AI Driven Smart Tourism Platform for Personalized Safe & Sustainable Travel Planning

Presenter
R25 -006

Meet Our Team

SRIKANTHAN.S

IT21821240

SENEVIRATHNE S.D.C.D

IT21831768

Thuwakaran. R

IT21835728

Table Of Content

BACKGROUND

- Modern travelers seek highly personalized experiences and seamless itinerary planning tools.
- Artificial Intelligence and real-time data are revolutionizing travel by enabling smarter and tailored experiences.
- Existing platforms lack advanced features like, collabrative travel platform, flexible itinerary management, and effective budget planning.

RESEARCH PROBLEM

01 Existing platforms fail to combine itinerary optimization, and cost management, making it difficult for travelers to access a comprehensive solution in one place.

02 Travelers have limited options to modify their plans dynamically based on changing preferences, unexpected events, or real-time conditions during their journey.

03 Most platforms do not focus on enabling group-based travel planning or connecting travelers with similar interests to collaborate on trips or events.

04 Current tools lack the ability to provide predictive cost estimates and adjust budgets dynamically, preventing travelers from effectively managing their expenses across different activities

OBJECTIVES

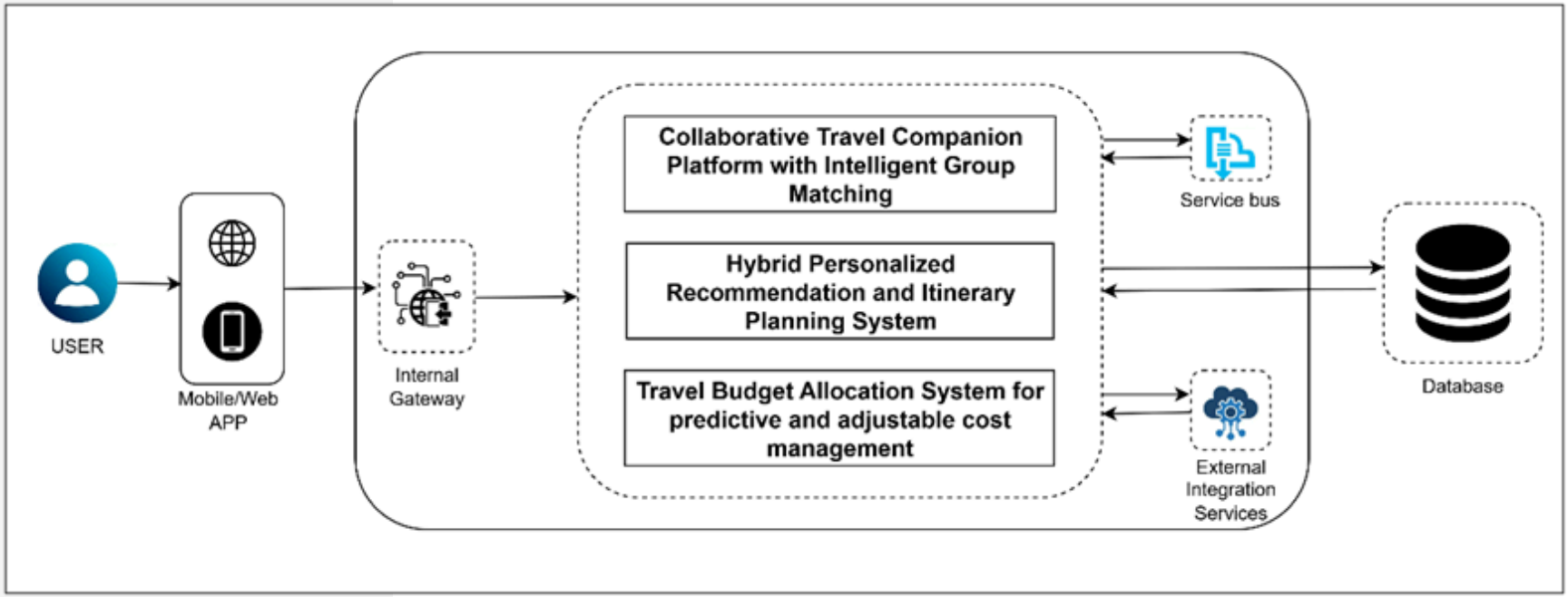
OBJECTIVE 01
Provide a Collaborative Travel Companion Platform for group matching and real-time communication

OBJECTIVE 02
Provide a feature hybrid Personalized Recommendation and itineraries planning system.

OBJECTIVE 03
Provide a feature Travel Budget Allocation System for predictive and adjustable cost management.

OBJECTIVE 04
Providing a web application that ensures performance, reliability, correctness, and excellent UI/UX.

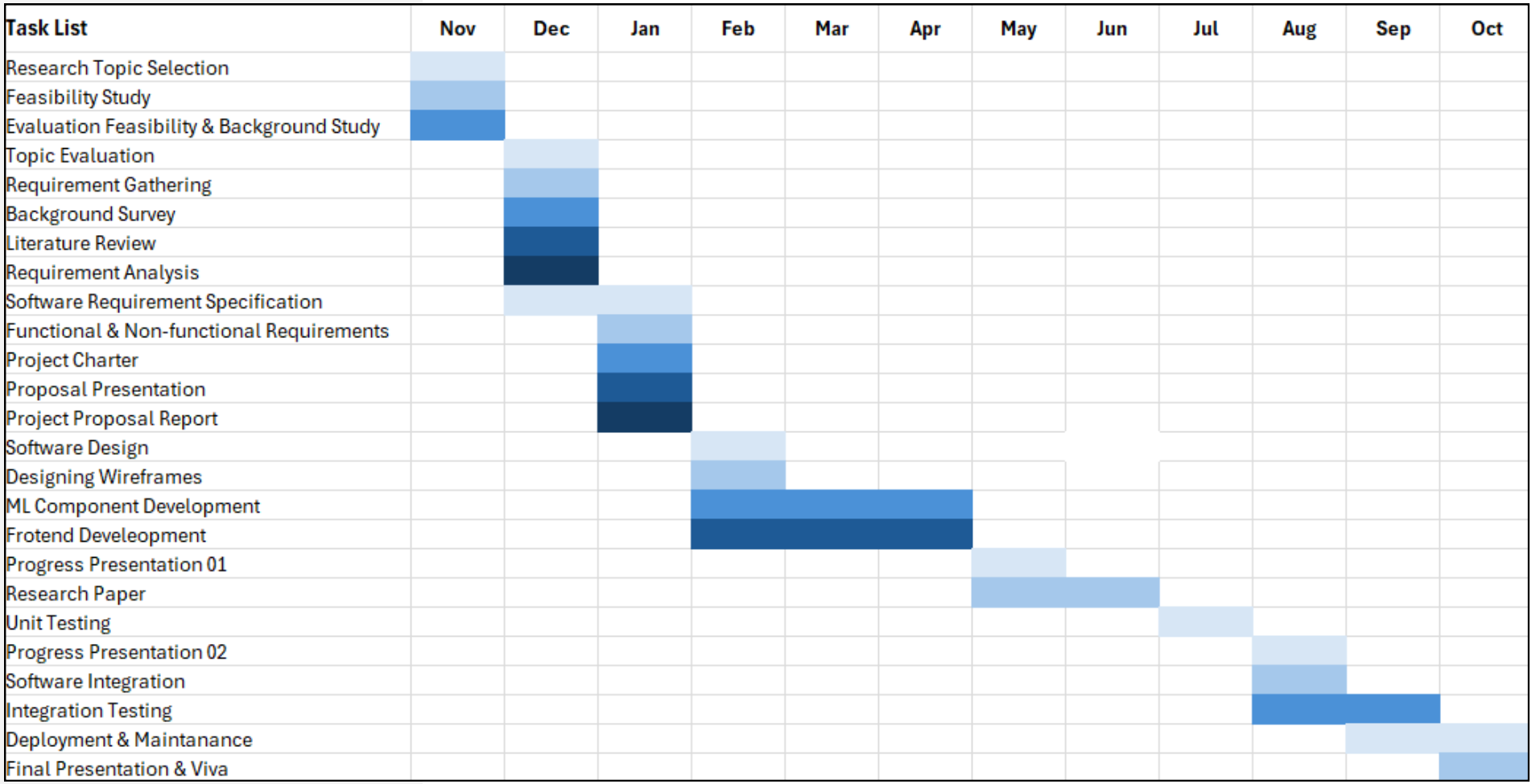
OVERALL SYSTEM DIAGRAM



GENERAL DETAILS

01. GANTT CHART
02. COMMERCIALIZATION PLAN
03. GITHUB DETAILS
04. PROJECT MANAGEMENT
05. RISK MANAGEMENT

GANTT CHART


Task List
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Research Topic Selection
Feasibility Study
Evaluation Feasibility & Background Study
Topic Evaluation
Requirement Gathering
Background Survey
Literature Review
Requirement Analysis
Software Requirement Specification
Functional & Non-functional Requirements
Project Charter
Proposal Presentation
Project Proposal Report
Software Design
Designing Wireframes
ML Component Development
Frotend Develeopment
Progress Presentation 01
Research Paper
Unit Testing
Progress Presentation 02
Software Integration
Integration Testing
Deployment & Maintanance
Final Presentation & Viva

COMMERCIALIZATION

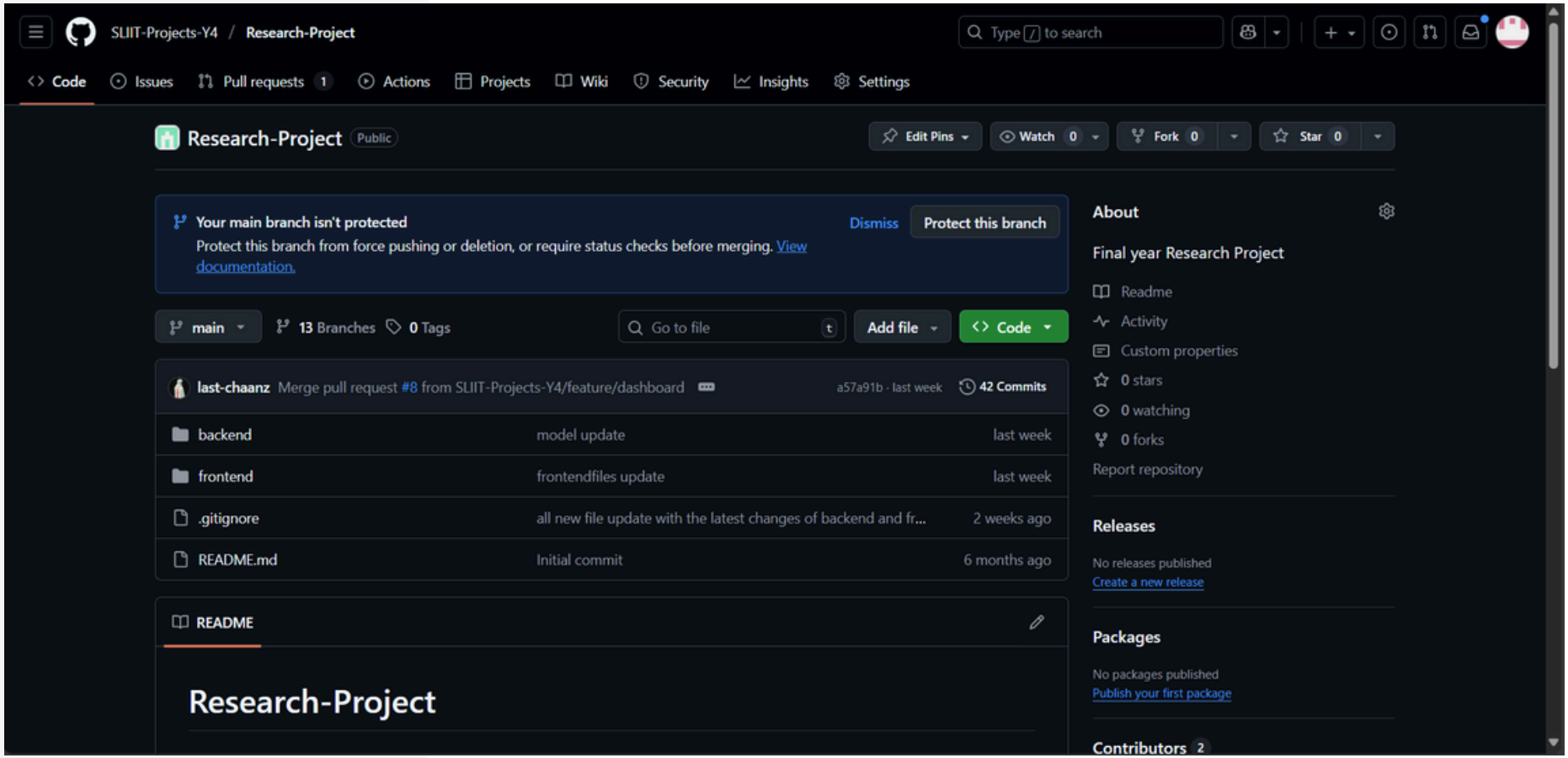
The proposed solution will be integrated with the web application and thses features are very unique

By comparing the competitors these features are new to the travel platform.

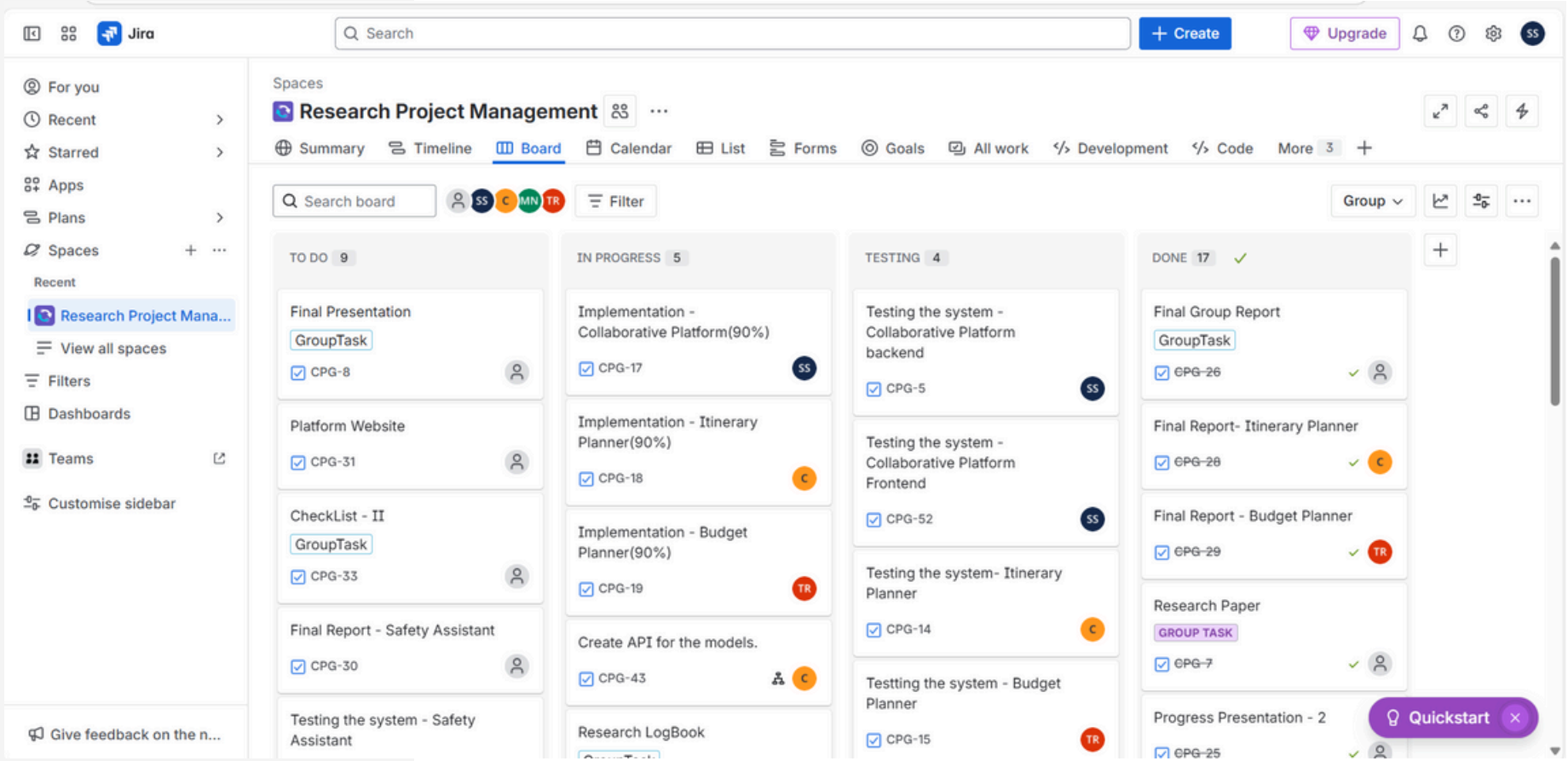
SUBSCRIPTION

ADVERTISING STRATEGY

GITHUB DETAILS



PROJECT MANAGEMENT



RISK MITIGATION

Risk	Trigger	Owner	Response	Resource Required
Risk with respect to the Project Team				
Minor delays in deliverables	Small miscommunications or changes in priority	Project Leader	Discuss with team to identify any small delays. Adjust timelines as needed.	Updated Project Schedule Plan/Gantt Chart
Risk with respect to the Panel/Supervisor(s)				
Slight misalignment on expectations	Minor changes in requirements	Project Leader	Confirm and clarify expectations through quick discussions. Update project documents if necessary.	Meeting Notes Updated Project Plan
Panel preferences shift slightly	Panel's evolving preferences or needs	Project Leader	Seek feedback regularly to stay aligned with expectations. Make small adjustments as needed.	Meeting Logs Updated Deliverables
Risk with respect to Project Execution				
Minor technical hiccups	Occasional bugs or troubleshooting	Project Leader	Allocate time for quick debugging and testing. Work with the team to address issues.	Debugging Tools Technical Documentation

IT21821240 - SRIKANTHAN.S

SPECIALIZING - SOFTWARE ENGINEERING

COLLABORATIVE TRAVEL COMPANION PLATFORM WITH INTELLIGENT GROUP MATCHING

INTRODUCTION

01. BACKGROUND
02. RESEARCH QUESTIONS
03. OBJECTIVES

BACKGROUND

- Travel platforms currently lack personalized group matching based on users' specific preferences.
- Limited tools for real-time coordination of activities and cost-sharing among group members.
- Lack of continuous feedback mechanisms to improve recommendations and user satisfaction.

RESEARCH PROBLEM

01 How can intelligent group matching improve the travel experience for users?

02 What features can facilitate effective real-time coordination among travel group members?

03 How can user-generated itineraries and shared experiences help others with similar preferences?

04 How can user groups support travellers in finding suitable travel buddies and planning cost-effective trips?

OBJECTIVES

OBJECTIVE 01
To match users with similar travel interests, styles, destinations, and budget preferences into compatible travel groups.

OBJECTIVE 02
To recommend existing travel groups that best suit the user's profile, avoiding the need to create new groups unless necessary.

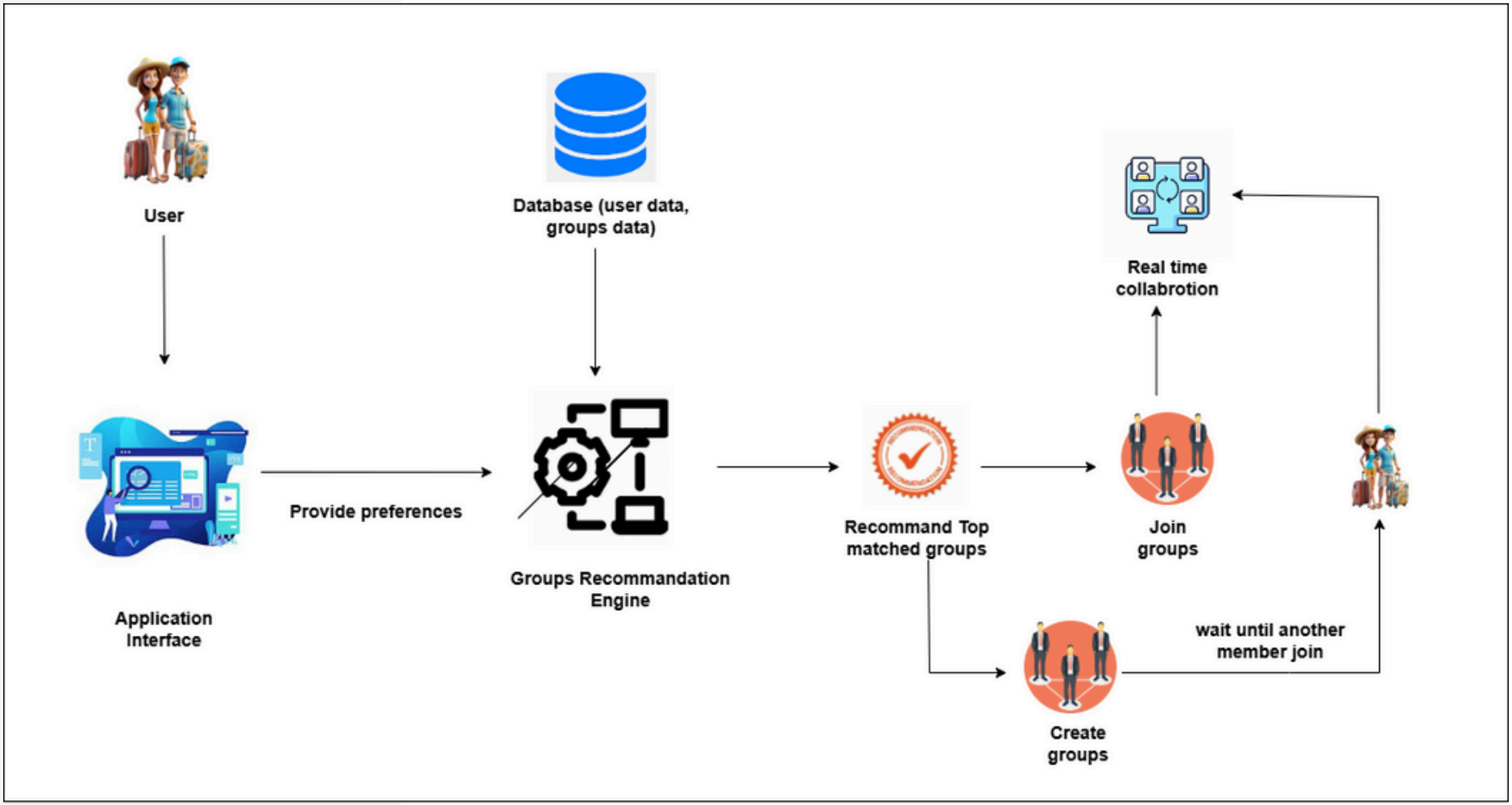
OBJECTIVE 03
To facilitate seamless communication between group members through a built-in chat system for planning and coordination.

OBJECTIVE 04
Ensure user-friendly interface and accessible for diverse traveler groups

METHODOLOGY

01. SYSTEM DIAGRAM
02. TOOLS AND TECHNOLOGIES
03. REQUIREMENTS
04. GITHUB COMMITS

SYSTEM DIAGRAM



TOOL AND TECHNOLOGIES

Programming Languages

Python

Project Management

Jira

Database

MongoDB

Frameworks &Libraries

lightgbm, pandas, numpy, fastapi, Scikit-learn,sentence-transformers, React

Other Tools

Git , Draw.io
Postman, Docker & Docker Compose

REQUIREMENTS

FUNCTIONAL

- Data Collection and Preprocessing
- Model Training
- User-Group Matching and Recommendations
- Real-Time Communication
- Trip Planning and Group Collaboration
- Platform Implementation

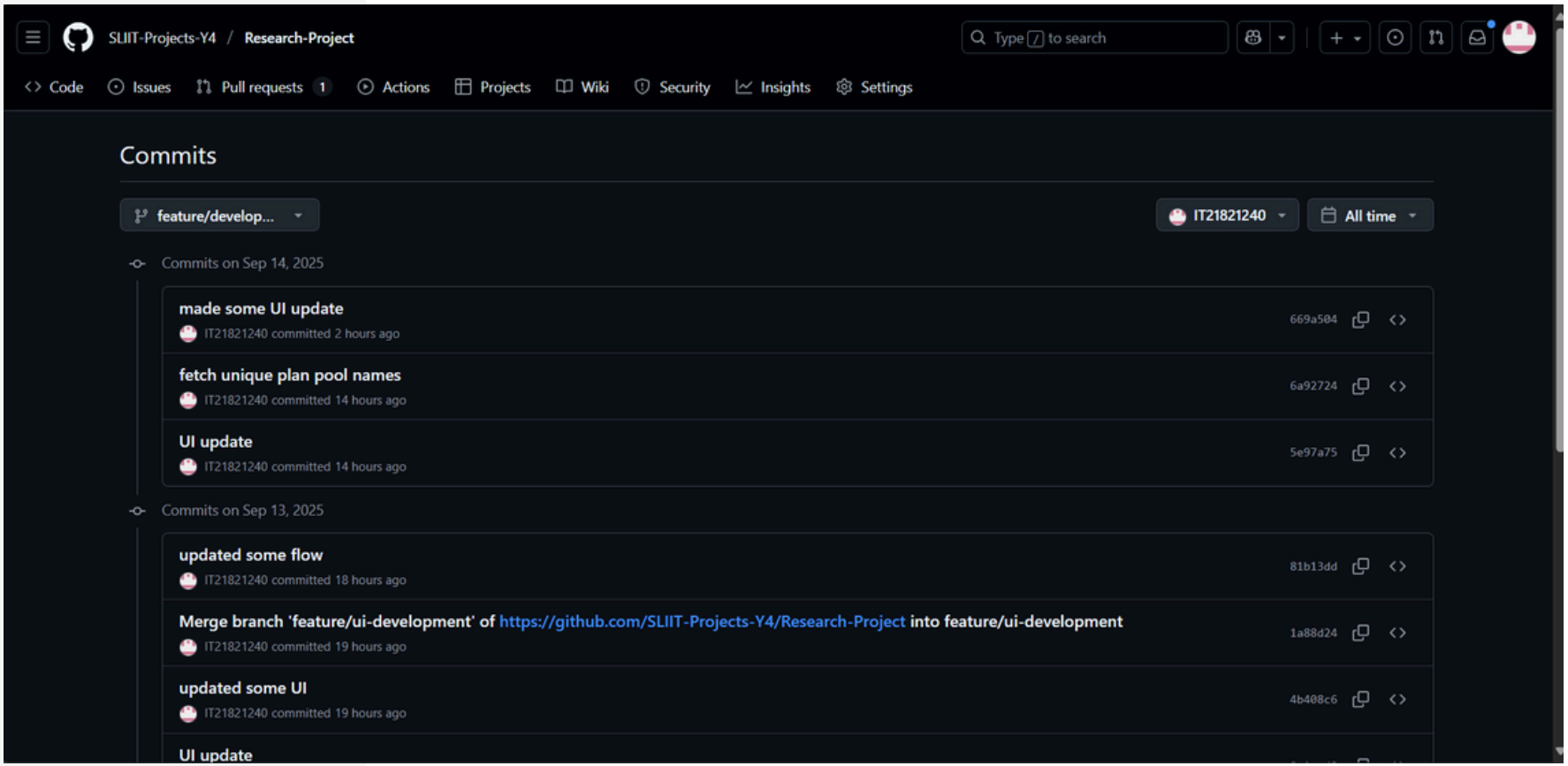
NON-FUNCTIONAL

- Performance
- Scalability
- Reliability
- Usability
- Security
- Maintainability

Standards and best practices

- Code quality
- Web security
- Modal quality
- Data Privacy

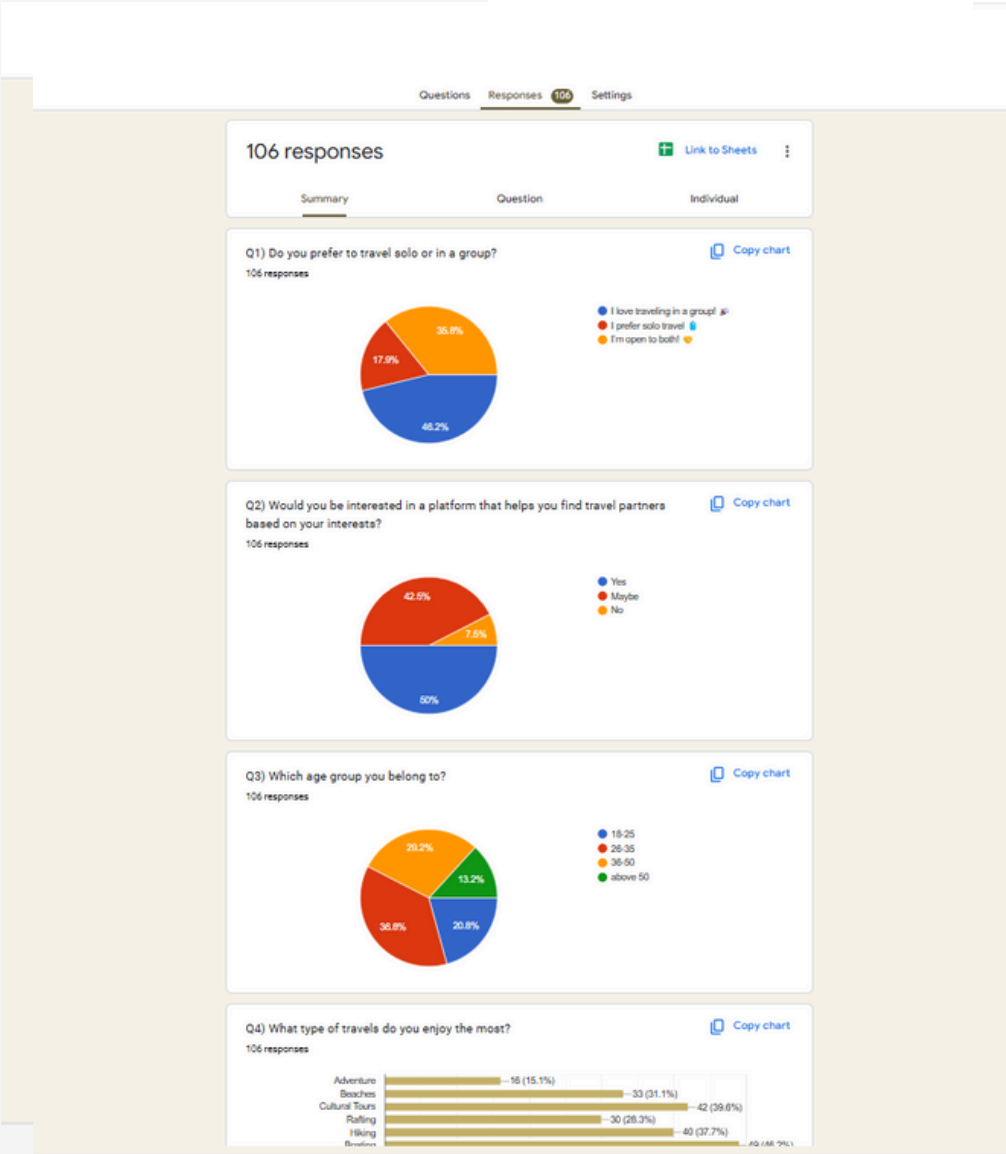
GITHUB COMMITS

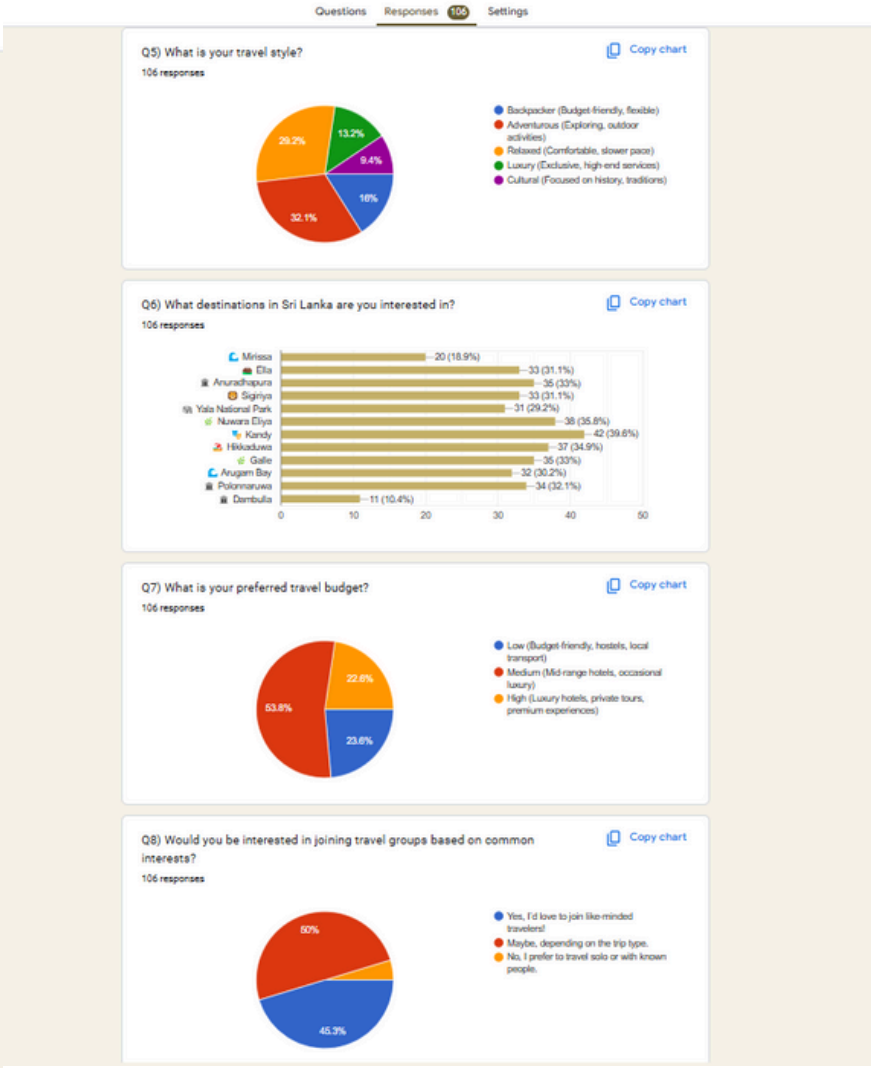


COMPLETION OF THE PROJECT

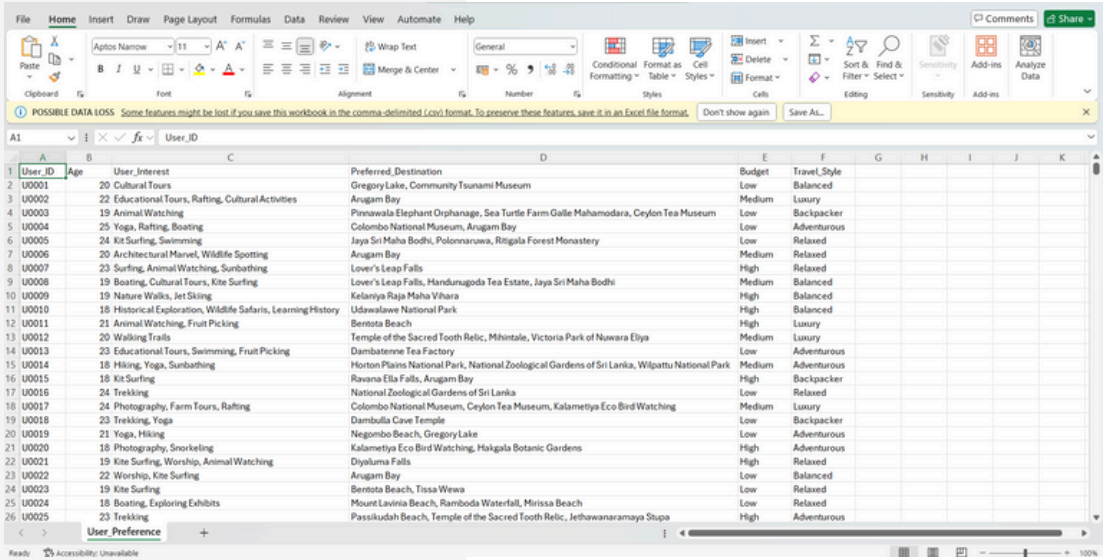
50%

DATA COLLECTION - SURVEY

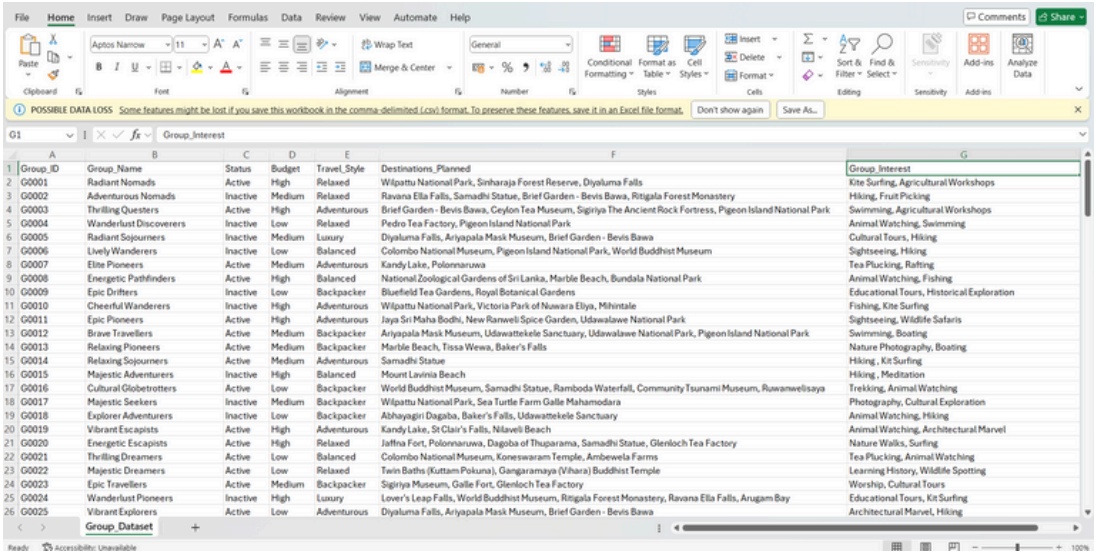




DATA SET CREATION



User_ID	Age	User_Interest	Preferred_Destination	Budget	Travel_Style
U0001	20	Cultural Tours	Gregory Lake, Community Tsunami Museum	Low	Balanced
U0002	22	Educational Tours, Rafting, Cultural Activities	Arugam Bay	Medium	Luxury
U0003	19	Animal Watching	Pinnawala Elephant Orphanage, Sea Turtle Farm Galle Mahamodara, Ceylon Tea Museum	Low	Backpacker
U0004	25	Yoga, Rafting, Boating	Colombo National Museum, Arugam Bay	Low	Adventurous
U0005	24	Kit Surfing, Swimming	Jaya Sri Maha Bodhi, Polonnaruwa, Ritigala Forest Monastery	Low	Relaxed
U0006	20	Architectural Marvel, Wildlife Spotting	Arugam Bay	Medium	Relaxed
U0007	23	Surfing, Animal Watching, Sunbathing	Lover's Leap Falls	High	Relaxed
U0008	19	Boating, Cultural Tours, Kite Surfing	Lover's Leap Falls, Handunugoda Tea Estate, Jaya Sri Maha Bodhi	Medium	Balanced
U0009	19	Nature Walks, Jet Skiing	Kelaniya Raja Maha Vihara	High	Balanced
U0010	18	Historical Exploration, Wildlife Safaris, Learning History	Udawalawe National Park	High	Balanced
U0011	21	Animal Watching, Fruit Picking	Bentota Beach	High	Luxury
U0012	20	Walking Trails	Temple of the Sacred Tooth Relic, Mihintale, Victoria Park of Nuwara Eliya	Medium	Luxury
U0013	23	Educational Tours, Swimming, Fruit Picking	Dambatenne Tea Factory	Low	Adventurous
U0014	18	Hiking, Yoga, Sunbathing	Horton Plains National Park, National Zoological Gardens of Sri Lanka, Wilpattu National Park	Medium	Adventurous
U0015	18	Kit Surfing	Ravana Ella Falls, Arugam Bay	High	Backpacker
U0016	24	Trekking	National Zoological Gardens of Sri Lanka	Low	Relaxed
U0017	24	Photography, Farm Tours, Rafting	Colombo National Museum, Ceylon Tea Museum, Kalametiya Eco Bird Watching	Medium	Luxury
U0018	23	Trekking, Yoga	Dambulla Cave Temple	Low	Backpacker
U0019	21	Yoga, Hiking	Negombo Beach, Gregory Lake	Low	Adventurous
U0020	18	Photography, Snorkeling	Kalametiya Eco Bird Watching, Hakgala Botanic Gardens	High	Adventurous
U0021	19	Kite Surfing, Worship, Animal Watching	Diyaluma Falls	High	Relaxed
U0022	22	Worship, Kite Surfing	Arugam Bay	Low	Balanced
U0023	19	Kite Surfing	Bentota Beach, Tissa Wewa	Low	Relaxed
U0024	18	Boating, Exploring Exhibits	Mount Lavinia Beach, Ramboda Waterfall, Mirissa Beach	Low	Relaxed
U0025	23	Trekking	Passikudah Beach, Temple of the Sacred Tooth Relic, Jethawanaramaya Stupa	High	Adventurous



Group_ID	Group_Name	Status	Budget	Travel_Style	Destinations_Planned	Group_Interest
G0001	Radiant Nomads	Active	High	Relaxed	Wilpattu National Park, Sinharaja Forest Reserve, Diyaluma Falls	Kite Surfing, Agricultural Workshops
G0002	Adventurous Nomads	Inactive	Medium	Relaxed	Ravana Ella Falls, Samadhi Statue, Brief Garden - Bevis Bawa, Ritigala Forest Monastery	Hiking, Fruit Picking
G0003	Thrilling Questers	Active	High	Adventurous	Brief Garden - Bevis Bawa, Ceylon Tea Museum, Sigiriya The Ancient Rock Fortress, Pigeon Island National Park	Swimming, Agricultural Workshops
G0004	Wanderlust Discoverers	Inactive	Low	Relaxed	Pedro Tea Factory, Pigeon Island National Park	Animal Watching, Swimming
G0005	Radiant Sojourners	Inactive	Medium	Luxury	Diyaluma Falls, Ariyapala Mask Museum, Brief Garden - Bevis Bawa	Cultural Tours, Hiking
G0006	Lively Wanderers	Inactive	Low	Balanced	Colombo National Museum, Pigeon Island National Park, World Buddhist Museum	Sightseeing, Hiking
G0007	Elite Pioneers	Active	Medium	Adventurous	Kandy Lake, Polonnaruwa	Tea Plucking, Rafting
G0008	Energetic Pathfinders	Active	High	Balanced	National Zoological Gardens of Sri Lanka, Marble Beach, Bundala National Park	Animal Watching, Fishing
G0009	Epic Drifters	Inactive	Low	Backpacker	Bluefield Tea Gardens, Royal Botanical Gardens	Educational Tours, Historical Exploration
G0010	Cheerful Wanderers	Inactive	High	Adventurous	Wilpattu National Park, Victoria Park of Nuwara Eliya, Mihintale	Fishing, Kite Surfing
G0011	Epic Pioneers	Active	High	Adventurous	Jaya Sri Maha Bodhi, New Ranweli Spice Garden, Udawalawe National Park	Sightseeing, Wildlife Safaris
G0012	Brave Travellers	Active	Medium	Backpacker	Ariyapala Mask Museum, Udawattekele Sanctuary, Udawalawe National Park, Pigeon Island National Park	Swimming, Boating
G0013	Relaxing Pioneers	Active	Medium	Backpacker	Marble Beach, Tissa Wewa, Baker's Falls	Nature Photography, Boating
G0014	Relaxing Sojourners	Active	Medium	Adventurous	Samadhi Statue	Hiking , Kit Surfing
G0015	Majestic Adventurers	Inactive	High	Balanced	Mount Lavinia Beach	Hiking , Meditation
G0016	Cultural Globetrotters	Active	Low	Backpacker	World Buddhist Museum, Samadhi Statue, Ramboda Waterfall, Community Tsunami Museum, Ruwanwelisaya	Trekking, Animal Watching
G0017	Majestic Seekers	Inactive	Medium	Backpacker	Wilpattu National Park, Sea Turtle Farm Galle Mahamodara	Photography, Cultural Exploration
G0018	Explorer Adventurers	Inactive	Low	Backpacker	Abhayagiri Dagaba, Baker's Falls, Udawattekele Sanctuary	Animal Watching, Hiking
G0019	Vibrant Escapists	Active	High	Adventurous	Kandy Lake, St Clair's Falls, Nilaveli Beach	Animal Watching, Architectural Marvel
G0020	Energetic Escapists	Active	High	Relaxed	Jaffna Fort, Polonnaruwa, Dagoba of Thuparama, Samadhi Statue, Glenloch Tea Factory	Nature Walks, Surfing
G0021	Thrilling Dreamers	Active	Low	Balanced	Colombo National Museum, Koneswaram Temple, Ambewela Farms	Tea Plucking, Animal Watching
G0022	Majestic Dreamers	Active	Low	Relaxed	Twin Baths (Kuttam Pokuna), Gangaramaya (Vihara) Buddhist Temple	Learning History, Wildlife Spotting
G0023	Epic Travellers	Active	Medium	Backpacker	Sigiriya Museum, Galle Fort, Glenloch Tea Factory	Worship, Cultural Tours
G0024	Wanderlust Pioneers	Inactive	High	Luxury	Lover's Leap Falls, World Buddhist Museum, Ritigala Forest Monastery, Ravana Ella Falls, Arugam Bay	Educational Tours, Kit Surfing
G0025	Vibrant Explorers	Active	Low	Adventurous	Diyaluma Falls, Ariyapala Mask Museum, Brief Garden - Bevis Bawa	Architectural Marvel, Hiking

DATA PRE PROCESSING

```python
# Load Dataset
user_df = pd.read_csv("/content/drive/My Drive/Colab Notebooks/User_Preference.csv")
group_df = pd.read_csv("/content/drive/My Drive/Colab Notebooks/Group_Dataset.csv")
interaction_df = pd.read_csv("/content/drive/My Drive/Colab Notebooks/User-Group_Interactions_Dataset.csv")
```

Python

```python
# Step 1: Preprocess multi-label fields
def split_multi(x):
    if pd.isna(x):
        return []
    return [i.strip() for i in x.split(',')] if isinstance(x, str) else []

user_df['User_Interest'] = user_df['User_Interest'].apply(split_multi)
user_df['Preferred_Destination'] = user_df['Preferred_Destination'].apply(split_multi)
group_df['Group_Interest'] = group_df['Group_Interest'].apply(split_multi)
group_df['Destinations_Planned'] = group_df['Destinations_Planned'].apply(split_multi)
```

Python

```python
#Step 2: Merge DataFrames into a single DataFrame
df = interaction_df.merge(user_df, on='User_ID').merge(group_df, on='Group_ID')
```

Python

```python
#Step 3: Ensure no missing values
f['Budget_x'] = df['Budget_x'].fillna('Unknown')
df['Travel_Style_x'] = df['Travel_Style_x'].fillna('Unknown')
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Current_Members'] = df['Current_Members'].fillna(df['Current_Members'].median())
```

Python

MODEL SELECTION

Selected Model LightGBM with LambdaRank

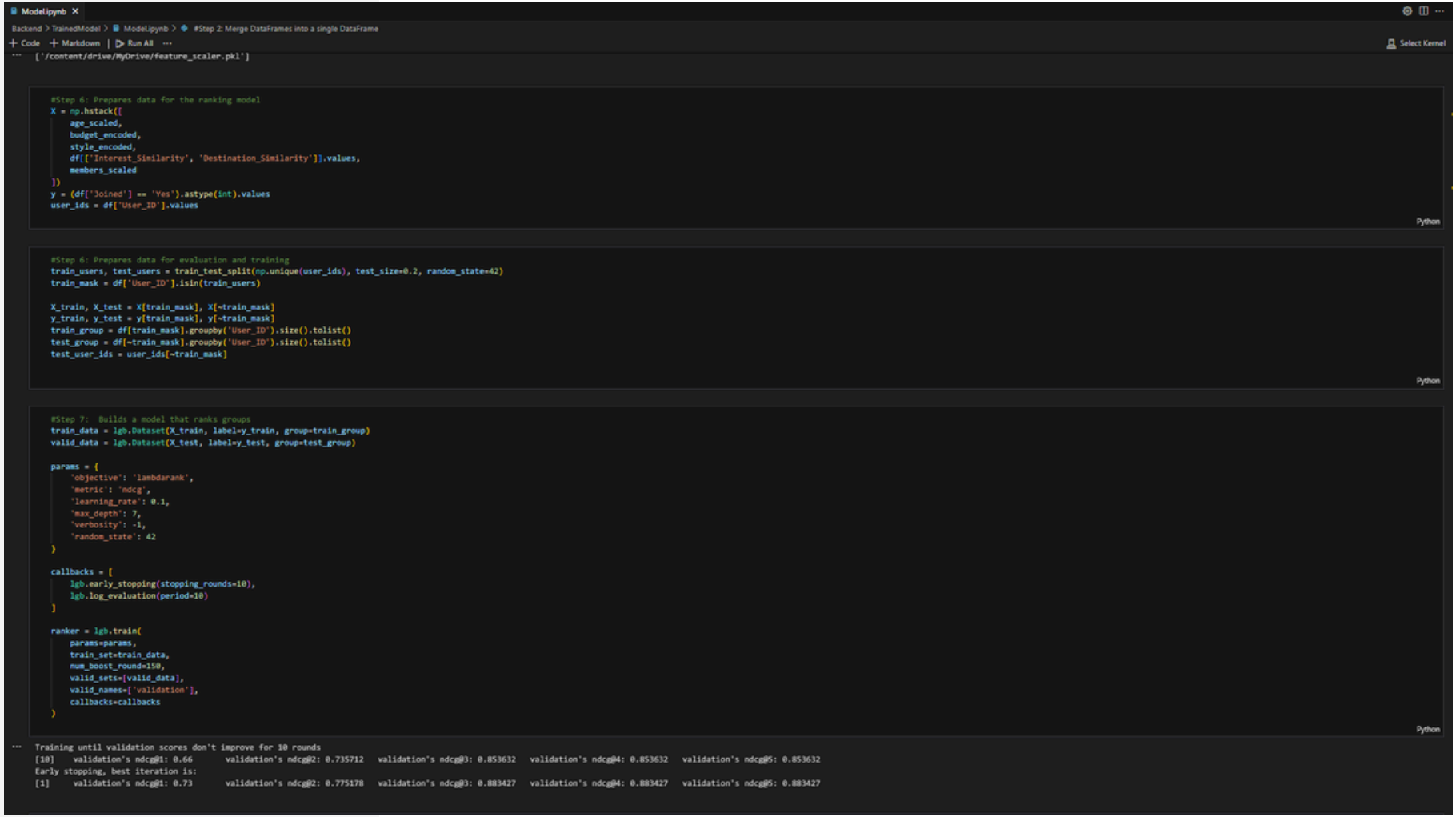
Explored Random Forest Classifier model to predict whether a user would join a group
It had two major limitations:

- It did not rank groups by relevance.
- It could only say “join or not”, not which group is most suitable

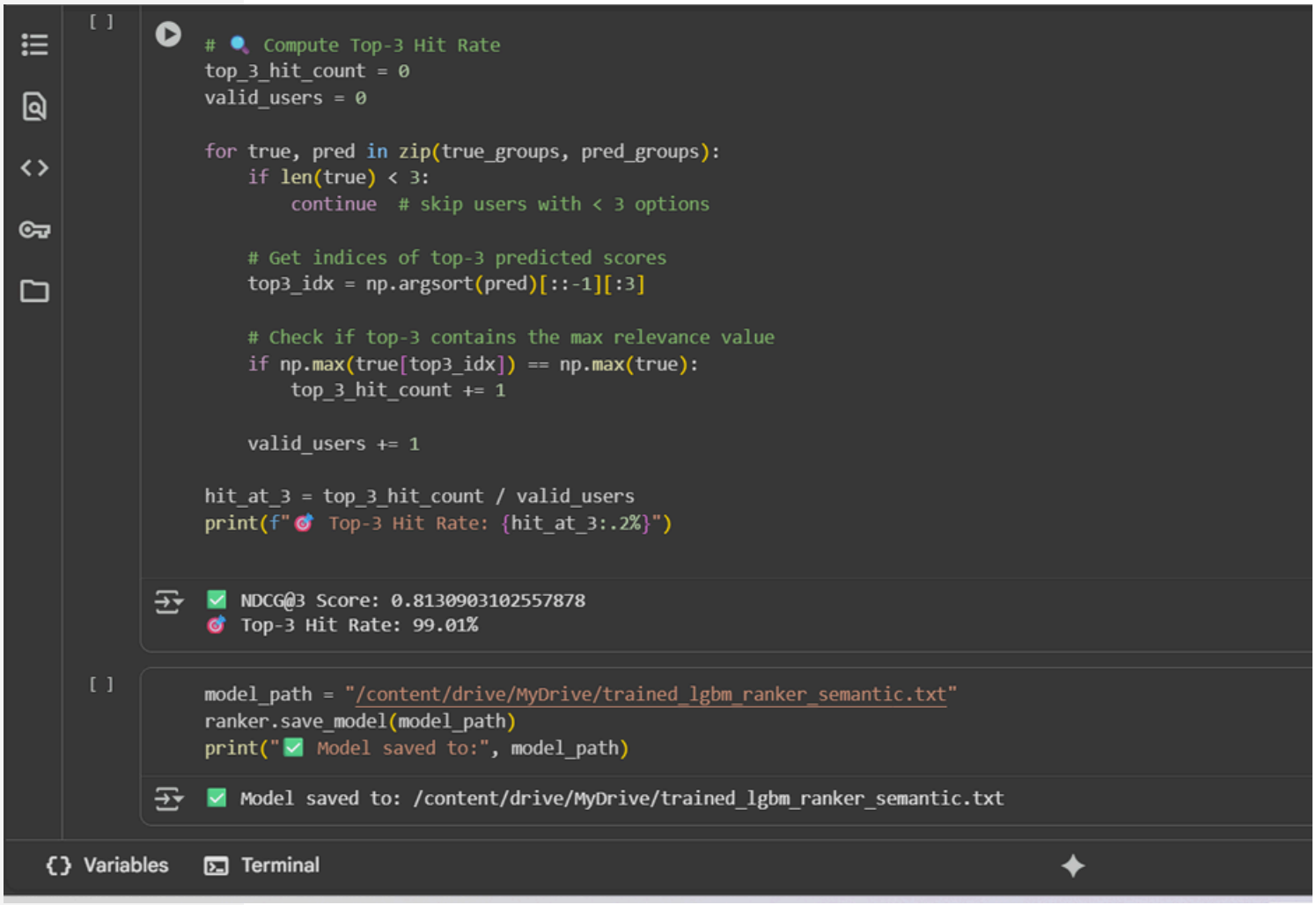
Reasons:

- Optimized for Ranking
- Supports Complex Features
- High Performance
- Better Accuracy
- Hybrid Capability

MODEL TRAINING



MODEL EVALUATION



```
# 🔍 Compute Top-3 Hit Rate
top_3_hit_count = 0
valid_users = 0

for true, pred in zip(true_groups, pred_groups):
    if len(true) < 3:
        continue  # skip users with < 3 options

    # Get indices of top-3 predicted scores
    top3_idx = np.argsort(pred)[::-1][:3]

    # Check if top-3 contains the max relevance value
    if np.max(true[top3_idx]) == np.max(true):
        top_3_hit_count += 1

    valid_users += 1

hit_at_3 = top_3_hit_count / valid_users
print(f"🎯 Top-3 Hit Rate: {hit_at_3:.2%}")
```

```
✅ NDCG@3 Score: 0.8130903102557878
🎯 Top-3 Hit Rate: 99.01%
```

```
model_path = "/content/drive/MyDrive/trained_lgbm_ranker_semantic.txt"
ranker.save_model(model_path)
print("✅ Model saved to:", model_path)
```

```
✅ Model saved to: /content/drive/MyDrive/trained_lgbm_ranker_semantic.txt
```

Variables Terminal

WORKING COMPONENT

```python
import joblib
import numpy as np
import pandas as pd
from sentence_transformers import SentenceTransformer
import lightgbm as lgb

# Load saved components
model = lgb.Booster(model_file="/content/drive/MyDrive/trained_lgbm_ranker_semantic.txt")
budget_encoder = joblib.load("/content/drive/MyDrive/budget_encoder.pkl")
style_encoder = joblib.load("/content/drive/MyDrive/style_encoder.pkl")
scaler = joblib.load("/content/drive/MyDrive/feature_scaler.pkl")
sbert = SentenceTransformer('all-MiniLM-L6-v2')

# Load group dataset
group_df = pd.read_csv("/content/drive/My Drive/Colab Notebooks/Group_Dataset.csv")
group_df['Group_Interest'] = group_df['Group_Interest'].fillna('').apply(lambda x: [i.strip() for i in x.split(',')])
group_df['Destinations_Planned'] = group_df['Destinations_Planned'].fillna('').apply(lambda x: [i.strip() for i in x.split(',')])
def recommend_groups_for_user(age, budget, interests, preferred_destinations=None, travel_style="Unknown"):
    # Clean up destinations list
    preferred_destinations = preferred_destinations if preferred_destinations else []

    # Compute user embeddings
    user_interest_text = ", ".join(interests)
    user_interest_emb = sbert.encode([user_interest_text])[0]

    if preferred_destinations:
        user_dest_text = ", ".join(preferred_destinations)
        user_dest_emb = sbert.encode([user_dest_text])[0]
    else:
        user_dest_emb = np.zeros(384)  # Fallback for destination

    # Group embeddings
    group_interest_embs = sbert.encode(group_df['Group_Interest'].apply(lambda x: ', '.join(x)).tolist())
    group_dest_embs = sbert.encode(group_df['Destinations_Planned'].apply(lambda x: ', '.join(x)).tolist())
```

[35]

Spaces: 4 CRLF {} Cell 7 of 17

```python
recommend_groups_for_user(
    age=23,
    budget="High",
    interests=["Hiking", "Nature"],
    travel_style="Backpacking"
)
```

[47] Python

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but OneHotEncoder was fitte
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but OneHotEncoder was fitte
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted
  warnings.warn(
```

	Group_ID	Group_Interest	Destinations_Planned	Current_Members	Score
19	G0020	[Nature Walks, Surfing]	[Jaffna Fort, Polonnaruwa, Dagoba of Thuparama...	8	0.151932
118	G0119	[Sightseeing, Wildlife]	[Mount Lavinia Beach, National Zoological Gard...	9	0.151932
74	G0075	[Meditation, Hiking]	[Negombo Beach, Abhayagiri Dagaba, Mount Lavin...	6	0.151932

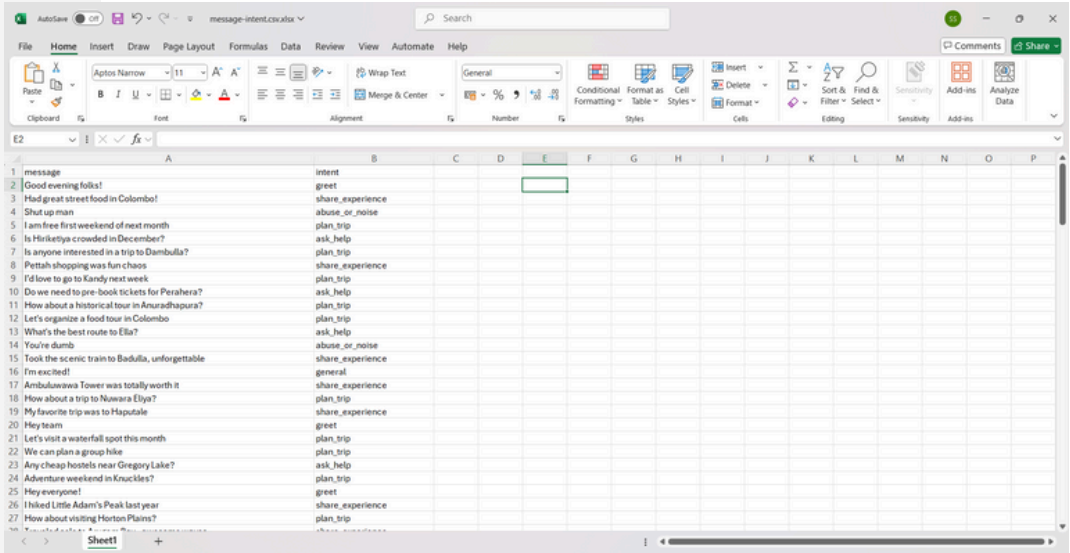
COMPLETION OF THE PROJECT

90%

KEY FUNCTIONALITIES

- **Personalized Group Recommendations**: Uses a trained LightGBM ranker with encoded user data to suggest the best-fit travel groups.
- **Machine Learning–Powered Matching**: Employs feature scaling and encoders along with a semantic ranker to make accurate group predictions.
- **Dynamic Group Creation**: If no suitable group exists, a user can create a new group and wait for other members to join.
- **AI Assistant (TripBot) with Multi-Intent Classification**: TripBot leverages an SBERT-based intent classifier and extractors to understand queries and provides contextual travel suggestions inside group chats.
- **Interactive Group Chat Features**: Enriched chat experience with polls, reactions, file sharing, translations, and TripBot integration—making group planning collaborative and engaging.

INTENT CLASSIFIER MODEL



	message	intent
2	Good evening folks!	greet
3	Had great street food in Colombo!	share_experience
4	Shut up man	abuse_or_noise
5	I am free first weekend of next month	plan_trip
6	Is Hiriketiya crowded in December?	ask_help
7	Is anyone interested in a trip to Dambulla?	plan_trip
8	Pettah shopping was fun chaos	share_experience
9	I'd love to go to Kandy next week	plan_trip
10	Do we need to pre-book tickets for Perahera?	ask_help
11	How about a historical tour in Anuradhapura?	plan_trip
12	Let's organize a food tour in Colombo	plan_trip
13	What's the best route to Ella?	ask_help
14	You're dumb	abuse_or_noise
15	Took the scenic train to Badulla, unforgettable	share_experience
16	I'm excited!	general
17	Ambuluwawa Tower was totally worth it	share_experience
18	How about a trip to Nuwara Eliya?	plan_trip
19	My favorite trip was to Haputale	share_experience
20	Hey team	greet
21	Let's visit a waterfall spot this month	plan_trip
22	We can plan a group hike	plan_trip
23	Any cheap hostels near Gregory Lake?	ask_help
24	Adventure weekend in Knuckles?	plan_trip
25	Hey everyone!	greet
26	I hiked Little Adam's Peak last year	share_experience
27	How about visiting Horton Plains?	plan_trip

```python
import pandas as pd
import joblib

# Load dataset
df = pd.read_excel("/content/drive/My Drive/dataset/message-intent.xlsx")
X = df["message"].astype(str).tolist()
y = df["intent"].astype(str).tolist()

# SBERT embeddings
model = SentenceTransformer("all-MiniLM-L6-v2")
X_vectors = model.encode(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_vectors, y, test_size=0.2, random_state=42)

# Train classifier
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train, y_train)

# Evaluate
preds = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, preds))

# Save model
joblib.dump(clf, "/content/drive/My Drive/dataset/sbert_intent_classifier.pkl")
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: FutureWarning: `encoder_attention_mask` is deprecated and will be removed in version 4.55.0 for `BertSdpa
  return forward_call(*args, **kwargs)
Accuracy: 0.9069767441860465
['/content/drive/My Drive/dataset/sbert_intent_classifier.pkl']
```

RECOMMENDER.PY

recommender.py 2, U

collaborative > ml > recommender.py > load_groups_from_db

```python
from sentence_transformers import SentenceTransformer
import lightgbm as lgb
from database.database import db
from typing import List, Optional

# Load models
model = lgb.Booster(model_file="TrainedModel/trained_lgbm_ranker_semantic.txt")
budget_encoder = joblib.load("TrainedModel/budget_encoder.pkl")
style_encoder = joblib.load("TrainedModel/style_encoder.pkl")
scaler = joblib.load("TrainedModel/feature_scaler.pkl")
sbert = SentenceTransformer('all-MiniLM-L6-v2')

# Load groups from MongoDB
async def load_groups_from_db():
    groups = await db.groups.find().to_list(length=None)

    if not groups:
        print("❌ No groups found in database.")
        return pd.DataFrame()  # safe empty return

    df = pd.DataFrame(groups)
    print("✅ Loaded DataFrame columns:", df.columns.tolist())

    if 'Group_Interest' in df.columns:
        df['Group_Interest'] = df['Group_Interest'].fillna('').apply(
            lambda x: [i.strip() for i in x] if isinstance(x, list) else []
        )
    else:
        print("⚠️ 'Group_Interest' missing from DataFrame. Adding empty lists.")
        df['Group_Interest'] = [[] for _ in range(len(df))]

    if 'Destinations_Planned' in df.columns:
        df['Destinations_Planned'] = df['Destinations_Planned'].fillna('').apply(
            lambda x: [i.strip() for i in x] if isinstance(x, list) else []
        )
    else:
        print("⚠️ 'Destinations_Planned' missing. Adding empty lists.")
        df['Destinations_Planned'] = [[] for _ in range(len(df))]

```

recommender.py 2, U

collaborative > ml > recommender.py > load_groups_from_db

```python
async def recommend_groups_for_user(age, budget, interests, preferred_destinations=None, travel_style="Unknown", exclude_id
    user_interest_text = ", ".join(interests)
    user_interest_emb = sbert.encode([user_interest_text])[0]

    if preferred_destinations:
        user_dest_text = ", ".join(preferred_destinations)
        user_dest_emb = sbert.encode([user_dest_text])[0]
    else:
        user_dest_emb = np.zeros(384)

    group_interest_embs = sbert.encode(group_df['Group_Interest'].apply(lambda x: ', '.join(x)).tolist())
    group_dest_embs = sbert.encode(group_df['Destinations_Planned'].apply(lambda x: ', '.join(x)).tolist())

    interest_sims = np.dot(group_interest_embs, user_interest_emb) / (
        np.linalg.norm(group_interest_embs, axis=1) * np.linalg.norm(user_interest_emb)
    )

    dest_sims = np.dot(group_dest_embs, user_dest_emb) / (
        np.linalg.norm(group_dest_embs, axis=1) * np.linalg.norm(user_dest_emb)
    ) if preferred_destinations else np.zeros(len(group_df))

    budget_encoded = budget_encoder.transform([[budget]])
    style_encoded = style_encoder.transform([[travel_style]])
    age_scaled = np.repeat(scaler.transform([[age]])[:, 0], len(group_df))
    members_scaled = scaler.transform(group_df[['Current_Members']])[:, 0]

    X_new = np.hstack([
        age_scaled.reshape(-1, 1),
        np.repeat(budget_encoded, len(group_df), axis=0),
        np.repeat(style_encoded, len(group_df), axis=0),
        interest_sims.reshape(-1, 1),
        dest_sims.reshape(-1, 1),
        members_scaled.reshape(-1, 1)
    ])

    scores = model.predict(X_new)
    group_df['Score'] = scores
    top_6 = group_df.sort_values(by='Score', ascending=False).head(6)

    return top_6[['Group_ID', 'Group_Name', 'Budget', 'Group_Interest', 'Destinations_Planned','Travel_Style', 'Current_Mem
```

GROUP CHAT IMPLEMENTATION

chat_ws.py 3, U

collaborative > routes > chat_ws.py > ...

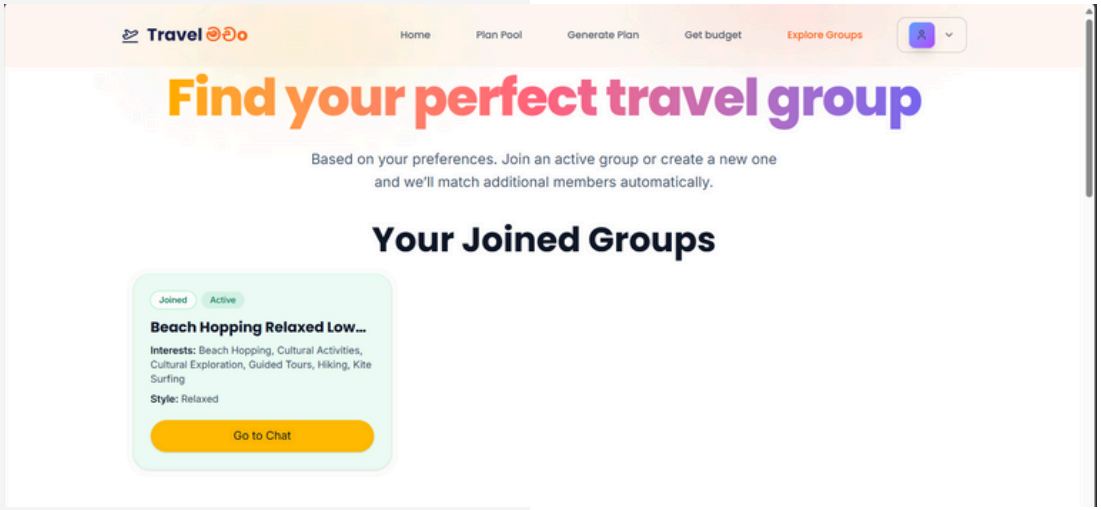
```python
UPLOAD_DIR = "uploads"
os.makedirs(UPLOAD_DIR, exist_ok=True)

def predict_intent(message: str) -> str:
    vector = sbert_model.encode([message])
    return clf.predict(vector)[0]

router = APIRouter()
active_connections: Dict[str, List[WebSocket]] = {}

# Group context reference
group_context: Dict[str, Dict] = {}

# Temporarily buffer experience messages per group/user
experience_buffer: Dict[str, Dict[str, Dict]] = {}

# Connect user to group
async def connect_user(group_id: str, websocket: WebSocket):
    await websocket.accept()
    if group_id not in active_connections:
        active_connections[group_id] = []
    active_connections[group_id].append(websocket)

# Disconnect user
def disconnect_user(group_id: str, websocket: WebSocket):
    if group_id in active_connections:
        active_connections[group_id].remove(websocket)

# Broadcast message to all users in a group
async def broadcast_message(group_id: str, message: dict):
    for connection in active_connections.get(group_id, []):
        await connection.send_json(message)
```

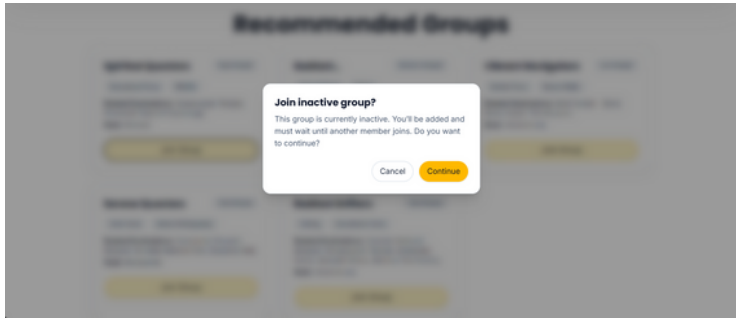
chat_ws.py 3, U

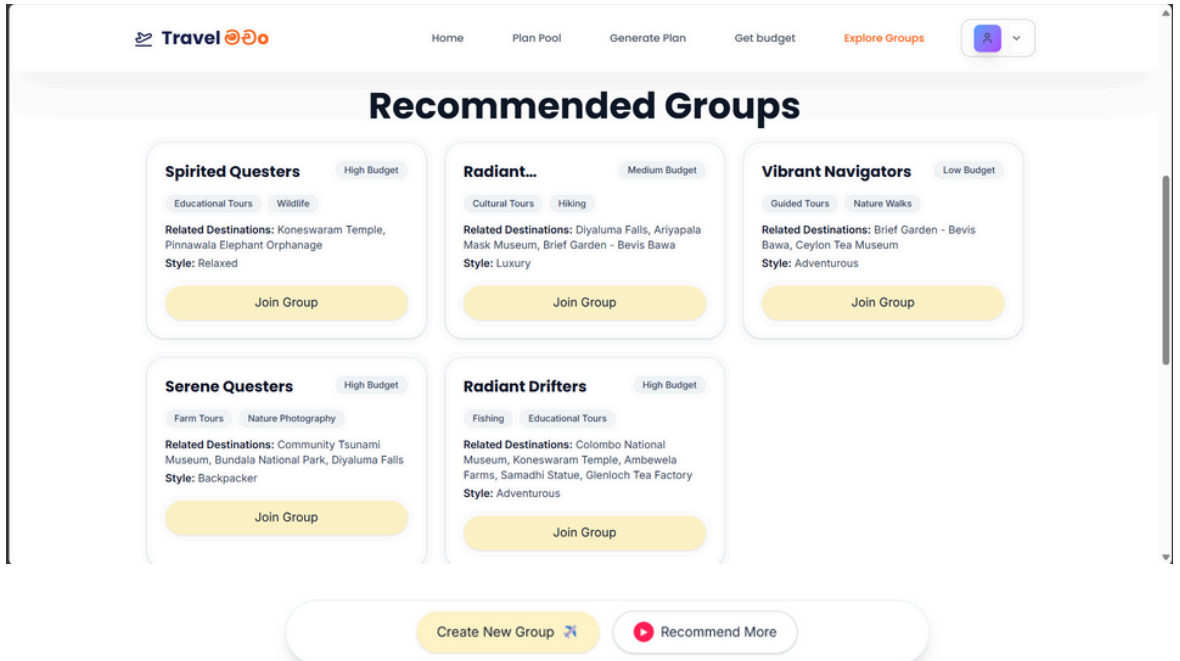
collaborative > routes > chat_ws.py > ...

```python
def clean_mongo(obj):
            for k, v in obj.items()
        }
    elif isinstance(obj, list):
        return [clean_mongo(i) for i in obj]
    return obj


async def flush_experience_after_silence(group_id, user_id, username, bot, timeout=35):
    await asyncio.sleep(timeout)
    group_buf = experience_buffer.get(group_id, {})
    user_buf = group_buf.get(user_id)
    if not user_buf or user_buf.get("intent") != "share_experience":
        return  # Don't flush if it's not meant to be experience

    if user_buf and user_buf["messages"]:
        combined = " ".join(user_buf["messages"])

        # Extract all destinations and activities
        destinations = extract_destination(combined)
        activities = extract_activity(combined)

        # Save to context
        await bot.record_experience(username, combined, destinations, activities)

        # Build rich response
        response = f"🙏 Thanks {username}, your tips have been saved!"
        if destinations:
            response += f" 🗺️ Related to {', '.join(f'**{d}**' for d in destinations)}."
        if activities:
            response += f" ✨ Tagged as {', '.join(f'**{a}**' for a in activities)}."

        await send_bot_message(group_id, response)

        # Reset buffer
        experience_buffer[group_id][user_id] = {
            "messages": [],
            "last_message_time": datetime.utcnow(),
        }
```

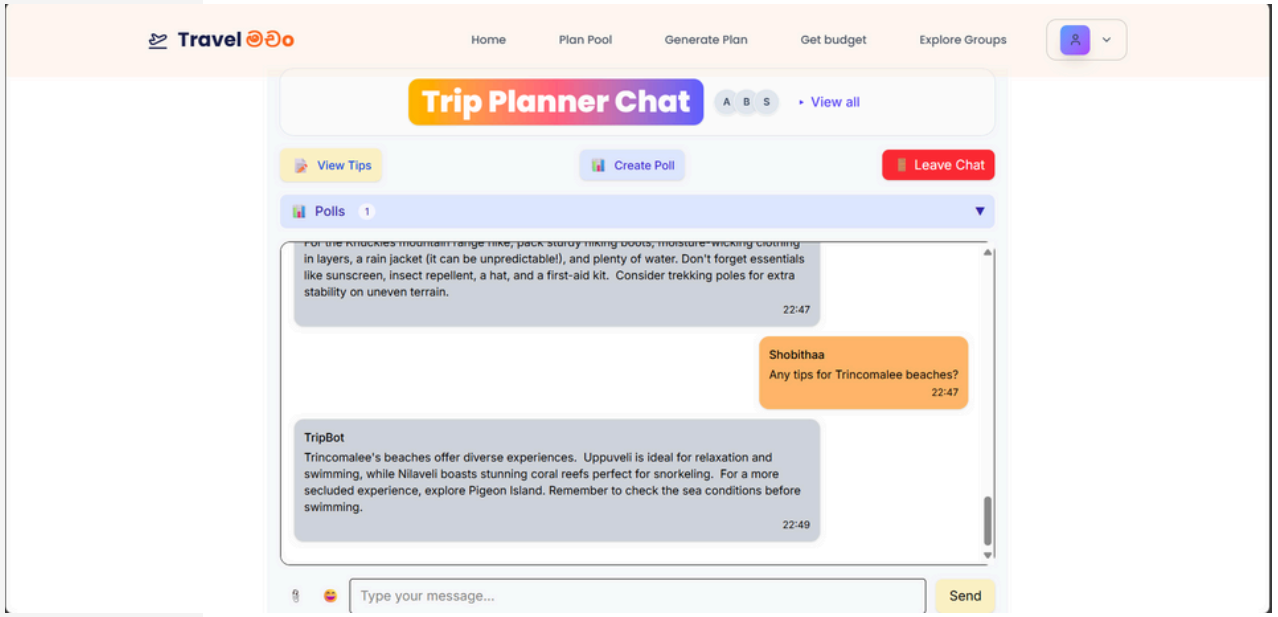
WORKING PRODUCT

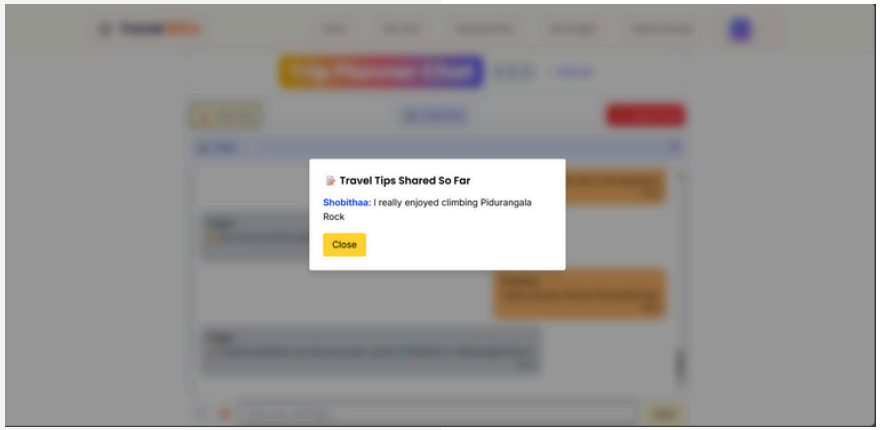


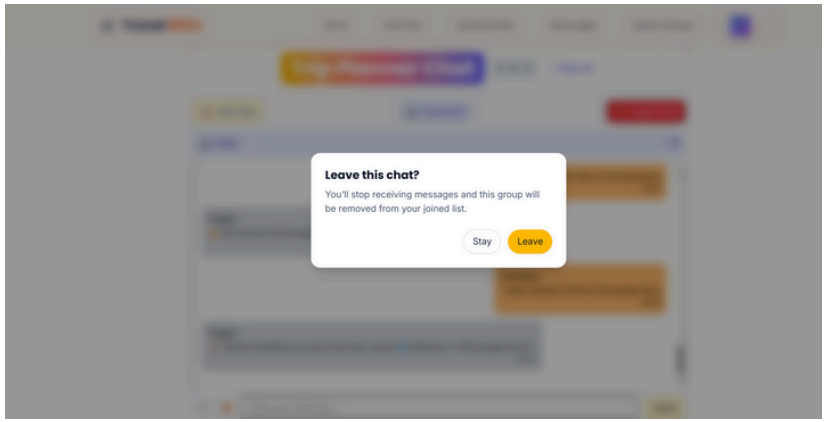




WORKING PRODUCT







REFERENCES

[1] J. Zhang, B. Chen, and M. Zhang, "A group travel recommender system based on group approximate constraint satisfaction," in Proc. IEEE Int. Conf. on Artificial Intelligence and Computer Applications (ICAICA), 2021.

[2] Y. Liu and X. Wang, "A systematic review of group recommender systems techniques," IEEE Access, vol. 9, pp. 45327–45340, 2021.

[3] S. Li, J. Zhao, and H. Wang, "An intelligent personalized recommendation for travel group planning based on reviews," in Proc. IEEE Int. Conf. on Information Technology Applications (ITA), 2022.

[4] A. N. Mirza and S. Rehman, "User profiling in travel recommender system using hybridization and collaborative method," in Proc. IEEE Int. Conf. on Communication, Computing and Signal Processing (ICCCSP), 2021.

[5] M. Z. Khan, M. R. Hossain, and A. I. Haque, "Collaborative filtering algorithm in the design of intelligent tourism service robots," in Proc. IEEE Int. Conf. on Computer and Information Technology (ICCIT), 2020.

[6] Y. Sun, J. Xu, and W. Wang, "Travel recommendation by mining people attributes and travel group types from community-contributed photos," IEEE Access, vol. 8, pp. 56110–56120, 2020.

[7] T. Lee and K. H. Lee, "Personalized tourism route recommendation system based on dynamic clustering of user groups," IEEE Trans. Ind. Inf., vol. 17, no. 4, pp. 2812– 2821, 2021.

IT21831768 - SENEVIRATHNE.S.D.C.D

SPECIALIZING - SOFTWARE ENGINEERING

AGENDA

01. BACKGROUND
02. RESEARCH QUESTIONS
03. OBJECTIVES
04. SYSTEM DIAGRAM

BACKGROUND

RECOMMENDATION SYSTEM

- Travelers often struggle to choose where to go, we use user demographics and preferences to recommend locations.
- Traditional systems offer little personalization.
- Combining multiple recommendation approaches improves relevance and accuracy.

PLANNER SYSTEM

- Efficient route planning is essential for tourism in Sri Lanka.
- Province-based optimization reduces travel time and wasted effort.
- Finally Optimize based on the distance of locations.

ITINERARY SWITCHER

- Replace the generated route with smarter alternative itineraries.
- Swap stops with locations that offer similar activities.

RESEARCH Problems?

01 How can a hybrid recommender (CBF + CF + ML) that uses user demographics and preferences outperform single-method baselines for top-N location suggestions in Sri Lanka?

02 To what extent does province-based route optimization reduce total travel time and backtracking compared with distance-only or greedy planners?

03 Can the itinerary switcher replace stops with similar-activity locations while keeping routes feasible and efficient, and how do users rate these swaps ?

OBJECTIVES

- Generate Personalized Recommendations
- Efficient Plan Optimization
- Suggest Smart Alternatives
- Enhance User Experience

METHODOLOGY

01. SYSTEM DIAGRAM
02. TOOL & TECHNOLOGIES
03. REQUIREMENTS

SYSTEM DIAGRAM

User

Selecting User Preferences

CBF + CF + ML
(Hybrid Filtering)

Recommend Locations

Plan Pool

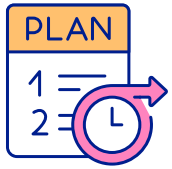
Start Location

End Location

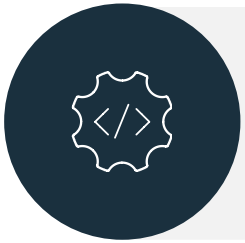
Generated Plan

Itinerary Planning

Optimize Plan



TOOL AND TECHNOLOGIES



Programming Languages

Python ,
JavaScript

Project Management

JIra

Database

MongoDB

Frameworks &Libraries

Pandas, Spacy, React,
Scikit-learn, mantine

Other Tools

Git , Draw.io Postman,
Docker

REQUIREMENTS

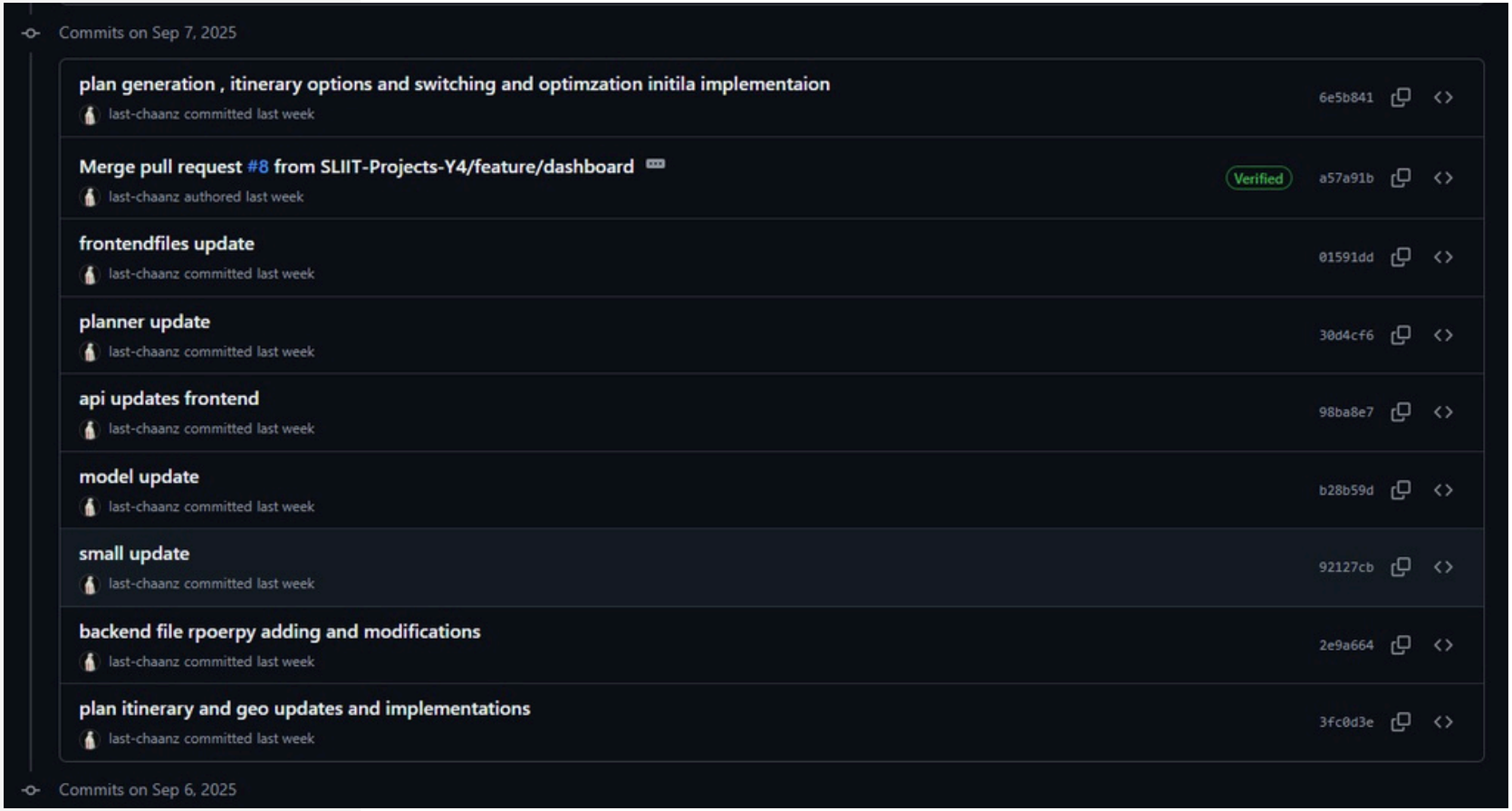
FUNCTIONAL

- User Demographic data collection and Preference collection.
- Real-time recommendation generation based on user inputs.
- Provide Itineraries based on rating, neaby and smart alternatives.
- Ability to optimize the plan after the itinerary changes

NON-FUNCTIONAL

- Performance - Recommending locations under 10 seconds for the user.
- Portability - Contrainerized docker environment.
- Secuirty - Protect User demographic and sensitive data

GITHUB COMMITS



COMPLETION OF THE PROJECT

50%

DATA COLLECTION

User_ID	User_Country	User_Age_Group	Gender	Travel_Companion	Location_Type	Preferred_Activities
User 1	Australia	36-50	Male	Family	['Nature & Wildlife Areas', 'Gardens', 'Beaches']	['Wildlife Safaris', 'Fishing', 'Factory Tours', 'Hiking']
User 2	Australia	26-35	Male	Friends	['Nature & Wildlife Areas', 'National Parks', 'Beaches']	['Cultural Tours', 'Boat Tours', 'Sightseeing', 'Snorkeling']
User 3	Australia	18-25	Male	Friends	['National Parks', 'Beaches']	['Boat Tours', 'Jet Skiing', 'Hiking', 'Rafting']
User 4	Portugal	18-25	Male	Solo	['Museums', 'Farms', 'Beaches']	['Yoga', 'Photography', 'Jet Skiing', 'Snorkeling']
User 5	Italy	18-25	Male	Family	['Religious Sites', 'Historic Sites', 'Beaches']	['Rafting', 'Wildlife Spotting', 'Hiking', 'Jet Skiing']
User 6	United Arab Emirates	18-25	Female	Solo	['Historic Sites', 'Beaches']	['Beach Hopping', 'Nature Walks', 'Photography', 'Hiking']
User 7	Indonesia	36-50	Male	Friends	['Historic Sites', 'Beaches']	['Historical Exploration', 'Hiking', 'Walking Trails', 'Bird Watching']
User 8	Canada	18-25	Female	Solo	['Historic Sites', 'Beaches', 'Bodies of Water']	['Nature Walks', 'Photography', 'Beach Hopping', 'Snorkeling']
User 9	New Zealand	36-50	Male	Family	['Beaches', 'Historic Sites', 'Farms']	['Agricultural Workshops', 'Historical Exploration', 'Wildlife Safaris', 'Hiking']
User 10	United Kingdom	51+	Female	Couple	['Museums', 'Historic Sites', 'Beaches']	['Cultural Activities', 'Meditation', 'Relaxation']
User 11	United Kingdom	18-25	Female	Friends	['Waterfalls', 'Beaches', 'Bodies of Water']	['Boat Tours', 'Beach Hopping', 'Snorkeling', 'Surfing']
User 12	Japan	36-50	Male	Family	['Gardens', 'Beaches']	['Fishing', 'Historical Exploration', 'Wildlife Safaris', 'Hiking']
User 13	Australia	26-35	Female	Family	['Historic Sites', 'Beaches']	['Photography', 'Yoga', 'Nature Walks']
User 14	United Kingdom	26-35	Male	Family	['Religious Sites', 'Beaches']	['Hiking', 'Sightseeing', 'Cultural Tours']
User 15	Germany	18-25	Female	Couple	['Beaches', 'Zoological Gardens']	['Relaxation', 'Yoga', 'Beach Hopping', 'Sunbathing']
User 16	Ukraine	18-25	Female	Solo	['Nature & Wildlife Areas', 'Beaches']	['Photography', 'Hiking', 'Nature Walks', 'Yoga']
User 17	United Kingdom	26-35	Female	Couple	['National Parks', 'Beaches']	['Sunbathing', 'Meditation', 'Yoga']
User 18	India	26-35	Male	Family	['Beaches', 'Nature & Wildlife Areas', 'Religious Sites']	['Wildlife Spotting', 'Factory Tours', 'Cultural Tours']
User 19	United Kingdom	51+	Male	Friends	['Waterfalls', 'Historic Sites', 'Beaches', 'Museums', 'Gardens	['Meditation', 'Wildlife Safaris', 'Worship']
User 20	Viet Nam	51+	Female	Friends	['Religious Sites', 'National Parks', 'Beaches']	['Cultural Exploration', 'Gardens', 'Worship', 'Nature Walks']

Location_ID	Location_Name	Located_City	Located_Province	Location_Type	Latitude	Longitude	Avg_Rating	Review_Count	Activities
LOC_1	Arugam Bay	Arugam Bay	Eastern Province	Beaches	6.8379771	81.8251687	4.06	187	['Swimming', 'Surfing', 'Sunbathing', 'Yoga', 'Boat
LOC_2	Bentota Beach	Bentota	Southern Province	Beaches	6.4234707	79.9953199	4.59	233	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling',
LOC_3	Hikkaduwa Beach	Hikkaduwa	Southern Province	Beaches	6.1377266	80.0990596	4.02	213	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling',
LOC_4	Jungle Beach	Unawatuna	Southern Province	Beaches	6.0186943	80.2394104	3.54	183	['Swimming', 'Sunbathing', 'Snorkeling', 'Hiking']
LOC_5	Kalutara Beach	Kalutara	Western Province	Beaches	6.6098466	79.9474179	3.67	218	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling',
LOC_6	Marble Beach	Trincomalee	Eastern Province	Beaches	7.7853051	81.4278984	3.97	145	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling']
LOC_7	Mirissa Beach	Mirissa	Southern Province	Beaches	5.9447028	80.4591608	4.27	217	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling',
LOC_8	Mount Lavinia Beach	Colombo	Western Province	Beaches	6.842362	79.8624086	4.13	180	['Swimming', 'Sunbathing', 'Kite Surfing']
LOC_9	Negombo Beach	Negombo	Western Province	Beaches	7.2055208	79.8512562	3.12	209	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling',
LOC_10	Nilaveli Beach	Nilaveli	Eastern Province	Beaches	8.7003072	81.19205	4.33	165	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling']
LOC_11	Passikudah Beach	Kalkudah	Eastern Province	Beaches	7.9299939	81.5611852	4.56	160	['Swimming', 'Surfing', 'Sunbathing', 'Snorkeling',
LOC_12	Gregory Lake	Nuwara Eliya	Central Province	Bodies of Water	6.95863634	80.77845098	3.96	256	['Boating', 'Walking Trails']
LOC_13	Kandy Lake	Kandy	Central Province	Bodies of Water	7.2912017	80.6420565	3.94	301	['Boating', 'Walking Trails']
LOC_14	Tissa Wewa	Tissamaharama	Southern Province	Bodies of Water	6.2895818	81.2821647	4.43	91	['Historical Exploration', 'Architectural Marvel']
LOC_15	Twin Baths (Kuttam Pokuna)	Anuradhapura	North Central Province	Bodies of Water	8.1995638	80.6326916	3.89	191	['Historical Exploration', 'Architectural Marvel']
LOC_16	Ambewela Farms	Nuwara Eliya	Central Province	Farms	6.8917934	80.8038646	3.9	136	['Factory Tours', 'Agricultural Workshops']
LOC_17	Bluefield Tea Gardens	Nuwara Eliya	Central Province	Farms	7.2564996	80.7214417	4.22	314	['Factory Tours', 'Agricultural Workshops']
LOC_18	Dambatenne Tea Factory	Haputale	Uva Province	Farms	6.7833795	81.0034028	3.36	172	['Factory Tours', 'Agricultural Workshops']
LOC_19	Damro Labookellie Tea Centre	Nuwara Eliya	Central Province	Farms	7.2564996	80.7214417	4.04	90	['Factory Tours', 'Agricultural Workshops']
LOC_20	Glenloch Tea Factory	Katukitula	Central Province	Farms	7.0771474	80.675667	3.82	302	['Factory Tours', 'Agricultural Workshops']

RECOMMENDATION SYSTEM

a). Three-tier approach

- Content-Based: Matches user activities with location offerings (40% weight)
- Collaborative: Leverages similar travelers' ratings (35% weight)
- Machine Learning: Predicts ratings using XGBoost model (25% weight)

b). Adaptive weighting adjusts based on available data

XGBOOST

```python
def _train_enhanced_model(self):
    """Train an enhanced XGBoost model with more features and hyperparameter tuning"""
    # Encode locations for ML
    location_encoder = LabelEncoder()
    location_encoder.fit(self.reviews_df['Location_Name'])
    self.reviews_df['Location_Name_encoded'] = location_encoder.transform(self.reviews_df['Location_Name'])
    self.locations_df['Location_Name_encoded'] = location_encoder.transform(self.locations_df['Location_Name'])

    # Prepare feature set - include all encoded features and engineered features
    features = ['User_Age_Group_encoded', 'Gender_encoded', 'Travel_Companion_encoded', 'Location_Name_encoded']

    # Add location type if available
    if 'Location_Type_encoded' in self.reviews_df.columns:
        features.append('Location_Type_encoded')

    # Add user country if available
    if 'User_Country_encoded' in self.reviews_df.columns:
        features.append('User_Country_encoded')

    # Add user rating behavior features if available
    for col in ['Avg_Rating', 'Rating_Count', 'Rating_Variability']:
        if col in self.reviews_df.columns:
            features.append(col)

    # Add activity features
    features.append('Activity_Match_Ratio')
    features.append('Activity_Count')

    # Add activity-specific features
    for col in self.reviews_df.columns:
        if col.startswith('Has_'):
            features.append(col)
```

```
XGBoost model performance: RMSE=0.4891, R²=0.7655
Interpretation:
  - RMSE: Average prediction error is 0.49 stars
  - R²: Model explains 0.77 of rating variance (excellent)
```

RANDOM FOREST

```python
def _train_rf_model(self):
    """Train a Random Forest model with hyperparameter tuning"""
    start_time = time.time()  # Start measuring training time

    # Encode locations for ML
    location_encoder = LabelEncoder()
    location_encoder.fit(self.reviews_df['Location_Name'])
    self.reviews_df['Location_Name_encoded'] = location_encoder.transform(self.reviews_df['Location_Name'])
    self.locations_df['Location_Name_encoded'] = location_encoder.transform(self.locations_df['Location_Name'])

    # Prepare feature set - include all encoded features and engineered features
    features = ['User_Age_Group_encoded', 'Gender_encoded', 'Travel_Companion_encoded', 'Location_Name_encoded']

    # Add location type if available
    if 'Location_Type_encoded' in self.reviews_df.columns:
        features.append('Location_Type_encoded')

    # Add user country if available
    if 'User_Country_encoded' in self.reviews_df.columns:
        features.append('User_Country_encoded')

    # Add user rating behavior features if available
    for col in ['Avg_Rating', 'Rating_Count', 'Rating_Variability']:
        if col in self.reviews_df.columns:
            features.append(col)

    # Add activity features
    features.append('Activity_Match_Ratio')
    features.append('Activity_Count')

    # Add activity-specific features
    for col in self.reviews_df.columns:
        if col.startswith('Has_'):
            features.append(col)
```

```
Random Forest model performance: RMSE=0.5235, R²=0.6835
Interpretation:
  - RMSE: Average prediction error is 0.52 stars
  - R²: Model explains 0.68 of rating variance (good)
```

OUTPUT

```
======================================================================
                    🏆 TOP RECOMMENDED LOCATIONS 🏆
======================================================================

🥇 TOP PICK: Wilpattu National Park (📍 Nature & Wildlife Areas)
   ⭐ Overall Score: 0.86/1.0
   🎯 Activity Match: 3/4 activities
   🏆 Ratings: CBF: 0.94, CF: 4.33, ML: 3.57
   🎪 Activities: 🐂 Wildlife Spotting, 🦅 Bird Watching, 🔍 Nature Walks
   📝 Based on 18 reviews
```

LOCATION ITINERARY SYSTEM

a). Smart Alternatives based on ratings and nearby.

b). Nearby Alternatives based on the distance of the selected location.

c). Top Rated Alternatives based on Travelers ratings.

```python
# 1. Get details for start and end cities
start_info = get_city_details_combined(start_city, locations_dataset)
end_info = get_city_details_combined(end_city, locations_dataset)

start_province = start_info["province"]
end_province = end_info["province"]

print(f"\nStart: {start_info['name']} in {start_province}")
print(f"End: {end_info['name']} in {end_province}")

# Track locations to exclude from city attractions (to avoid duplication)
excluded_locations = [start_info["name"], end_info["name"]]
```

```python
# 2. Find the province corridor (path of provinces between start and end)
province_corridor = find_province_path(start_province, end_province)
print(f"\nProvince corridor: {' → '.join(province_corridor)}")
```

```python
# 3. Get plan pool locations by province corridor
corridor_plan = get_plan_pool_by_corridor(province_corridor, locations_dataset,
 plan_pool_locations)
```

```python
itinerary = []

# Start with start city
itinerary.append({
    "name": start_info["name"],
    "type": "City" if start_info.get("type") is None else start_info["type"],
    "province": start_province,
    "is_city": True,
    "lat": start_info["lat"],
    "lng": start_info["lng"]
})

# Current position is start city
current_lat, current_lng = start_info["lat"], start_info["lng"]
```

OUTPUT

```
📋 TRIP SUMMARY
================================================================================
📍 FROM: Galle Fort (Southern Province)
🗺️ TO: Anuradhapura (North Central Province)
🔴 TOTAL DISTANCE: 309.9 km
🏛️ ATTRACTIONS: 5
🛣️ ROUTE: 🏖️ Southern Province → 🗺️ Western Province → 🌲 North Western Province → 🏛️
 North Central Province
```

PLANNER SYSTEM

- Get the start and end city and create the plan for the selected plan pool items for the given radius.
- Locations are matched based on the distance from the start city to end city.
- Province based route matching included.

```python
# Convert numeric columns and fill missing values with 0
locations["Avg_Rating"] = pd.to_numeric(locations["Avg_Rating"], errors='coerce').fillna(0)
locations["Review_Count"] = pd.to_numeric(locations["Review_Count"], errors='coerce').fillna(0)
# Fill missing activities with an empty string
locations['Activities'] = locations['Activities'].fillna('')

# Normalize numerical features using MinMaxScaler
scaler = MinMaxScaler()
locations[["Avg_Rating", "Review_Count"]] = scaler.fit_transform(locations[["Avg_Rating",
 "Review_Count"]])
```

```python
plan = get_itinerary_plan(
    location_name="Galle Fort",
    included_provinces=["Central Province", "Southern Province", "Western Province"],
    top_n=5
)
```

```python
def recommend_by_activities(user_activities, top_n=5):

    # Clean user activities
    user_set = set([act.strip().lower() for act in user_activities])

    scores = {}
    for idx, row in locations.iterrows():
        loc_set = clean_activities(row["Activities"])
        if not user_set and not loc_set:
            sim = 1.0
        else:
            union = user_set.union(loc_set)
            sim = len(user_set.intersection(loc_set)) / len(union) if union else 0.0
        scores[row["Location_Name"]] = sim

    scores_df = pd.DataFrame(scores.items(), columns=["Location",
     "Activities_Jaccard_Similarity"])
    scores_df = scores_df.sort_values("Activities_Jaccard_Similarity", ascending=False)
    return scores_df.head(top_n)
```

OUTPUT

```
📍 LOCATION: Galle Fort
================================================================================

📌 LOCATION DETAILS
------------------------------
📍 Type: 📍 Historic Sites
🌄 Province: 🏜️ Southern Province
⭐ Rating: ⭐⭐⭐⭐ (4.5/5)
🎯 Activities: 👀 ['Sightseeing', 📸 'Photography', 📜 'Learning History', 🧭 'Cultural Tours']
```

```
🌟 TOP RECOMMENDED PLACES (HYBRID SYSTEM)
------------------------------------------------------------
1. 📍 Sigiriya The Ancient Rock Fortress
   📊 Match: 🔘 0.66
   ⭐ Rating: ⭐⭐⭐⭐✨ (4.7/5)
   🌄 Province: 🏔️ Central Province
```

```
📍 SIMILAR HISTORIC SITES LOCATIONS
------------------------------------------------------------
1. 📍 Sigiriya The Ancient Rock Fortress
   ⭐ Rating: ⭐⭐⭐⭐✨ (4.7/5)
   🌄 Province: 🏔️ Central Province
```

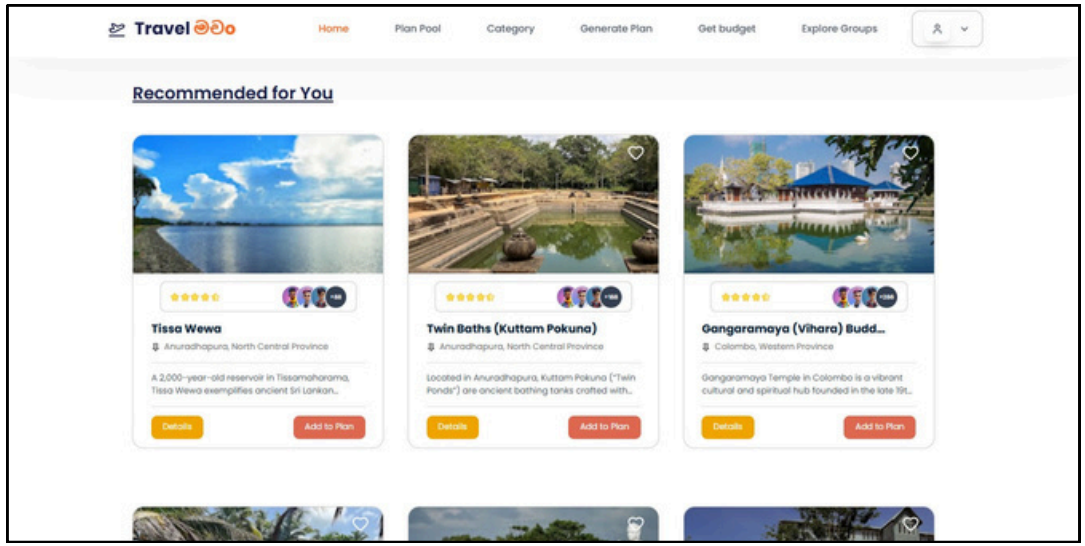
COMPLETION OF THE PROJECT

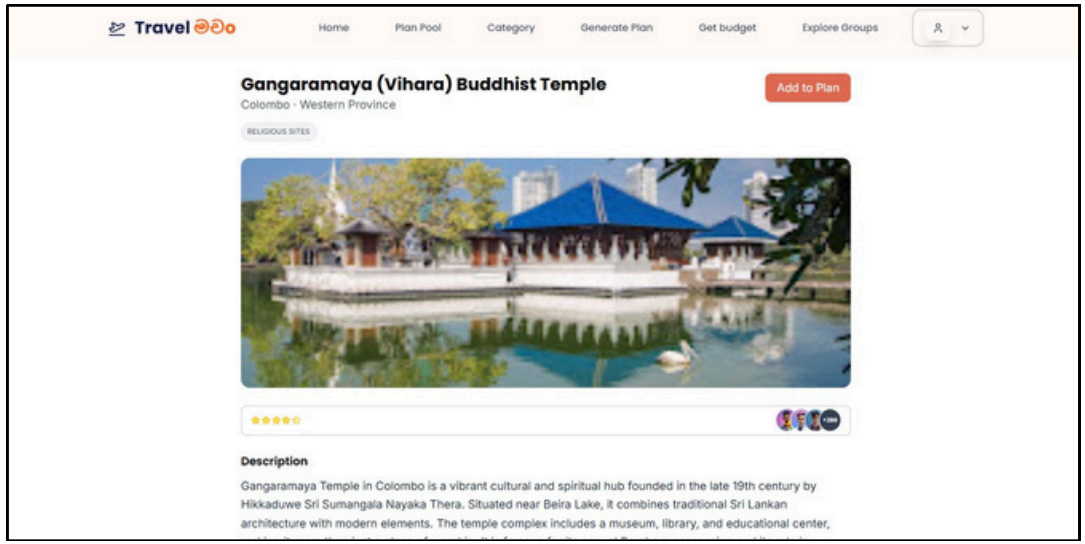
90%

KEY FUNCTIONALITIES

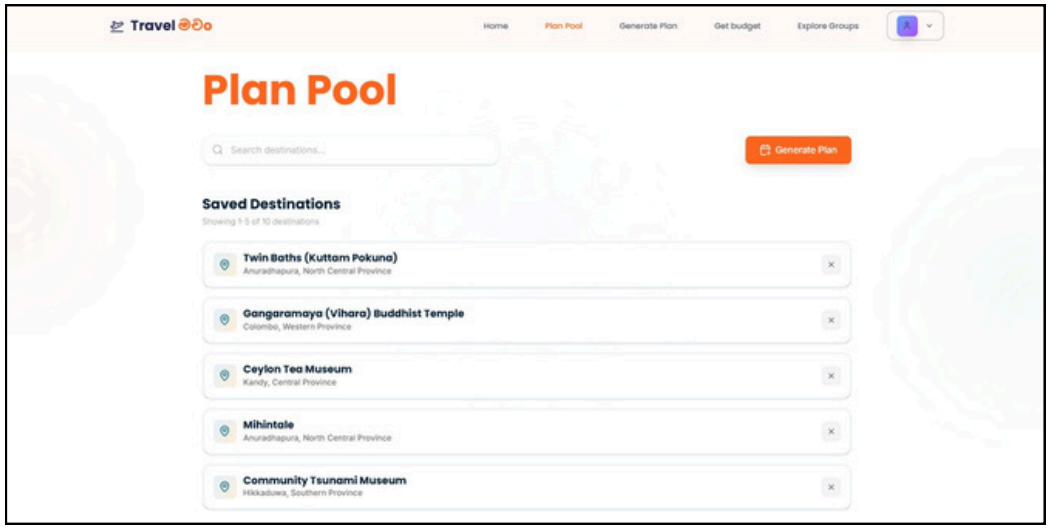
- **Preference Collection & Personalization:** Collect age group, travel companion, and other user demographic details to deliver personalized suggestions.
- **AI-Powered Recommendations:** Uses XGBoost Model to suggest destinations and attractions that best match user preferences.
- **Trip Planning Workflow:** Generate trip planning workflow for the given start and end city with the saved travel locations.
- **Itinerary Management:** Provides similar alternatives and smart suggestion for lgenerated plan locations.
- **Plan Optimization:** Optimize the trip planning route after the itinerary changes according to the distances of each location.

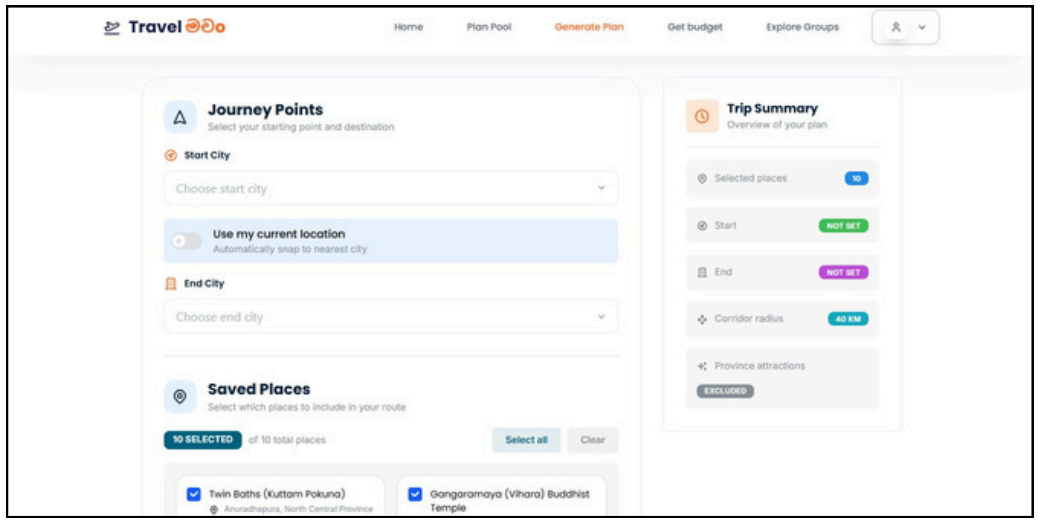
WORKING PRODUCT



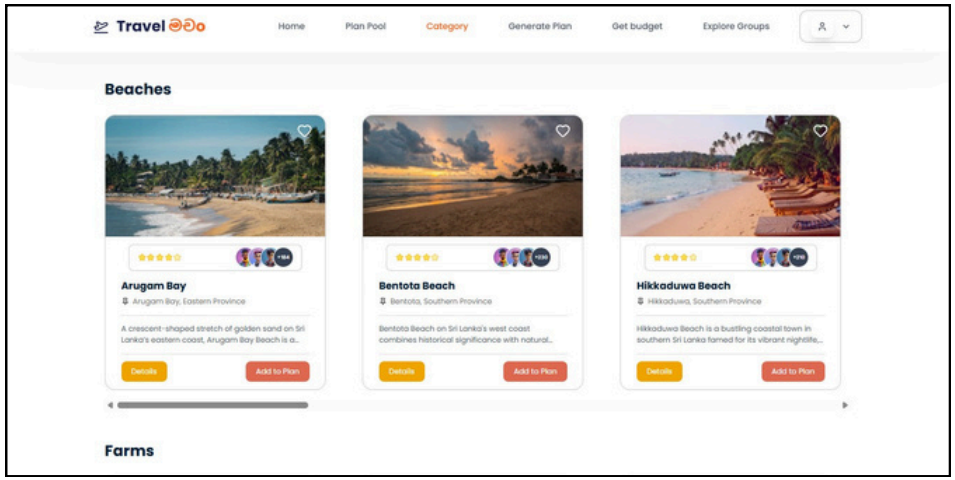


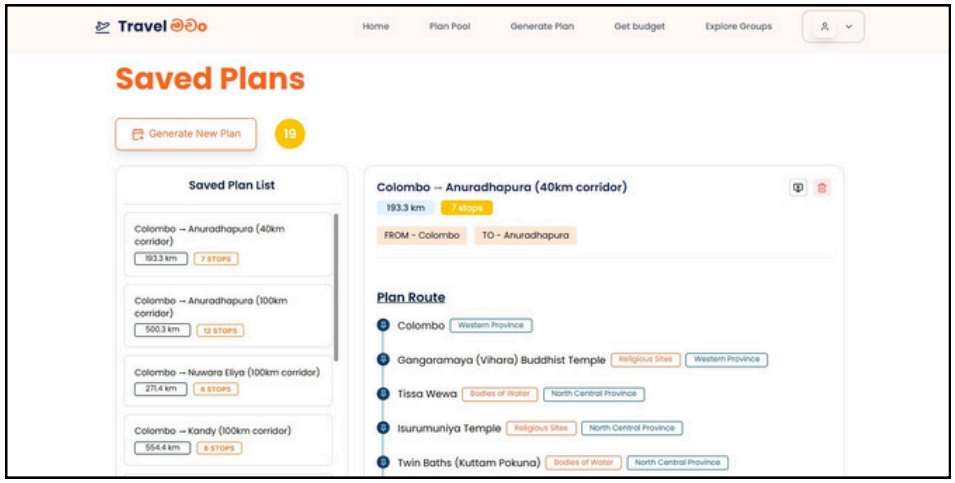
WORKING PRODUCT



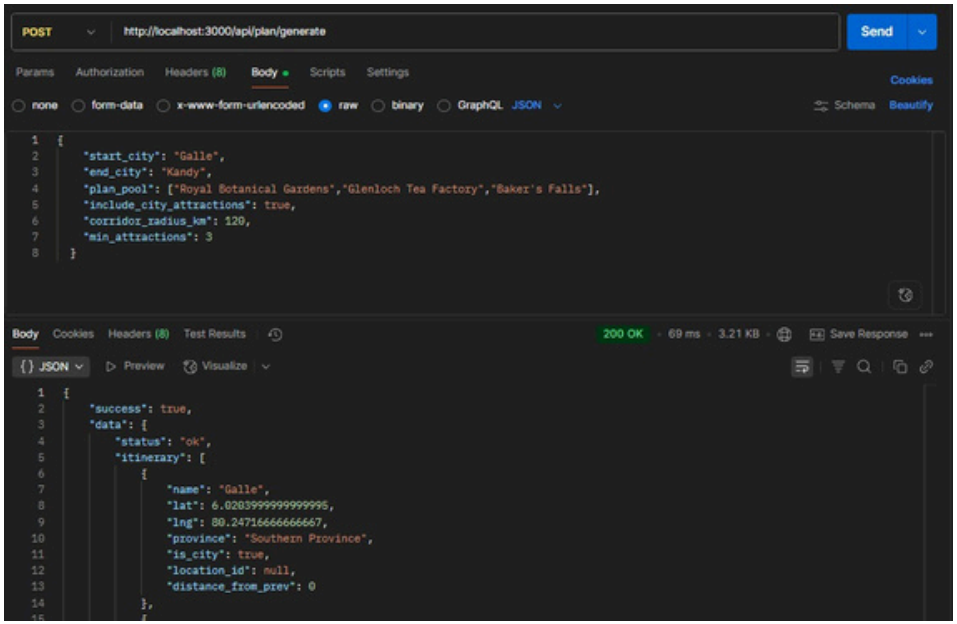


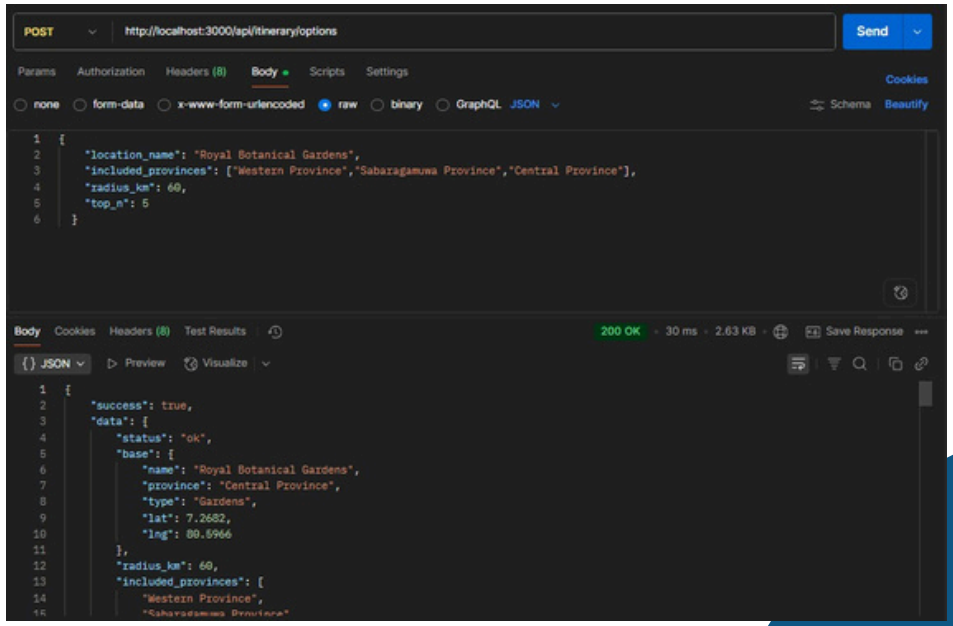
WORKING PRODUCT





POSTMAN API CHECKS





REFERENCES

[1] Overgoor, G., Chica, M., Rand, W., & Weishampel, A. (2019). Letting the computers take over: Using AI to solve marketing problems. California Management Review, 61(4), 156-185.

[2] Topol, E. J. (2019). High-performance medicine: the convergence of human and artificial intelligence. Nature medicine, 25(1), 44-56.

[3] Duarte, L., Torres, J., Ribeiro, V., & Moreira, I. (2020). Artificial Intelligence Systems applied to tourism: A Survey. arXiv preprint arXiv:2010.14654.

[4] https://wttc.org/LinkClick.aspx?fileticket=2VluL2e-nAA%3D&portalid=0&utm

[5] Gretzel, U., Sigala, M., Xiang, Z., & Koo, C. (2015). Smart tourism: foundations and developments. Electronic markets, 25, 179-188.

[6] Yang, X., Zhang, L., & Feng, Z. (2024). Personalized tourism recommendations and the E-tourism user experience. Journal of Travel Research, 63(5), 1183-1200.

IT21835728 - THUWAKARAN. R

SPECIALIZING - SOFTWARE ENGINEERING

INTRODUCTION

01. BACKGROUND
02. RESEARCH QUESTIONS
03. MAIN AND SUB OBJECTIVES
04. METHODOLOGY
05. COMPLETION OF PROJECT

BACKGROUND

Static Budgeting Tools

- Traditional budgeting tools rely on outdated, static data and manual processes, which don't adjust to fluctuating travel costs or changing user preferences.

Lack of Personalization

- Current platforms offer generalized recommendations that don't meet the unique needs of travelers, such as varying group sizes, budgets, and dynamic itineraries.

Limited Real-Time Adaptation

- Without real-time data integration or dynamic adaptability, platforms fail to adjust to unforeseen changes like weather, price fluctuations, or schedule conflicts, leaving travelers with inefficient plans.

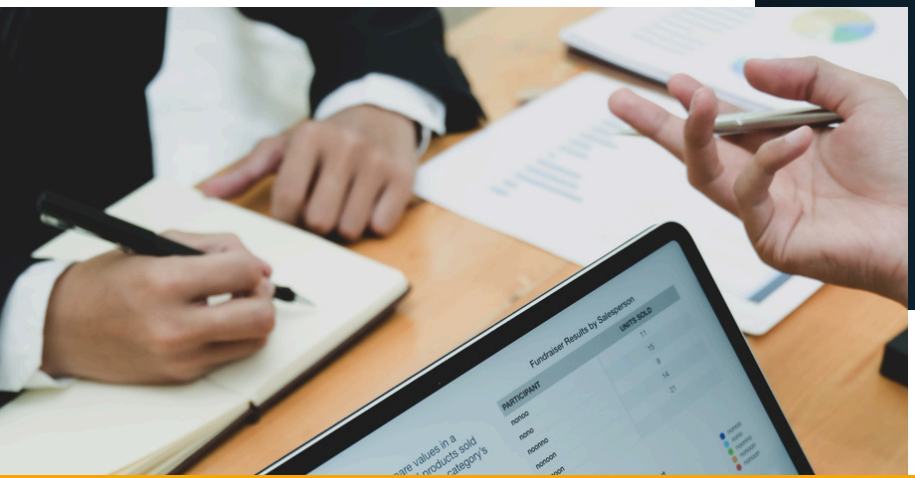
RESEARCH Questions?

01 What are the limitations of current tools for travel budget allocation?

02 How can predictive algorithms and real-time data improve the accuracy and reliability of travel budgeting?

03 How can user-centric design enhance accessibility and usability for a broader audience?

04 How does the proposed system address these gaps and stand out from existing solutions?



OBJECTIVES

- ## Develop a Dynamic Budget System

Create an intelligent Travel Budget Allocation System that adapts to real-time data and user preferences, offering personalized and accurate budget recommendations.

- ## Implement Predictive Algorithms

ntegrate predictive models to estimate travel costs (accommodations, transport, meals) using historical data and real-time inputs for accurate forecasts.

- ## Design a User-Centric Interface

Build an intuitive interface allowing travelers to customize budget preferences and receive tailored recommendations based on their travel style and goals.

- ## Integrate APIs & Validate System

Integrate external APIs for real-time cost data and conduct rigorous testing to ensure system accuracy, reliability, and adaptability to changing market conditions and user needs.

SMART BUDGET ALLOCATION & ADAPTIVE PLANNING

"Our budget system tailors recommendations based on traveler type and package, offering options like affordable hostels for solo travelers and resorts for families, ensuring a personalized, budget-friendly experience."

Customizable Packages: Basic, Moderate, and Premium options tailored to different traveler types, offering varying levels of amenities and services.

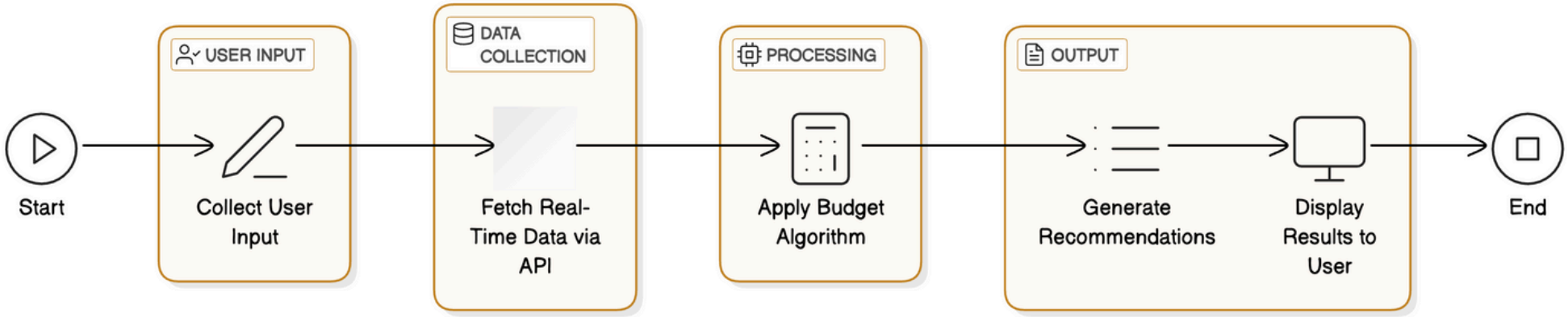
Diverse Travel Companions: Plans customized for Solo, Couple, Family, and Friends, based on group size and preferences.

Dynamic Cost Predictions: Predicts breakdowns for Transport, Stay, Activities, Food, and Accommodation, adjusting based on real-time pricing and user preferences.

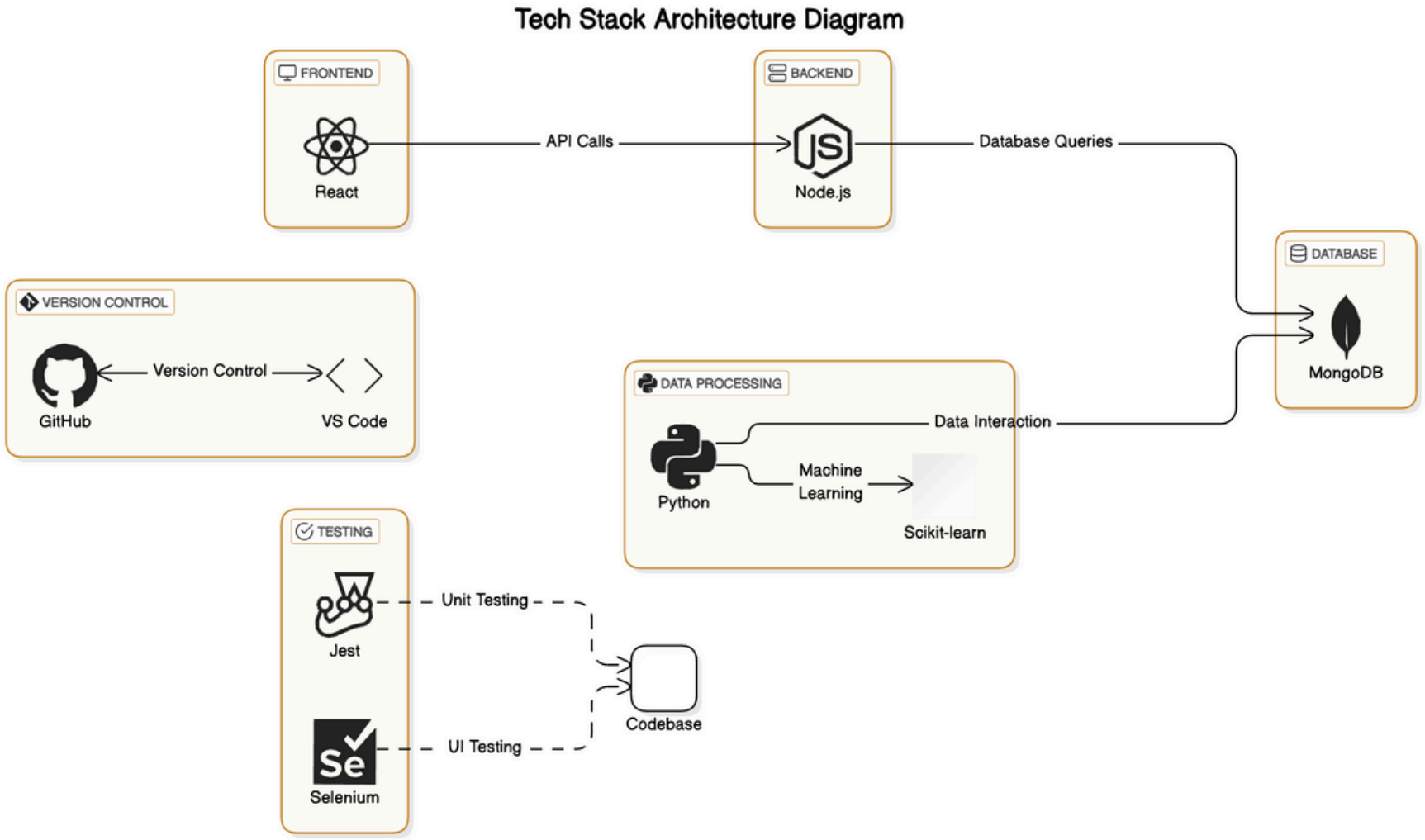
METHODOLOGY

01. SYSTEM DIAGRAM
02. TOOLS AND TECHNOLOGIES
03. REQUIREMENTS
04. GITHUB COMMITS

BUDGET PREDICTION SYSTEM DIAGRAM



TOOLS & TECHNOLOGIES



REQUIREMENTS

FUNCTIONAL

- Real-Time Data Integration
- Accurate Predictions
- Customizable Suggestions
- Notification System
- Collaboration

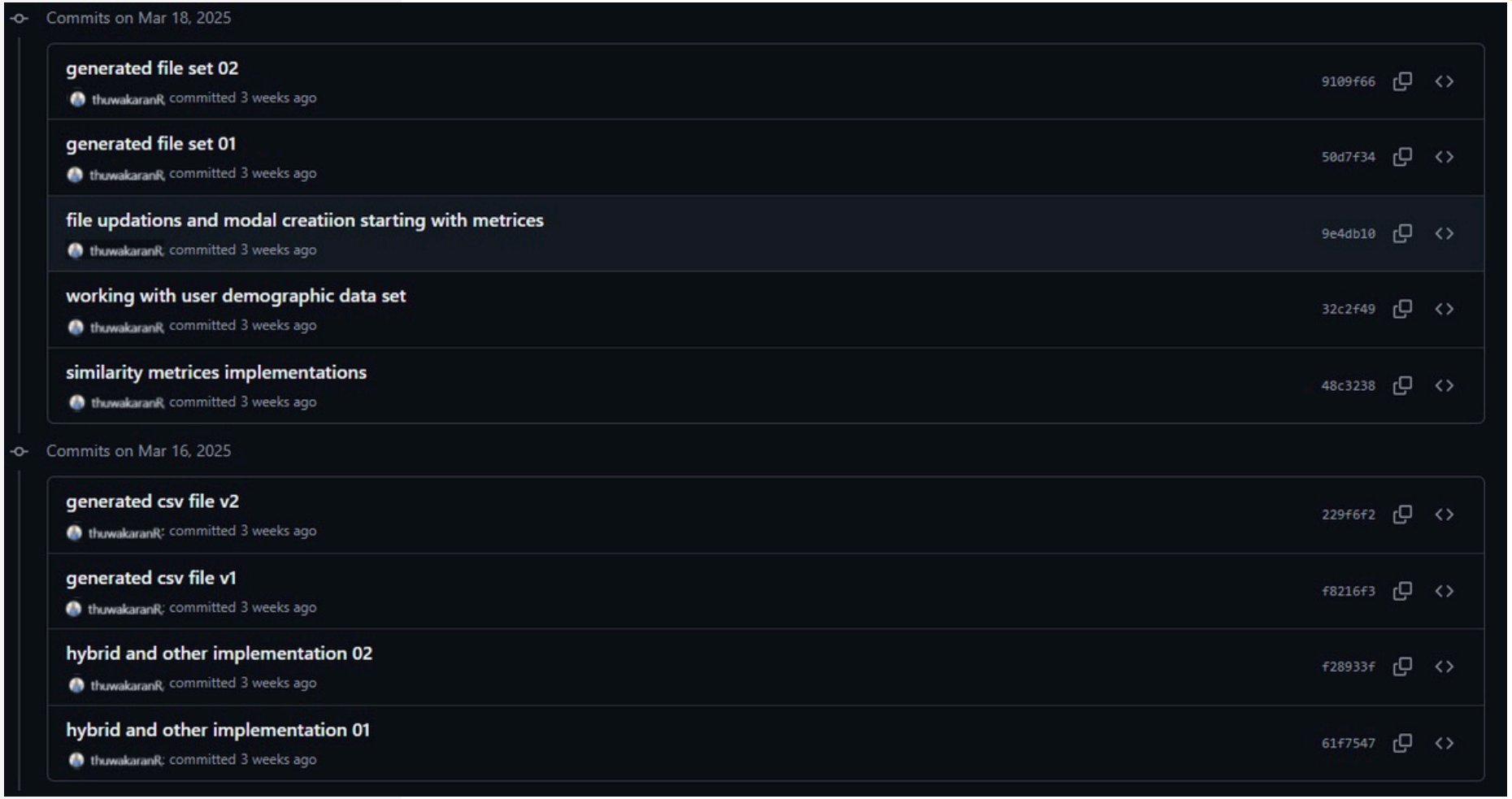
NON-FUNCTIONAL

- Reliability of Predictions
- Fast Response Times
- User-Friendly Interface
- Maintainability
- Localization

BEST PRACTICES

- Code quality
- Web security
- Modal quality
- Data Privacy

GITHUB COMMITS



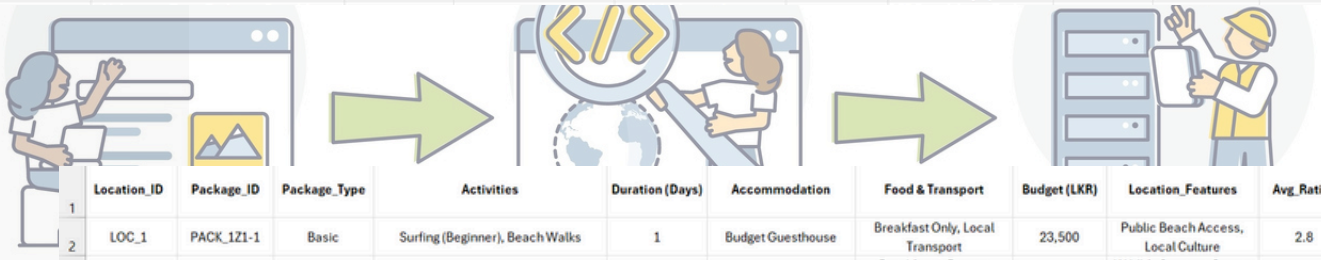
COMPLETION OF THE PROJECT

COMPLETION OF THE PROJECT

50%

DATA COLLECTION

2617	LOC_76	PACK_76Z3-2	Premium	Sunrise Visit, Attend Thaipoosam Festival with Private Viewing, Explore Temple with Private Guide	1	Luxury Resort	All-Inclusive, Private Transport	20,500	Exclusive Sunrise Visit, Festival Experience	3.9	70	Couple
2618	LOC_76	PACK_76Z2-3	Moderate	Visit Temple, Attend Special Worship, Visit the Garden and Pond	1	Boutique Hotel	Full Board, Shared Transport	15,300	Worship, Spiritual Serenity	3.6	55	Couple
2619	LOC_76	PACK_76Z3-3	Premium	Private Cultural Experience: Attend Special Worship, Explore Temple Architecture, Visit the Pond	1	Luxury Resort	All-Inclusive, Private Transport	21,500	Tailored Experience, Private Guide	2.7	80	Couple
2620	LOC_76	PACK_76Z2-2	Moderate	Attend the Annual Festival, Explore Temple Parades with Music & Dance	1	Boutique Hotel	Full Board, Shared Transport	15,000	Festival Atmosphere, Parades	3.9	50	Couple
2621	LOC_76	PACK_76Z3-4	Premium	Full Day Experience: Nallur Kandaswamy Temple, Lord Muruga's Worship, Special Blessings	1	Luxury Boutique Stay	All-Inclusive, VIP Transport	22,500	Exclusive Experience, VIP Worship	4.9	90	Couple
2622	LOC_1	PACK_1Z1-1	Basic	Surfing (Beginner), Beach Walks	1	Budget Guesthouse	Breakfast Only, Local Transport	34,500	Public Beach Access, Local Culture	2.8	95	Family
2623	LOC_1	PACK_1Z1-2	Basic	Lagoon Safari, Elephant Rock Visit	1	Homestay	Breakfast & Dinner, Shared Transport	38,500	Wildlife Spotting, Scenic Views	3.2	80	Family
2624	LOC_1	PACK_1Z2-1	Moderate	Surfing (Intermediate), Jeep Safari in Kumana	2	Beachfront Hotel	Half Board, Private Transport	51,000	Beachfront View, Guided Tours	4.1	120	Family
2625	LOC_1	PACK_1Z2-2	Moderate	Snorkeling, Yoga, Local Cuisine Tour	2	Boutique Villa	Full Board, Shared Transport	54,000	Traditional Food, Cultural Immersion	3.7	110	Family
2626	LOC_1	PACK_1Z3-1	Premium	Private Surfing Lessons, Private Boat Tours	2	Luxury Resort	Full Board, Private Chauffeur	81,000	Private Beach Access, Luxury Spa	4.6	150	Family

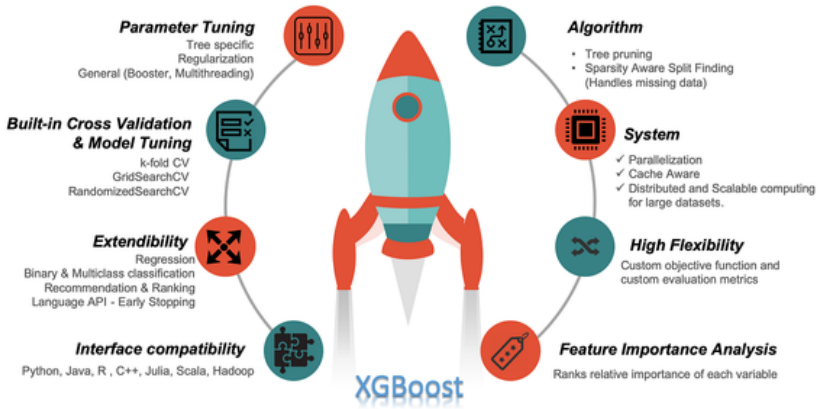


	Location_ID	Package_ID	Package_Type	Activities	Duration (Days)	Accommodation	Food & Transport	Budget (LKR)	Location_Features	Avg_Rating	Review_Count	Travel_Companion
1												
2	LOC_1	PACK_1Z1-1	Basic	Surfing (Beginner), Beach Walks	1	Budget Guesthouse	Breakfast Only, Local Transport	23,500	Public Beach Access, Local Culture	2.8	95	Solo
3	LOC_1	PACK_1Z1-2	Basic	Lagoon Safari, Elephant Rock Visit	1	Homestay	Breakfast & Dinner, Shared Transport	27,500	Wildlife Spotting, Scenic Views	3.2	80	Solo
4	LOC_1	PACK_1Z2-1	Moderate	Surfing (Intermediate), Jeep Safari in Kumana	2	Beachfront Hotel	Half Board, Private Transport	40,000	Beachfront View, Guided Tours	4.1	120	Solo
5	LOC_1	PACK_1Z2-2	Moderate	Snorkeling, Yoga, Local Cuisine Tour	2	Boutique Villa	Full Board, Shared Transport	43,000	Traditional Food, Cultural Immersion	3.7	110	Solo
6	LOC_1	PACK_1Z3-1	Premium	Private Surfing Lessons, Private Boat Tours	2	Luxury Resort	Full Board, Private Chauffeur	70,000	Private Beach Access, Luxury Spa	4.6	150	Solo
7	LOC_1	PACK_1Z3-2	Premium	Helicopter Tour, Exclusive Beachfront Dining	3	5-Star Resort	All-Inclusive, Luxury Transport	90,000	VIP Beachfront, Exclusive Experience	4.3	130	Solo
8	LOC_1	PACK_1Z3-3	Premium	Yala Safari Camping, Traditional 'Oruwa' Rafting	2	Luxury Safari Camp	Full Board, Private Guide	100,000	Unique Wildlife Encounters	4.8	140	Solo
9	LOC_1	PACK_1Z1-3	Basic	Beach Volleyball, Turtle Watching	1	Hostel Dorm	Breakfast Only, Shared Transport	22,500	Public Beach Access, Marine Life	2.5	70	Solo
10	LOC_1	PACK_1Z1-4	Basic	Mangrove Kayaking, Fishing Experience	1	Eco Lodge	Breakfast & Dinner, Local Transport	25,000	Scenic Views, Local Culture	3.1	65	Solo
11	LOC_1	PACK_1Z2-3	Moderate	Stand-up Paddleboarding, Cultural Dance Show	2	Eco Boutique Hotel	Full Board, Private Transport	41,500	Traditional Food, Cultural Immersion	4.4	100	Solo
12	LOC_1	PACK_1Z2-4	Moderate	Hiking, Tea Plantation Visit	2	Hillside Retreat	Half Board, Shared Transport	39,500	Scenic Views, Nature Trails	3.8	105	Solo
13	LOC_1	PACK_1Z2-5	Moderate	Sunset Cruise, Local Handicraft Workshop	2	Riverside Bungalow	Full Board, Local Transport	42,000	Traditional Crafts, River Views	4	90	Solo

MODEL SELECTION

-XGBoost Regressor with RandomizedSearchCV-

- **Explored XGBoost Regressor to predict the budget for a given travel input setup using customized features.**
- **Used RandomizedSearchCV for tuning hyperparameters to improve prediction accuracy and model generalization.**



- Fine-tuned Performance
- Better Accuracy
- Flexible and Powerful
- Efficient and Fast

BUDGET PREDICTION – CODE EXECUTION FLOW

```
[ ] df = pd.concat([df, df_activities], axis=1)
    df.drop(columns=['Activities'], inplace=True)  # Drop original activities column

    # Add Travel_Companion column and encode it
    travel_companion_map = {'Solo': 0, 'Couple': 1, 'Family': 2, 'Friends': 3}
    df['Travel_Companion'] = df['Travel_Companion'].map(travel_companion_map)

    # Define Rating Ranges (0-1, 1-2, 2-3, 3-4, 4-5)
    def categorize_rating(rating):
        return f"{int(rating)}-{int(rating) + 1}"

    df['Rating_Range'] = df['Avg_Rating'].apply(categorize_rating)
```

```
[ ] # Select features and target variable
    X = df.drop(columns=['Package_ID', 'Accommodation', 'Food & Transport', 'Budget (LKR)', 'Rating_Range'])
    y = df['Budget (LKR)']

    # Split dataset into training (80%) and testing (20%)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ] # Initialize XGBoost regressor model
    xgb_model = XGBRegressor(random_state=42)

    # Train the model
    xgb_model.fit(X_train, y_train)
```

```
[ ] # Make predictions
    y_pred = xgb_model.predict(X_test)

    # Evaluate the model
    mae = mean_absolute_error(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred) ** 0.
    r2 = r2_score(y_test, y_pred)

    # Display evaluation results
    print(f"✅ XGBoost Model Evaluation:\nMAE: {mae
```

```
✅ XGBoost Model Evaluation:
MAE: 2769.10 LKR
RMSE: 5242.96 LKR
```

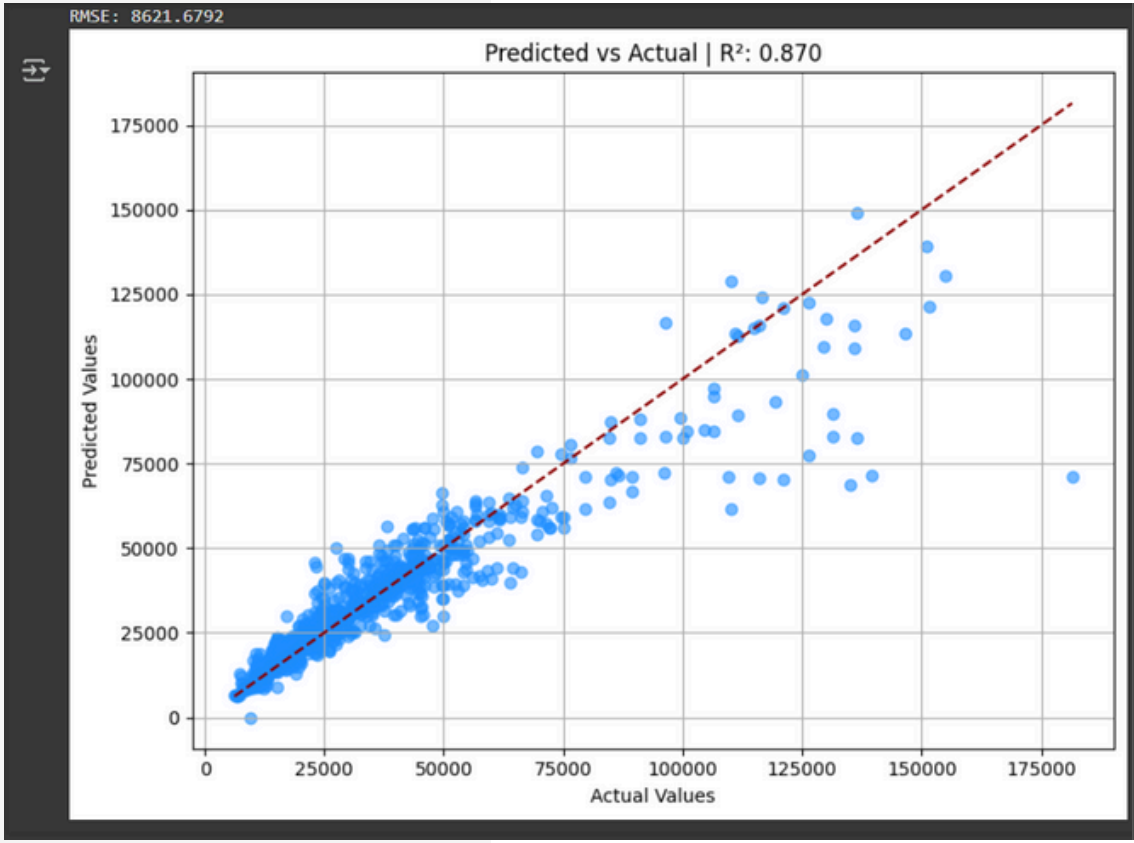
XGBOOST

```
    if selected_packages:
        final_results = pd.DataFrame(selected_packages)
        final_results = final_results[[
            'Location', 'Package_ID', 'Package_Type', 'Days'
            'Food & Transport', 'Avg_Rating', 'Activities',
        ]]

        # Format the predicted budget to two decimal places
        final_results['Predicted_Budget'] = final_results['P

        total_duration = final_results['Days'].sum()
        total_budget = round(total_budget, 2)
        possible_combinations.append((final_results, total_d

return possible_combinations
```

```
dict(X_test)

# Evaluate
r2_tuned = r2_score(y_test, y_pred_tuned)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_tuned))  # fixed RMSE calculation

# Output
print("\nBest Parameters Found:")
print(random_search.best_params_)
print(f"R² Score: {r2_tuned:.4f}")
print(f"RMSE: {rmse:.4f}")

# Plotting predicted vs actual
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_tuned, alpha=0.6, color='dodgerblue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='darkred', linestyle='--')
plt.title(f"Predicted vs Actual | R²: {r2_tuned:.3f}")
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.grid(True)
plt.tight_layout()
plt.show()
```

ACCURACY & ANALYSIS

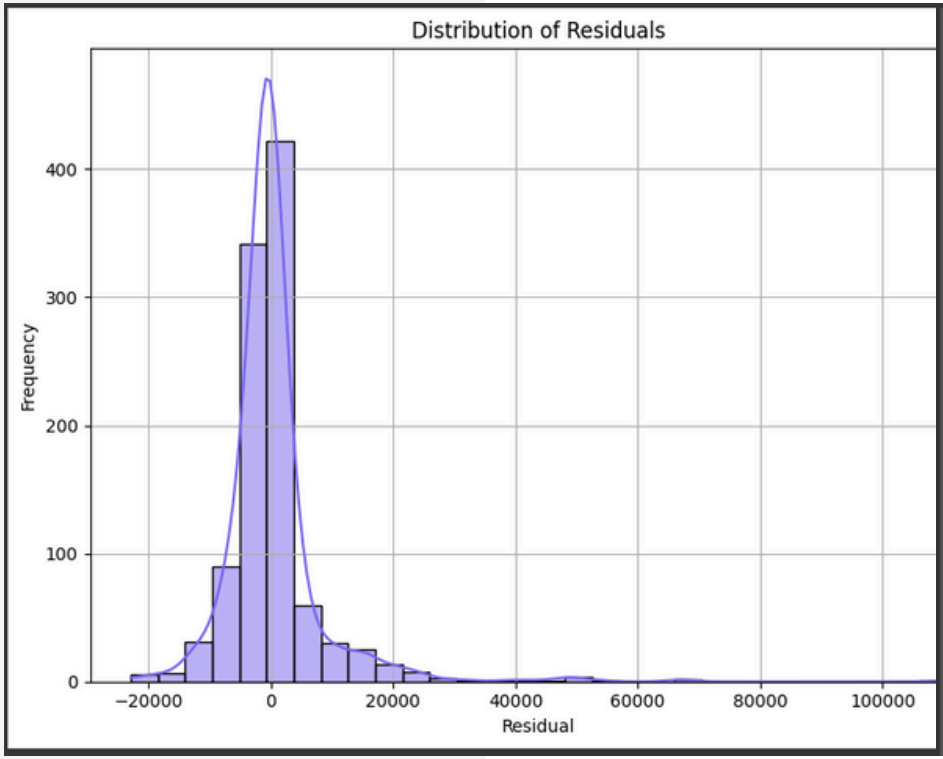
- Plotting




```python
# Simplified hyperparameter grid to reduce model capacity
param_grid = {
    'n_estimators': [50, 75, 100],
    'learning_rate': [0.05, 0.1, 0.2],
    'max_depth': [2, 3],
    'min_child_weight': [5, 10],
    'gamma': [0.2, 0.3],
    'subsample': [0.6, 0.7],
    'colsample_bytree': [0.6, 0.7],
    'reg_alpha': [0, 0.1],
    'reg_lambda': [0, 0.1]
}

# Set up RandomizedSearchCV
random_search = RandomizedSearchCV(
    XGBRegressor(random_state=42, n_jobs=-1),
    param_distributions=param_grid,
    n_iter=50,  # fewer iterations needed here
    cv=5,
    verbose=2,
    random_state=42,
    n_jobs=-1
)

# Train the model
random_search.fit(X_train, y_train)
best_model = random_search.best_estimator_
```

ACCURACY & ANALYSIS

- Plot Histogram




```python
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate residuals
residuals = y_test - y_pred_tuned

# Plot histogram of residuals
plt.figure(figsize=(8, 6))
sns.histplot(residuals, kde=True, bins=30,
plt.title("Distribution of Residuals")
plt.xlabel("Residual")
plt.ylabel("Frequency")
plt.grid(True)
plt.tight_layout()
plt.show()
```

BUDGET PREDICTION & ITERATIVE RESULTS

"HERE'S HOW OUR BUDGET SYSTEM FITS IN. IT PULLS LIVE COST DATA, APPLIES PREDICTION LOGIC, AND RETURNS AN OPTIMIZED, ADAPTIVE BUDGET PLAN BASED ON USER INPUTS AND PREFERENCES."

```python
locations = ["LOC_11", "LOC_42", "LOC_6"]
package = "Moderate"
total_days = 8
rating_range = "4-5"
travel_companion = "Solo"

predicted_packages = predict_budget_multiple_options(locations, package, total_days, rating_range, travel_companion, df, best_model, le_location, le_packag

# Display the results
for i, (packages, total_duration, total_budget) in enumerate(predicted_packages, 1):
    print("=" * 72)
    print(f"{'✨ Trip Plan ' + str(i) + ' ✨':^72}")  # Center-aligned title
    print("=" * 72)

    # Display the package details
    print(packages)

    # Display travel companion
    print("-" * 72)
    print(f"👪 Travel Companion: {travel_companion}")

    print("-" * 72)
    print(f"🕒 Total Duration: {total_duration} days")
    print(f"💰 Total Budget: {total_budget:.2f} LKR")
    print("=" * 72, "\n")
    print("\n")
```

```
                              ✨ Trip Plan 1 ✨
========================================================================
    Location   Package_ID Package_Type  Days       Accommodation  \
697   LOC_42  PACK_42Z2-4     Moderate   2.0  Seaside Guesthouse
696   LOC_42  PACK_42Z2-1     Moderate   1.0      Boutique Villa
101    LOC_6   PACK_6Z2-3     Moderate   2.0       4-Star Resort

                Food & Transport  Avg_Rating  \
697   Full Board, Shared Transport         4.0
696   Half Board, Shared Transport         4.1
101  Full Board, Private Transport         4.3

                                        Activities Predicted_Budget
697      [Morning Safari, Nature Walk, Spot Flamingos]         25405.94
696  [Birdwatching, Explore Flora & Fauna, Lagoon T...         20413.94
101  [Boat Ride to Trincomalee, Whale Watching (Sea...         51253.43
------------------------------------------------------------------------
👪 Travel Companion: Solo
------------------------------------------------------------------------
🕒 Total Duration: 5.0 days
💰 Total Budget: 97073.31 LKR
========================================================================
```

```
                              ✨ Trip Plan 2 ✨
========================================================================
    Location   Package_ID Package_Type  Days       Accommodation  \
200   LOC_11  PACK_11Z2-8     Moderate   3.0   Beachfront Resort
697   LOC_42  PACK_42Z2-4     Moderate   2.0  Seaside Guesthouse
103    LOC_6   PACK_6Z2-5     Moderate   2.0    Beachfront Hotel

                Food & Transport  Avg_Rating  \
200  Full Board, Private Transport         4.0
697   Full Board, Shared Transport         4.0
103  Full Board, Private Transport         4.1

                                        Activities Predicted_Budget
200  [Relax on Kalkudah Beach, Watersports at Pasik...         26724.60
697      [Morning Safari, Nature Walk, Spot Flamingos]         25405.94
103                  [Diving, Underwater Photography]         43833.84
------------------------------------------------------------------------
👪 Travel Companion: Solo
------------------------------------------------------------------------
🕒 Total Duration: 7.0 days
💰 Total Budget: 95964.38 LKR
========================================================================
```

```
                              ✨ Trip Plan 3 ✨
========================================================================
    Location   Package_ID Package_Type  Days       Accommodation  \
200   LOC_11  PACK_11Z2-8     Moderate   3.0   Beachfront Resort
696   LOC_42  PACK_42Z2-1     Moderate   1.0      Boutique Villa
103    LOC_6   PACK_6Z2-5     Moderate   2.0    Beachfront Hotel

                Food & Transport  Avg_Rating  \
200  Full Board, Private Transport         4.0
696   Half Board, Shared Transport         4.1
103  Full Board, Private Transport         4.1

                                        Activities Predicted_Budget
200  [Relax on Kalkudah Beach, Watersports at Pasik...         23317.40
696  [Birdwatching, Explore Flora & Fauna, Lagoon T...         16623.75
103                  [Diving, Underwater Photography]         28720.92
------------------------------------------------------------------------
👪 Travel Companion: Solo
------------------------------------------------------------------------
🕒 Total Duration: 6.0 days
💰 Total Budget: 68662.07 LKR
========================================================================
```

COMPLETION OF THE PROJECT

90%

KEY FUNCTIONALITIES

- **AI-Powered Predictions** – Generate multiple trip plan options based on user-selected preferences (locations, days, package, ratings, companion).
- **Iterative Plan Options** – Provide alternative detailed itineraries with budget, activities, and duration breakdowns.
- **User Plan Selection** – Enable users to review, choose, and confirm one preferred plan.
- **Plan Management** – Save confirmed plans in the system for later retrieval, with options to view or delete.
- **Chatbot Assistance** – AI chatbot supports users in customizing or refining plans when system-generated ones don't meet needs.
- **Comprehensive API** – FastAPI endpoints for prediction, confirmation, retrieval, deletion, and chatbot integration.
- **PDF Exporting** – Download finalized plans as professional PDF reports for easy sharing.

FAST API – IMPLEMENTATION

PREDICT TRIP PLAN

```python
# Predict trip plan
@app.post("/predict", summary="Predict trip budget options")
async def predict_trip(req: TripRequest):
    try:
        results = predict_budget_multiple_options(
            locations=req.locations,
            package=req.package,
            total_days=req.total_days,
            rating_range=req.rating_range,
            travel_companion=req.travel_companion,
        )

        data_to_save = req.dict()
        data_to_save["predictions"] = [
            {
                "plan": plan.get("plan", []),
                "total_days": plan.get("total_days"),
                "total_budget": plan.get("total_budget"),
                "travel_companion": plan.get("travel_companion"),
            }
            for plan in results
        ]

        predictions_collection.insert_one(data_to_save)

        return {"combinations": data_to_save["predictions"]}

    except Exception as e:
        print("ERROR in /predict:", traceback.format_exc())
        raise HTTPException(status_code=500, detail=f"Prediction failed: {str(e)}")
```

CHAT BOT IMPLEMENTATION

```python
# ChatBot using Cohere
You, last month | 1 author (You)
class ChatRequest(BaseModel):
    message: str

You, last month | 1 author (You)
class ChatResponse(BaseModel):
    reply: str

@app.post("/chatbot", response_model=ChatResponse, summary="Chatbot response using Cohere")
async def chatbot_endpoint(chat_request: ChatRequest):
    user_message = chat_request.message

    try:
        response = co.chat(
            message=user_message,
            model="command-r-plus",
            temperature=0.7,
        )

        return ChatResponse(reply=response.text.strip())

    except Exception as e:
        print("Cohere Error:", traceback.format_exc())
        raise HTTPException(status_code=500, detail=f"Chatbot error: {str(e)}")
```

CONFIRM PLAN

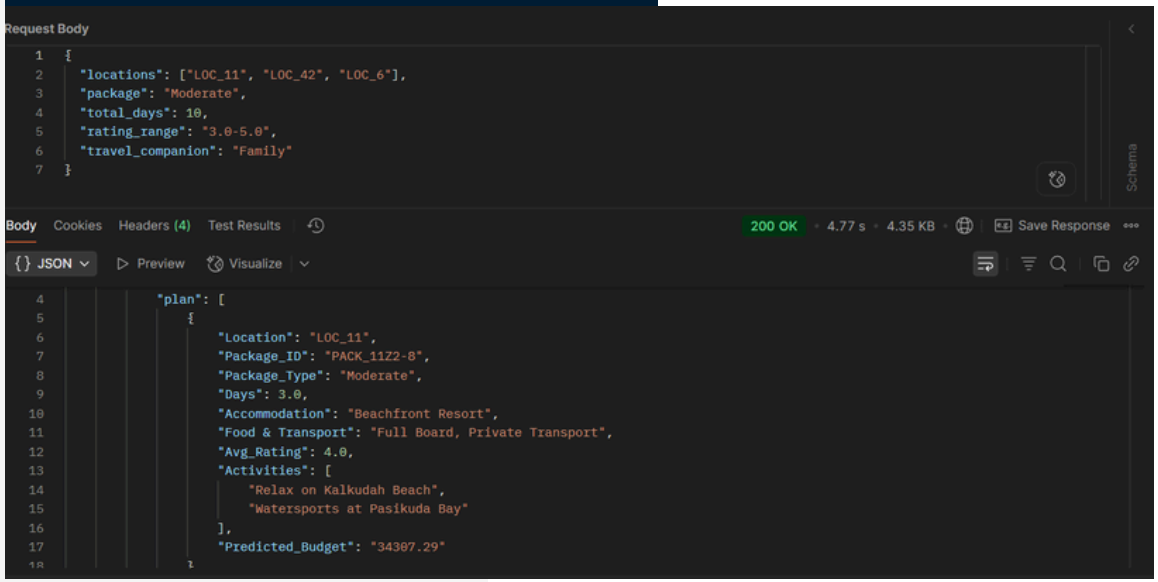
```python
# Confirm selected trip plan
@app.post("/confirm", summary="Confirm a selected trip plan")
async def confirm_plan(req: ConfirmPlanRequest):
    try:
        doc = {
            "plan_number": req.plan_number,
            "package_ids": req.package_ids,
            "confirmed_at": req.confirmed_at.isoformat(),
            "user_id": req.user_id,
            "full_plan": req.full_plan,
        }

        result = confirmed_plans_collection.insert_one(doc)

        return {
            "message": "Plan confirmed successfully",
            "id": str(result.inserted_id)
        }

    except Exception as e:
        print("ERROR in /confirm:", traceback.format_exc())
        raise HTTPException(status_code=500, detail=f"Failed to confirm plan: {str(e)}")
```

GET ALL PLANS

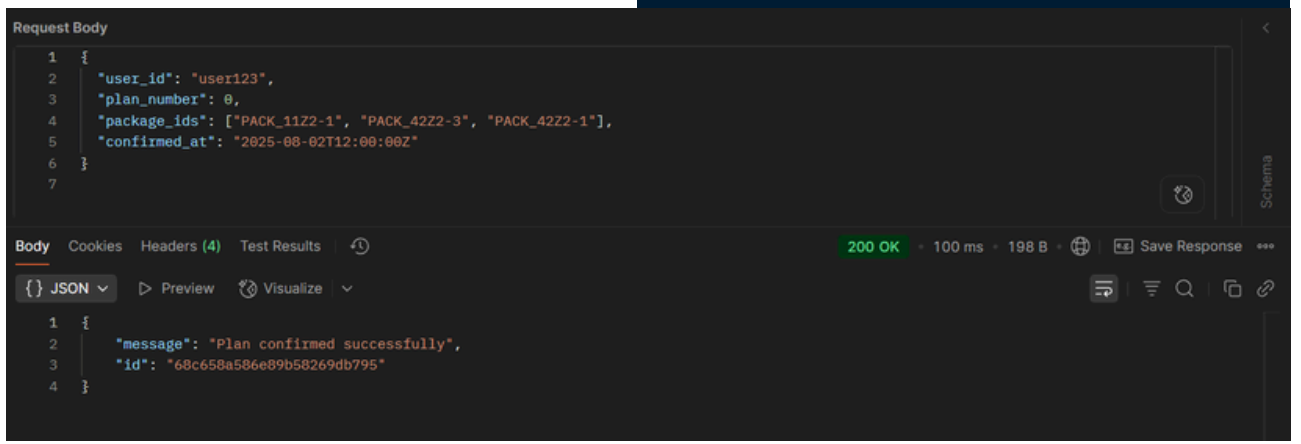
```python
# Get all confirmed plans
@app.get("/confirmed-plans", summary="Get all confirmed plans")
async def get_confirmed_plans(user_id: str = None):
    try:
        query = {}
        if user_id:
            query["user_id"] = user_id

        plans = list(confirmed_plans_collection.find(query))

        for p in plans:
            p["_id"] = str(p["_id"])
            if "confirmed_at" in p and hasattr(p["confirmed_at"], "isoformat"):
                p["confirmed_at"] = p["confirmed_at"].isoformat()

        return {"confirmed_plans": plans}

    except Exception as e:
        print("ERROR in /confirmed-plans:", traceback.format_exc())
        raise HTTPException(status_code=500, detail=f"Failed to fetch confirmed plans: {str(e)}")
```

POSTMAN API TEST

BUDGET PREDICTION POST CALL



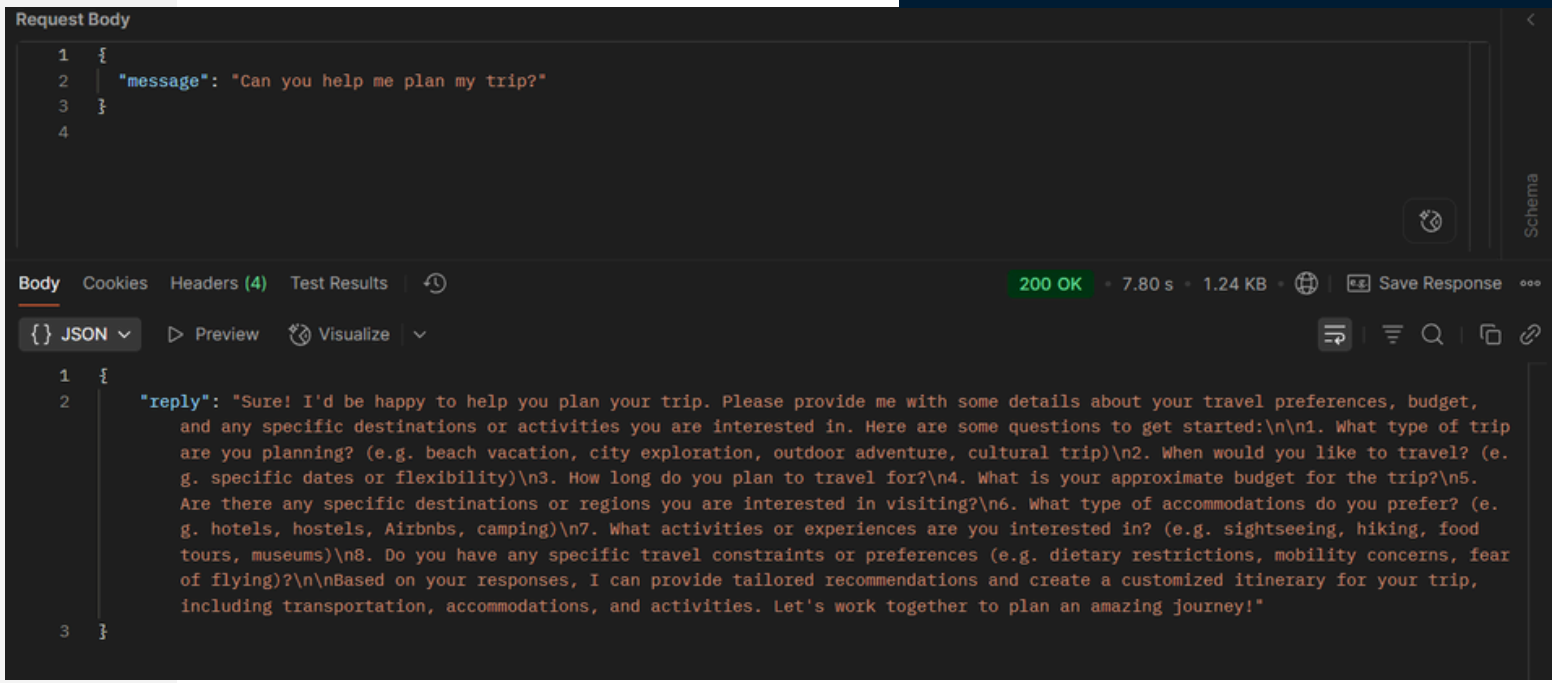
CONFIRMED PLAN POST CALL



POSTMAN API TEST

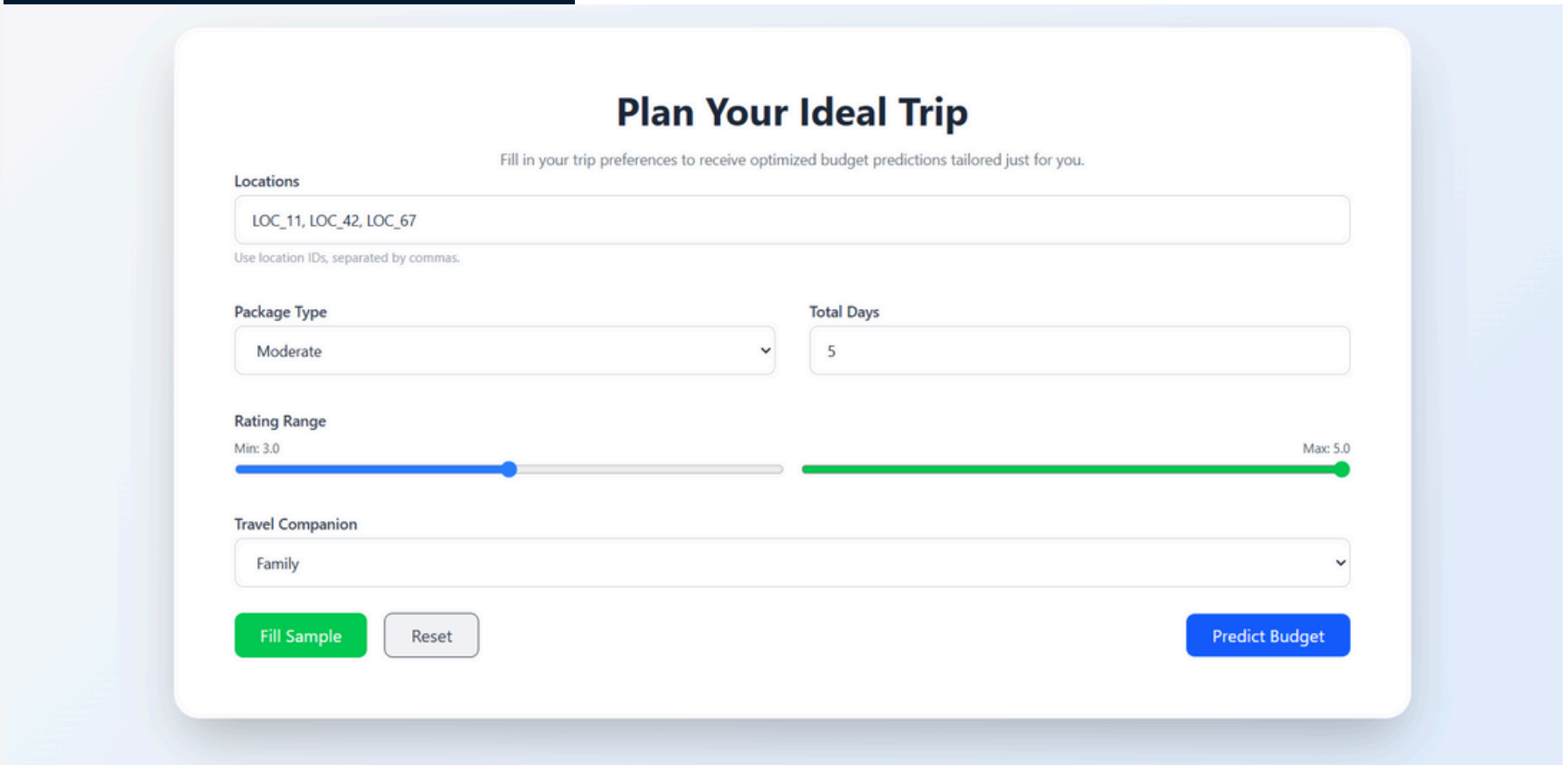
BUDGET PREDICTION GET CALL

CHATBOT POST CALL



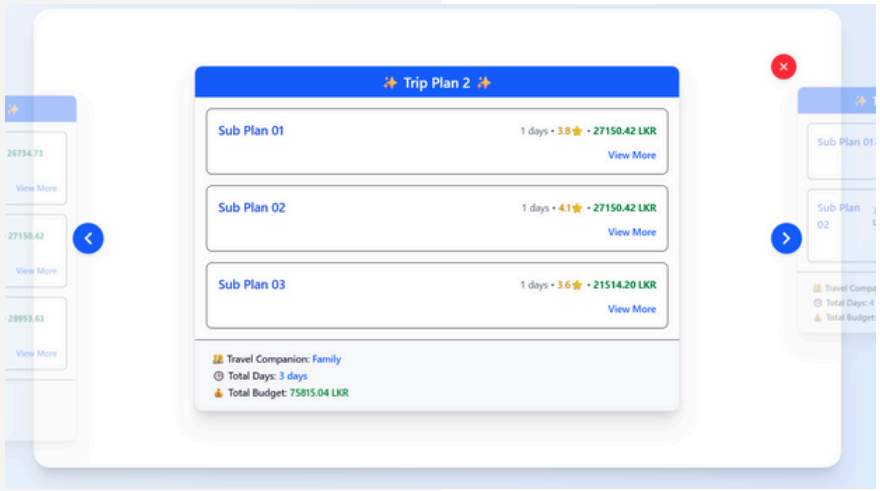
WORKING BUDGET PREDICTION

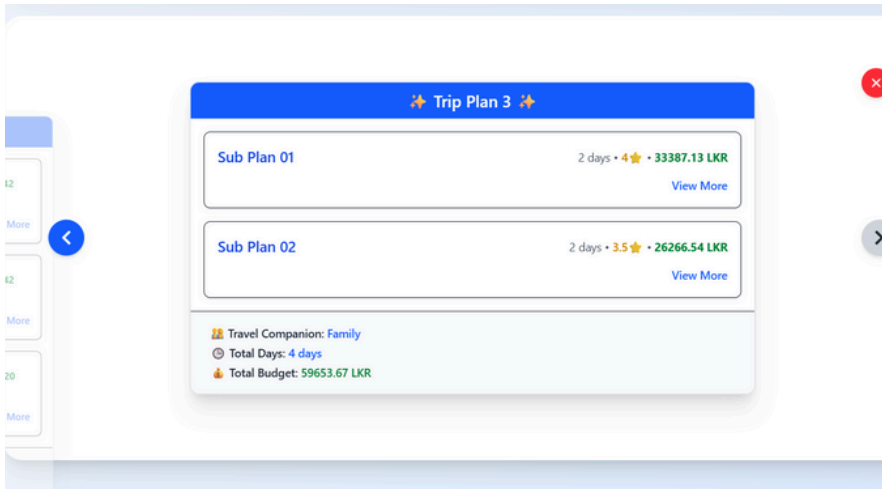
USER INTERACTION PAGE

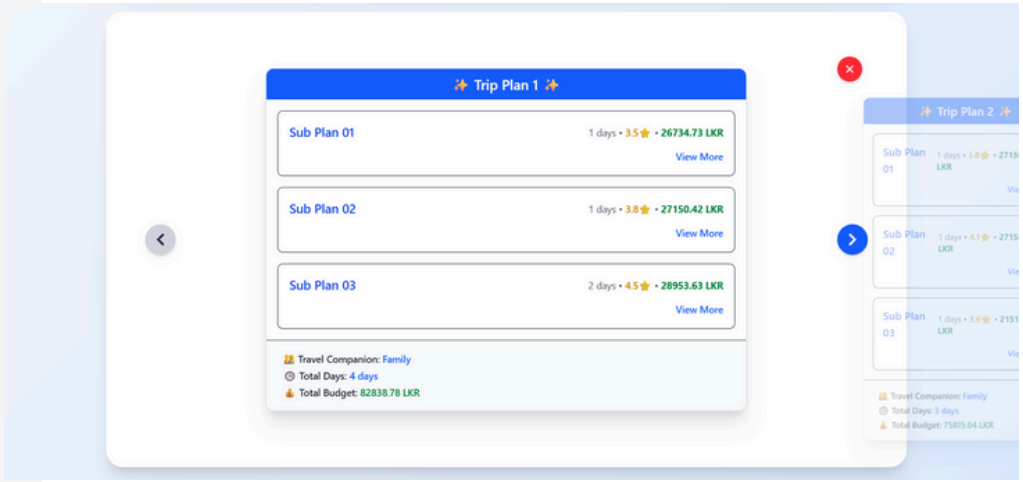


WORKING BUDGET PREDICTION

ITERATIVE BUDGET PLANS

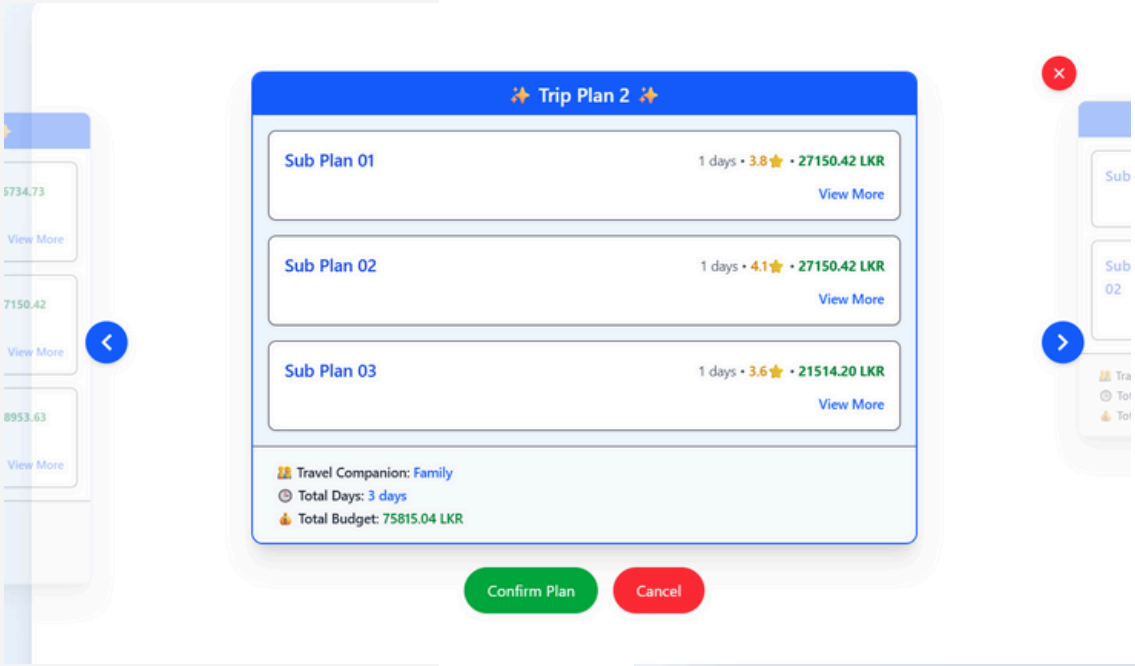


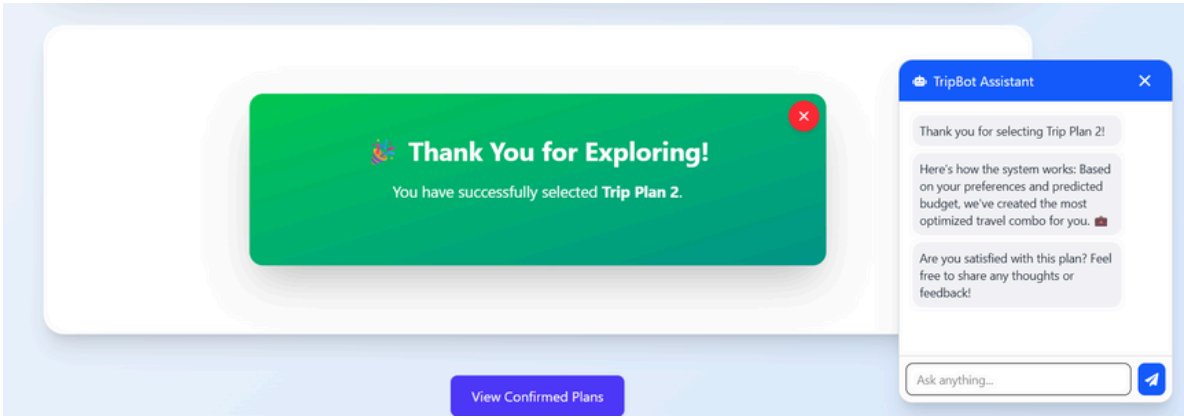




WORKING BUDGET PREDICTION

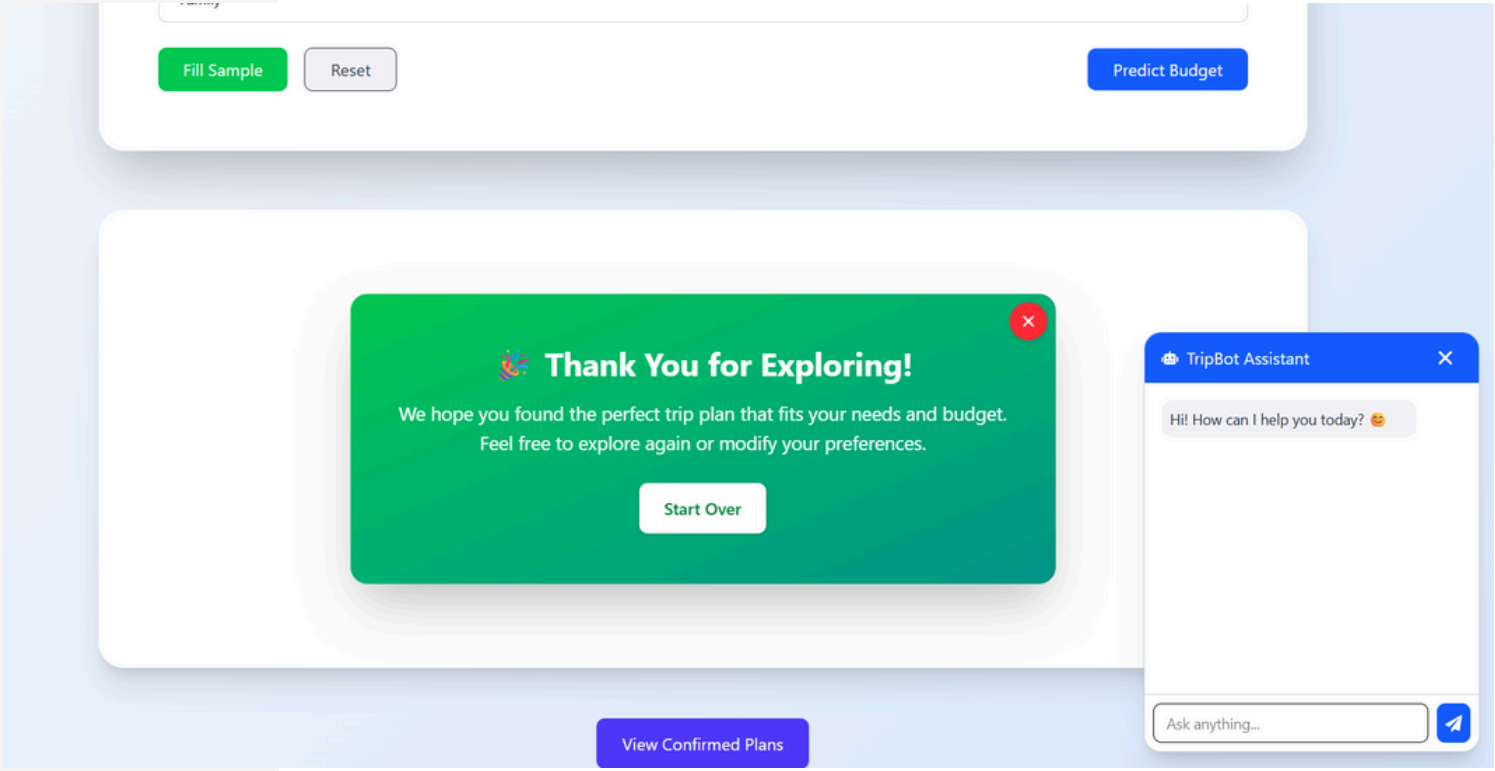
CONFIRMED PLAN & CHATBOT VIEW





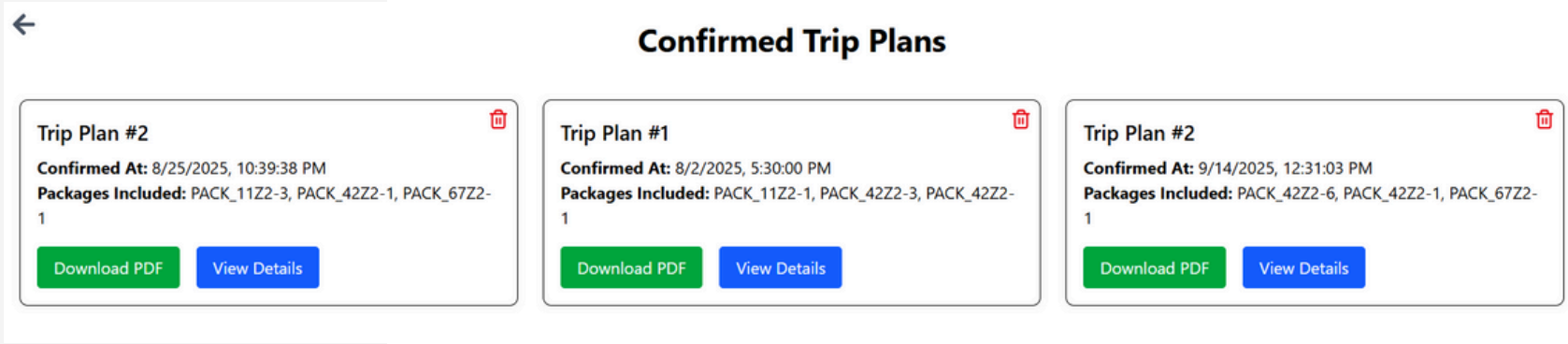
WORKING BUDGET PREDICTION

CANCELLED PLAN & CHATBOT VIEW

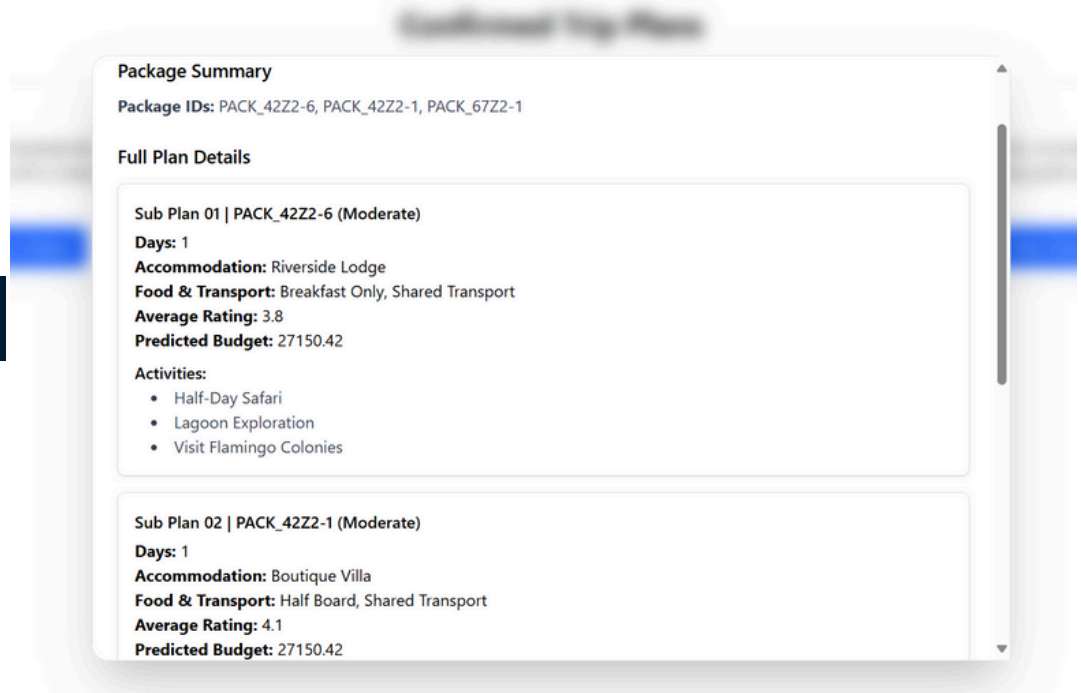


WORKING BUDGET PREDICTION

CONFIRMED PLANS SCREEN

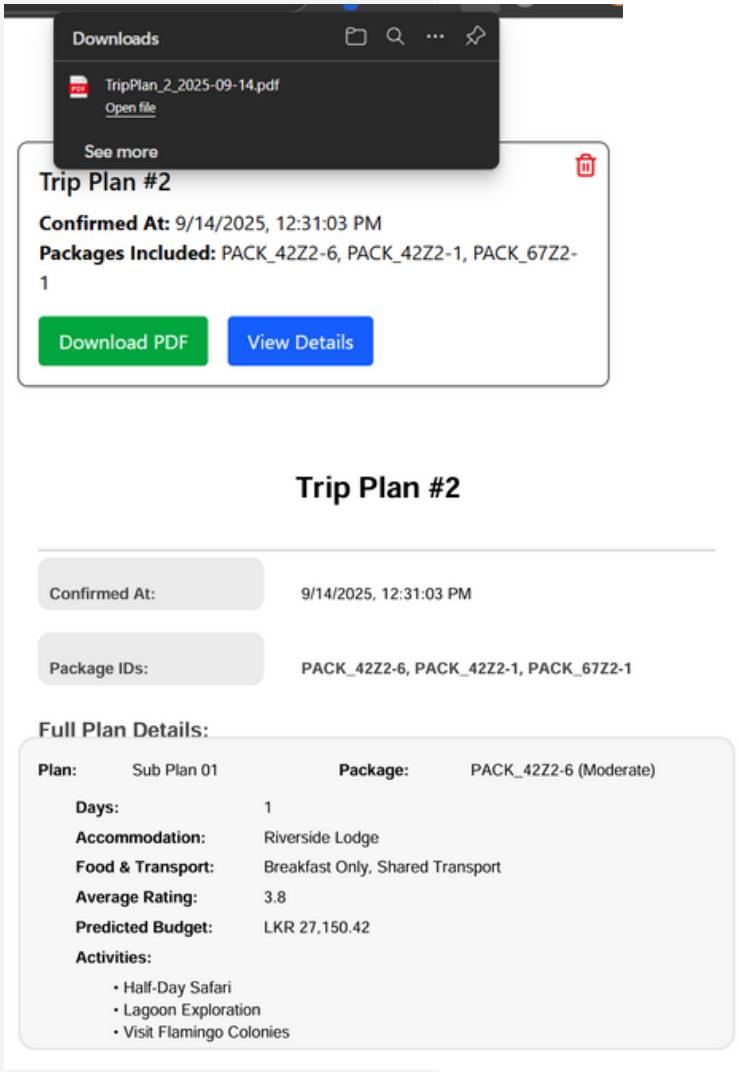


VIEW PLANS SCREEN

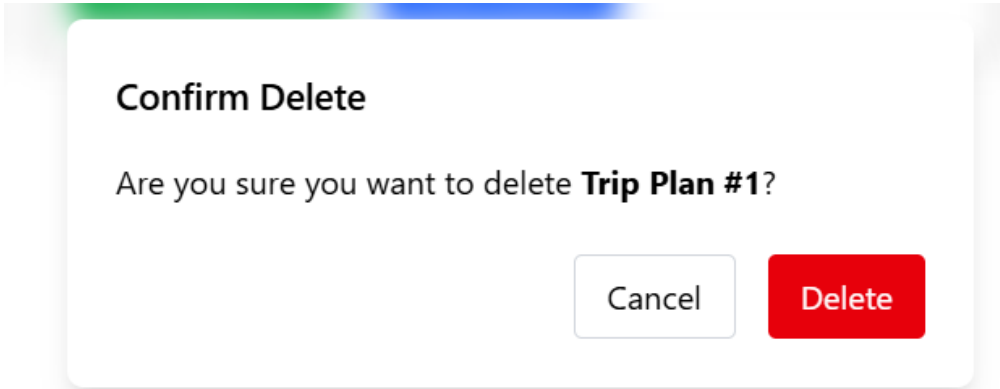


WORKING BUDGET PREDICTION

PDF VIEW OF SELECTED PLAN



DELETE SCREEN VIEW



REFERENCES

[1] J. Anderson, M. Smith, and L. Roberts, "Limitations in static budgeting models: Examining traditional travel budgeting tools," Journal of Travel Management, vol. 34, no. 2, pp. 45-62, 2020.

[2] H. Tan, C. Lee, and S. Tan, "Role of real-time data in budget predictions: Integrating APIs for travel budgeting systems," International Journal of Data Analytics, vol. 19, no. 1, pp. 30-42, 2021.

[3] X. Zhang, L. Wang, and Q. Chen, "Importance of user-centric designs in travel budgeting tools," Journal of User Experience Design, vol. 12, no. 4, pp. 58-75, 2022.

[4] J. Smith and K. Thomas, "Machine learning for travel budgeting: Enhancing cost predictions with predictive algorithms," Journal of Artificial Intelligence in Tourism, vol. 16, no. 3, pp. 215-228, 2021.

[5] M. Green and D. Turner, "Real-time data integration: A new frontier in travel budgeting systems," Journal of Travel Technology, vol. 21, no. 1, pp. 50-65, 2020.

THANK YOU