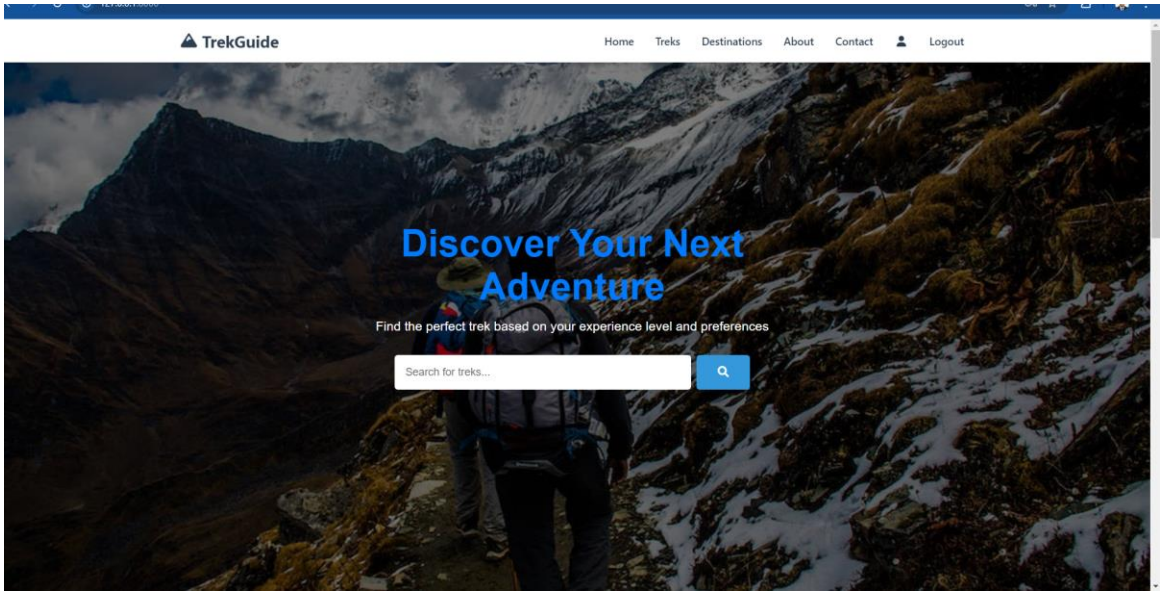
APPENDICES



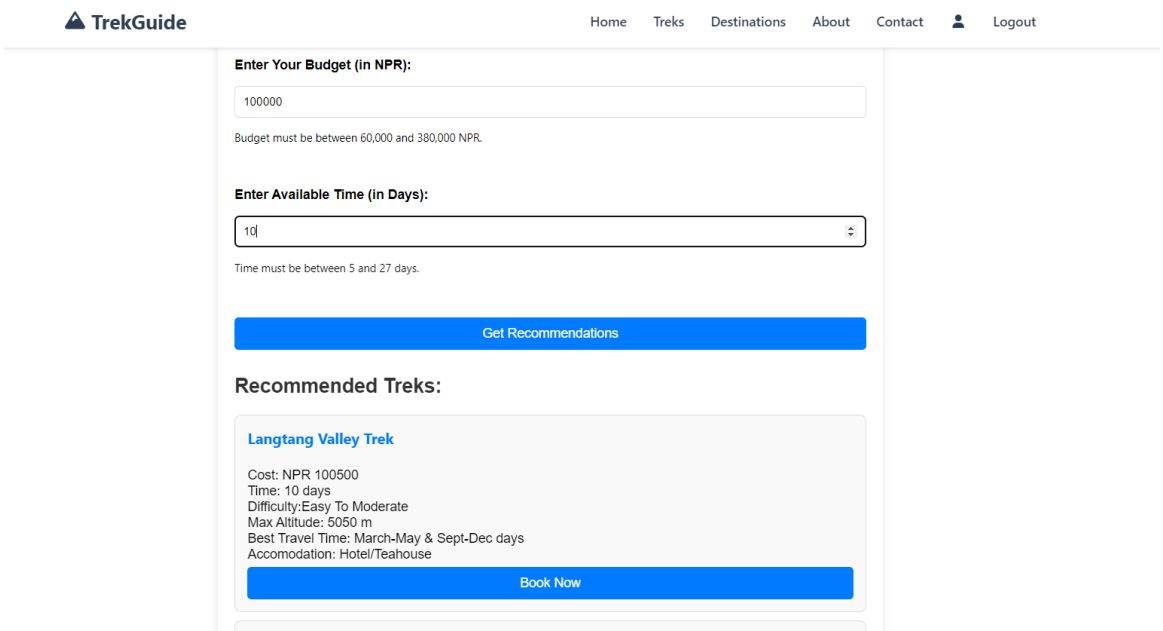
Signup-page



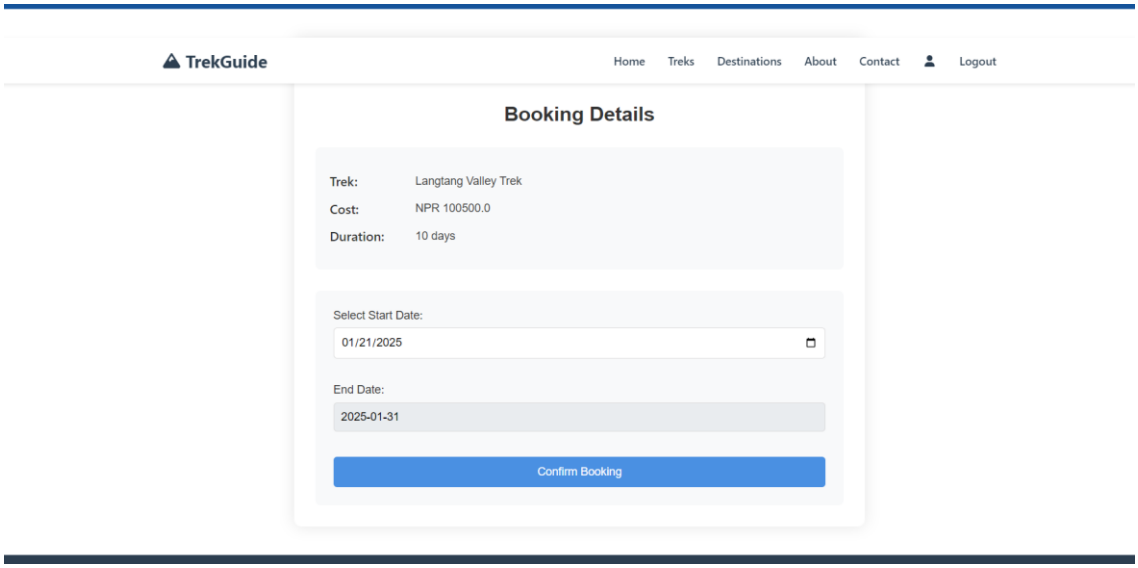
: Login page



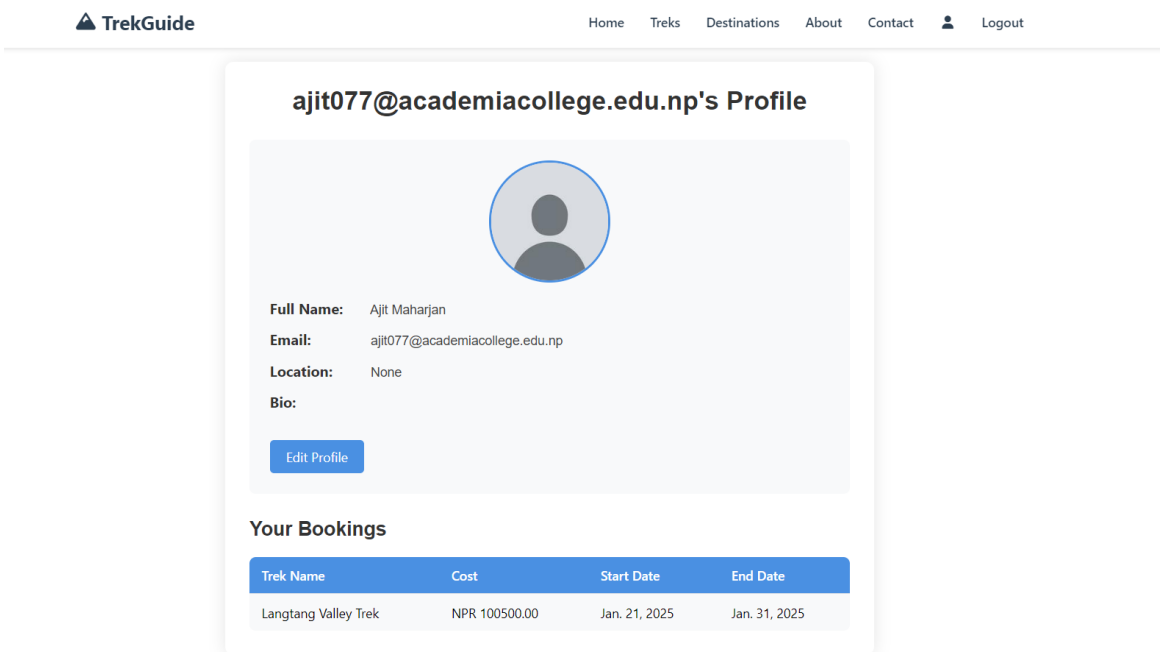
Home page



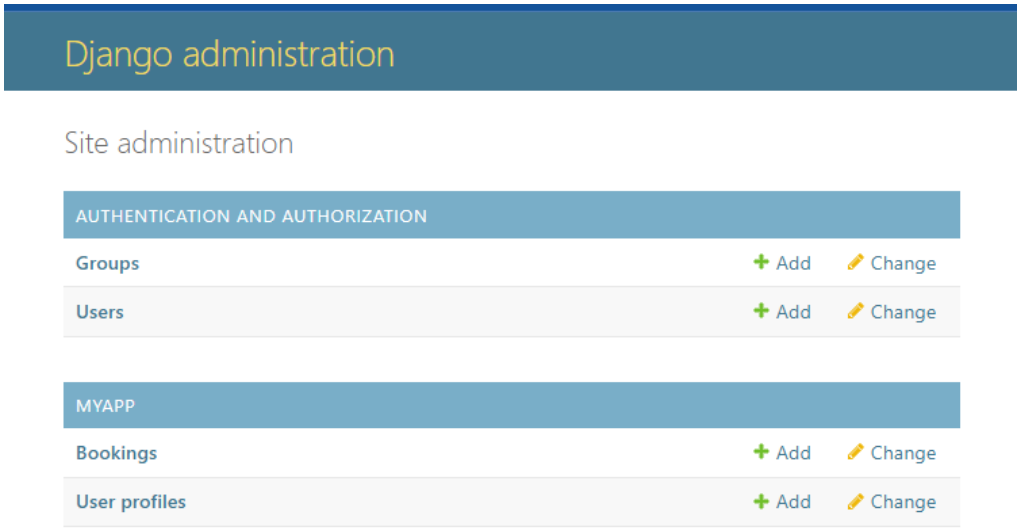
Recommendation page



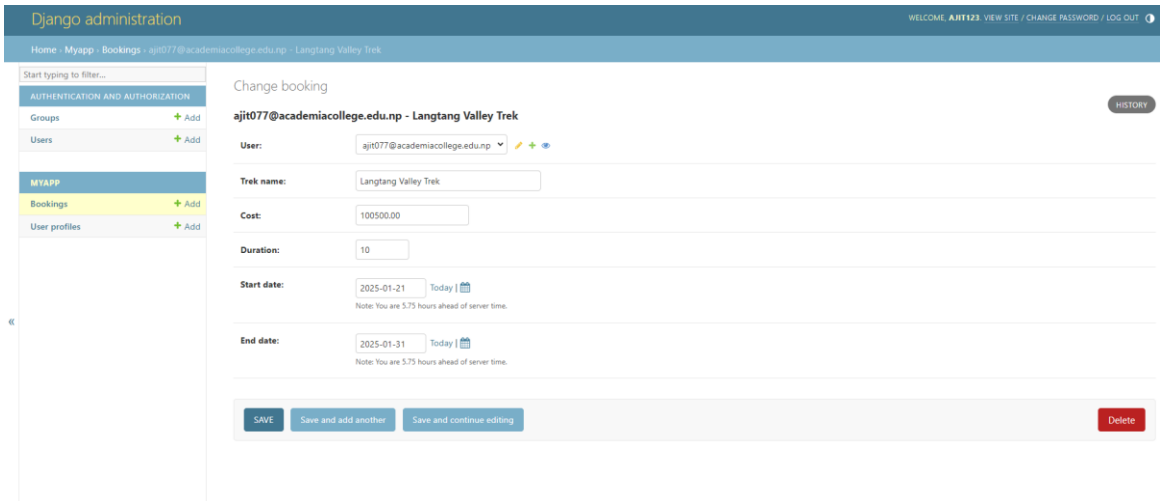
Booking page



User profile page



Django administration page



Booking database page

Source Code

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import MinMaxScaler
import pickle

# Step 1: Load and preprocess the data
def preprocess_data(file_path):
    # Load data
    data = pd.read_csv(file_path)

    # Drop unnecessary columns
    data = data.drop(columns=["Unnamed: 0"], errors='ignore')

    # Clean and convert "Cost" column to numerical format
    data["Cost"] = data["Cost"].str.replace(r"[^\d]", "", regex=True).astype(int)

    # Clean and convert "Time" column to numerical format
    data["Time"] = data["Time"].str.strip().str.replace(" days", "", case=False).astype(int)

    return data

# Step 2: Build recommendation system using cosine similarity
def build_model(data):
    # Normalize "Cost" and "Time" columns
    scaler = MinMaxScaler()
    normalized_data = scaler.fit_transform(data[["Cost", "Time"]])

    # Compute cosine similarity
    similarity_matrix = cosine_similarity(normalized_data)

    return similarity_matrix, scaler

# Main workflow
if __name__ == "__main__":
    # File path
    file_path = "Trek-Data.csv"

    # Preprocess data
    trek_data = preprocess_data(file_path)

    # Build model
    similarity_matrix, scaler = build_model(trek_data)

    # Save model and scaler
    with open("trek_recommendation_model.pkl", "wb") as model_file:
        pickle.dump((similarity_matrix, scaler, trek_data), model_file)
```

```
    print("Model saved successfully!! ")

def recommend_treks(user_cost, user_time, top_n=6):
    # Normalize user input
    user_input = scaler.transform([[user_cost, user_time]])
    # Compute similarity
        user_similarity = cosine_similarity(user_input, scaler.transform(trek_data[["Cost",
"Time"]]))[0]
    # Find top N most similar treks
     recommended_indices = user_similarity.argsort()[-top_n * 2:][::-1]  # Get more entries
initially
    recommendations = trek_data.iloc[recommended_indices]
    # Drop duplicates by Trek name
                                                unique_recommendations                    =
recommendations.drop_duplicates(subset="Trek").head(top_n)
    return unique_recommendations
```