SMART TOUR: AN AI-DRIVEN MULTILINGUAL CONTEXT-AWARE TOURIST GUIDE PROGRESSIVE WEB APPLICATION

BY

IBEKWE KOSIOCHUKWU HASEL
MATRIC NUMBER: GOU/U22/CSC/788

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY,

GODFREY OKOYE UNIVERSITY, ENUGU STATE IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.SC.) DEGREE IN COMPUTER SCIENCE

SUPERVISOR:
ENGR. DR. KINGSLEY I. CHIBUEZE

JUNE 2026

APPROVAL

This is to certify that this project work titled **SMART TOUR: AN AI-DRIVEN MULTILINGUAL CONTEXT-AWARE TOURIST GUIDE PROGRESSIVE WEB APPLICATION** written by **IBEKWE KOSIOCHUKWU HASEL** has been read and approved as meeting the requirements for the award of the degree of Bachelor of Science (B.Sc.) in Computer Science, Department of Computer Science, Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu State.

Engr. Dr. Kingsley I. Chibueze	Date
Project Supervisor	
Dr. Frank U. Okebanama	Date
Head of Department	
Prof. Ifeyinwa Ajah	Date
Dean Of Faulty	
	Date
External Examiner	

CERTIFICATION

I, **IBEKWE KOSIOCHUKWU HASEL**, with Matriculation Number **GOU/U22/CSC/788**, hereby certify that this project work titled **SMART TOUR: AN AI-DRIVEN MULTILINGUAL CONTEXT-AWARE TOURIST GUIDE PROGRESSIVE WEB APPLICATION** is an original work carried out by me under the supervision of **Engr. Dr. Kingsley I. Chibueze** in the Department of Computer Science, Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu. I also certify that this work has not been submitted to any other university or institution for the award of a degree or diploma.

_____ _____

IBEKWE KOSIOCHUKWU HASEL Date

(Student)

DEDICATION

This work is dedicated to God Almighty, the source of all wisdom and knowledge and to my family, whose love, prayers and support have sustained me throughout this journey.

ACKNOWLEDGEMENTS

All glory and honour belong to God Almighty, whose wisdom and strength through His grace enabled me to carry out this research.

Gratitude is sincerely expressed to the project supervisor, **Engr. Dr. Kingsley I. Chibueze**, for his counsel, patience and highly valued advice in carrying out this research.

Gratitude is expressed to the head of department, **Dr. Frank U. Okebanama**, and all lecturers from the Department of Computer Science, Godfrey Okoye University.

Deep gratitude is expressed to my loving father, **Mr. John Ibekwe** and members of the family and friends for their prayers and encouragement during this process. Their faith in me throughout the process was greatly appreciated.

Thanks also go to my friends **Michael** and **Chisom** for being great company throughout the process.

Lastly, I wish to thank all participants in the user testing of the Smart Tour application for their valuable time and feedback.

ABSTRACT

The tourism sector plays a critical role in economic development, yet tourists in developing regions such as Enugu State, Nigeria continue to face significant challenges in accessing personalised, context-aware digital tourism information. This study presents Smart Tour — an AI-driven, multilingual, context-aware tourist guide Progressive Web Application developed to address this gap. The system was built on a full-stack JavaScript architecture comprising React as the frontend PWA, Node.js with Express as the RESTful API backend, and Supabase-managed PostgreSQL as the relational database. A hybrid AI recommendation engine was implemented, combining knowledge-based pre-filtering, content-based cosine similarity ranking, and collaborative filtering enhancement. The system supports seven languages via i18next with automatic browser language detection, renders an interactive GPS-linked map using React-Leaflet and OpenStreetMap, and is deployed as an installable PWA with offline capability through a Workbox service worker. Developed using Agile Scrum methodology across eight two-week sprints, all twenty functional test cases passed. Non-functional testing confirmed an initial load time of 2.1 seconds, a recommendation API response time of 1.3 seconds, and a Lighthouse PWA score of 92 out of 100, demonstrating that a handcrafted hybrid scoring model is a viable and interpretable alternative to machine learning approaches for tourism recommendation in resource-constrained emerging market contexts.

Keywords: Smart Tourism, Progressive Web Application, Artificial Intelligence, Recommendation System, GPS, Multilingual Support, Agile Scrum, Enugu State, Nigeria.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

Tourism is among the rapidly growing sectors of the world economy, contributing greatly to the economy of a country, cultural exchange and connectivity throughout the globe. As stated by the United Nations World Tourism Organization (UN Tourism, 2024), there were 1.4 billion tourists in 2018 worldwide. The growth has been consistent in the sector, making it an important economic component for many countries. In Nigeria, tourism helps in creating job opportunities, earning foreign exchange and promoting culture through a wide array of activities, from places like Enugu State, which includes Ngwo Pine Forest, Awhum Waterfall and Opi Lake.

The evolution of ICT has had a profound impact on how travellers plan, experience and remember their travels. As noted by Sarkar et al. (2023), ICTs have allowed for the collection, dissemination and retrieval of information concerning tourists' travel destinations, travel routes and other places of interest through the use of internet sites. In recent years, there has been an increased reliance on information technologies by tourists who seek to gain knowledge about their destinations prior to travelling to them.

However, despite all the technological developments that have taken place, the role of conventional tour guides and website systems still lacks the necessary dynamism, personalisation and relevance needed by today's tourists. Information overload is likely to occur when the system does not personalise according to the location, time, language and individual needs of the visitor. Furthermore, most online tourism websites currently demand the installation of a native app, a constant internet connection and a specific language option – features that pose significant challenges, especially in developing countries.

These are some of the reasons why smart tourism systems have come up. They incorporate technologies like the Global Positioning System (GPS), artificial intelligence (AI), geographic information systems (GIS) and progressive web applications (PWAs). The result is that they provide real-time, location-based, and personalised tourism assistance from multiple devices without needing native installations. Still, despite the potential offered by these technologies, incorporating them into a single tourist guide system has not been fully realised. This is especially so when considering a tourist guide system for developing nations such as Enugu, Nigeria.

This research aims to fill the void through the design and development of Smart Tour, an artificial intelligence-driven multilingual context-aware tourist guide progressive web application which integrates location-based services, a hybrid artificial intelligence recommendation engine, multilingualism and PWA accessibility.

1.2 Statement of the Problem

Today's tourists rely heavily on digital tools to provide them with all the necessary information on travelling destinations and accommodation arrangements, modes of transport and leisure activities. Yet, a close scrutiny of current tourist guide software has shown that there are various issues inherent to them.

First, many currently available tourist guide apps provide non-personalised content which is static in nature. This lack of personalisation often causes tourists – especially new visitors – to be bombarded with too much information. As a result, tourists find it challenging to pinpoint locations that suit their personal tastes and fit within their travel budgets.

In addition, communication barriers pose one of the major challenges faced by tourists coming from other countries. The current websites used in the promotion of tourism in the country are not very supportive because they lack sufficient translations. As a result, tourists visiting Nigeria find it difficult to comprehend various information on the platforms.

Third, today's travel applications are still relying on a primitive level of context awareness, mostly based on simple location-based searches but not on temporal information or past behaviour preferences of users. Therefore, tourists miss out on suitable tourist spots according to their travel patterns and present situations.

Fourth, existing systems often do not have complex AI-based recommendation engines that can analyse the preferences, browsing history, reviews, and popularity trends to provide personalised and valuable recommendations. Without such intelligent systems, the tourists will not be able to make rational decisions about their travel choices.

Finally, many tourist applications require a native installation on an iOS or Android phone, need to be connected to the Internet all the time and take up large amounts of space in the phone memory, issues that restrict the applications from being used by most people using lower-grade phones, a majority of which are found in Nigeria.

All these constraints indicate the need for an integrated, intelligent tourist guide system based on the web, which will combine the use of AI in personalisation, GPS for context-awareness, multi-language support and PWAs in one easily accessible platform suitable for tourists visiting Enugu and similar cities.

1.3 Aim and Objectives of the Study

The aim of the study titled Smart Tour is to design and develop an AI-driven, multilingual, context-aware tourist guide Progressive Web Application that provide personalised, location-based information and recommendations to improve tourists' experience during their travels.

The specific objectives of the study are to:

1. Develop and deploy a PWA-based tourist guide system that offers cross-platform usability, offline browsing functionality, improved speed performance and a great user experience without the need for downloading any native app on the device.
2. Develop an advanced tourist destination app with integrated GPS-enabled location services and an interactive searching and filtering tool so that tourists can discover their desired points of interest and places in the surrounding area according to various filters.
3. Develop a multilingual support feature that allows users to select languages automatically by detecting their device's language setting and manually from a selection of English, French, German, Spanish and Italian.
4. Create an AI-based recommendation engine that creates personalised recommendations for travel destinations, taking into account users' preferences, visited locations before, rating popularity, and other parameters such as location and time.
5. Creates an intuitive user interface design that is easy to use and accessible from both mobile phones and computers.

1.4 Significance of the Study

The study offers significant contributions to several stakeholders involved, including those to the overall knowledge base of smart tourism and intelligent web applications.

From the tourists' perspective, the Smart Tour system allows them to use the intelligent system that offers tourists relevant and timely information on available locations, accommodations, dining places and recreation centres. Using AI, the intelligent system filters through several

parameters and saves tourists a lot of time and effort by suggesting places based on individual needs and history of visits. By offering a multilingual interface, the Smart Tour system ensures that tourists can easily communicate and engage with the system even if they cannot understand English.

The tourism industry in Enugu, Nigeria, will find the system useful for its digital marketing purposes. The system will assist in recommending places to visit, hotels, restaurants and other forms of entertainment by providing ratings and popular trends. This will help to promote tourism to underdeveloped places where the system could recommend tourists to visit.

Academic and research-wise, this study contributes to the literature by increasing the understanding of how context-aware computing, hybrid recommender systems, multi-language NLP capabilities and PWAs can be effectively combined into one tourist application. The study presents a sample architecture and decisions which can be used in further development of smart location-based applications by researchers.

In terms of the developers and system architects, the project presents a viable framework for designing an AI-based multilingual context-aware web application. The system design, recommendation algorithms and approach to implementing the PWA as presented in this paper provide a valuable template for building a tourist system that is scalable and user-centric in other emerging economies.

1.5 Scope of the Study

The research work is centred on the design, development, and evaluation of Smart Tour: An AI-Driven Multilingual Context-Aware Tourist Guide Progressive Web Application. In terms of location, the proposed system aims at tourists who visit Enugu State, Nigeria, where the database of the tourist destinations includes all locations such as hotels, restaurants, and leisure centres in Enugu State, Nigeria. Nevertheless, the system can be used in different geographical environments.

In terms of functionality, the scope includes GPS-powered detection of location along with the display of points of interest around that location on an interactive map generated through Leaflet.js. Search and filter modules enable searching and filtering according to categories, budget, ratings, and distance. The AI recommendation system utilises the hybrid approach to recommendation generation through content-based, collaborative, and knowledge-based filtering techniques.

The multilingual module supports five languages: English, French, German, Spanish and Italian and is applied for text in application interfaces, location names and location descriptions. Automated translation of review content by users is provided with known limitations in the translation quality related to idiomatic and culturally orientated expressions. User account capabilities include managing favourite locations, keeping track of visited locations, posting reviews and ratings and managing language and notification settings.

The study is carried out using the web-based Progressive Web Application and does not encompass any development on either the native iOS or Android applications. The following functionalities are out of the scope of the project: booking/reservation services, payment gateway integration, customer live support/live chat and advanced offline maps caching other than the service worker cache offered by the PWA itself.

1.6 Limitations of the Study

There are several restrictions on the scope of this research project, and these are highlighted below to give an honest view of the scope and applicability of the proposed system.

Firstly, the Smart Tour solution requires an active internet connection for generating maps, accessing live GPS information, handling recommendation algorithms using AI and translating languages. Individuals who are in regions that lack network connectivity, which is quite common in rural areas in Enugu State, will have their performance impacted negatively, or some functions will be inaccessible. However, although the Progressive Web App solution can cache previous data using the service worker, offline functionality cannot be achieved.

Secondly, the precision of location services using GPS depends on the condition of the hardware of the user's device and the strength of satellite signals. It may not work well in congested urban areas with high buildings and other obstacles or when indoors and underground, where there may be low accuracy in detecting the location and hence poor recommendations of nearby options.

Thirdly, the feature of multilingual translation depends on machine translation technologies and does not guarantee perfection in linguistic translation, especially user-generated content like reviews, where there can be many cultural and linguistic nuances that the machine will not be able to translate properly. In case of accurate and reliable translations of a multilingual version of the product, professional human translation will be necessary.

Fourth, the performance of the recommendation algorithm depends directly on the quantity and quality of data regarding user interactions. New users who have few interactions will not be able to benefit as much from personalised recommendations compared to other established users. However, the cold start problem can be alleviated partly by the knowledge-based filter module, which makes recommendations based on explicit user statements about their interests, rather than behavioural history.

Finally, the research only considers a web application version, which means that some hardware capabilities of the mobile device itself are not available to the application, including the capability to track location even when running in the background and Bluetooth capability for close proximity tracking.

1.7 Definition of Terms

The following key terms are used throughout this study and are defined here for clarity:

Artificial Intelligence (AI): The capacity of computer-based systems to perform activities which normally need human intelligence, such as reasoning, learning, decision-making and solving problems. The artificial intelligence referred to in this research relates to the recommendation engine analysing user data to provide recommendations for tourists.

Progressive Web Application (PWA): It refers to a website application constructed by utilising common web technologies such as HTML, CSS, and JavaScript, which has been enhanced further by integrating functionalities such as working offline, receiving push notifications and adding an icon on the home screen, facilitated by service workers and web app manifests.

Geographic Information System (GIS): A structure that facilitates the acquisition, storage, analysis, management and presentation of data related to spatial or geographic characteristics. In this case, GIS will be used in displaying the tourism destination, route and points of interest in a map.

Global Positioning System (GPS): This refers to a navigation system based on satellites that can offer live geographic coordinate information regarding the position of the device. In this research, GPS technology is used to establish the location of the tourist.

Recommendation System: The system uses AI to predict and recommend objects or services that are related to users' preferences and needs. The recommendation system used in this

research recommends touristic places based on the analysis of users' preferences, visits, ratings and other contexts.

Context-Aware Computing: A computing approach where a system is able to sense the user's environment, which may include location, time, activities and user preferences, and use this information to provide more appropriate output. In this research, context awareness helps the system make recommendations according to the location and time of day for the tourist.

Multilingual Support: Refers to the ability of a software system to present its user interface, information and other facilities in several languages. In this research, the multilingual feature allows tourists who speak different languages to communicate with the Smart Tour software in their language of choice.

Hybrid Filtering: It is a technique whereby two or more filtering techniques, such as content-based filtering, collaborative filtering and knowledge-based filtering, are integrated to provide better recommendations compared to when each technique is used individually.

Smart Tourism: The use of ICT technologies such as AI, Internet of Things (IoT), mobile computing and Geographic Information Systems (GIS) for increasing the effectiveness, sustainability, personalisation and quality of tourism experiences for tourists and destination managers.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

A critical analysis of current literature related to the design and development of the Smart Tour system will be presented in this chapter. Seven different themes have been considered for the purpose, which include smart tourism, tourist guide system, artificial intelligence in tourism, recommendation systems and filtering techniques, multilingual tourist systems, context awareness computing and Geographic Information System (GIS)/Global Positioning System (GPS). Moreover, the use of Progressive Web Applications as the technology in which the software application will be developed and deployed will also be critically reviewed. This chapter ends with a summary of the research gaps found in the current literature.

2.2 Smart Tourism

Smart Tourism can be defined as an approach where digital technologies are applied in tourism practices and services in order to enhance the overall experience for the travelers. Smart tourism is a technology-based concept which aims at making the practice of tourism efficient, convenient, personalized and interactive. According to Koo et al. (2025), smart tourism is a result of the integration of ICT and the tourism industry. In addition, Koo et al. (2025) argue that smart tourism is a conceptual framework that involves the gathering and exploitation of information from tourism industries, social media platforms and clients through the use of advanced technologies.

The technologies that make up smart tourism include mobility, GIS, cloud computing, artificial intelligence, IoT and other online platforms. The technologies come into play in order to support tourists even before they embark on the journey or after it. The activities that tourists engage in using these tools include searching for destinations, planning travel, managing itineraries and navigation in real time. For example, a tourist visiting Enugu can be able to use smart tourism to get information about places to visit, prices, reviews and even directions.

Although this is the case, several issues with the adoption of smart tourism have been brought to attention. According to Wang et al., most systems focus on just one aspect of the tourists' experience, such as making arrangements for the hotel or planning the itinerary. In addition, Buhalis and Amaranggana state that smart tourism should include the real-time gathering of information coming from different resources, which cannot be achieved in many existing

platforms. This problem will be overcome through the creation of a platform that includes all of the mentioned features at once.

2.3 Tourist Guide Systems

The tourist guide systems refer to information technology platforms through which information is conveyed to tourists regarding destinations, accommodation facilities, restaurants, travel routes and emergency services. The development of tourist guide systems is characterized by the development of digital technology. These have evolved from printed touristic brochures and maps through the introduction of online websites to AI-powered touristic websites and mobile phone applications (Sarkar et al., 2023).

The early computer-based tourist information guide systems, according to the study conducted by Abbasi-Moud et al. (2022), were indeed revolutionary for their times, although they were plagued by several constraints, the most prominent one being their inability to adjust to changing contexts or user needs. The content was presented to each user irrespective of their location, interests or even the language spoken, thus making the information irrelevant to them.

Current tour guide apps have come a long way from their predecessors. Today, they include artificial intelligence, GPS tracking, GIS maps and multi-language interfaces. But a detailed analysis of existing platforms, including TripAdvisor and Google Maps, demonstrates that although they do well in specific areas such as crowd-sourced feedback and navigation, there is no tour guide app that combines all required components into one functional tool (Sarkar et al., 2023). Most need to be installed natively, making them inaccessible for users with limited storage space.

This is exactly what the Smart Tour system proposes to solve. The system is a web-based, PWA-supported tour guide that runs without any need for installation, supports multilingualism, makes use of GPS for real-time location services and provides AI-powered personalized recommendations, functions that none of the present systems have managed to combine into one application dedicated to tourists visiting places like Enugu, Nigeria.

2.4 Artificial Intelligence in Tourism

Artificial intelligence (AI) can be defined as the ability of computers to do things which would normally require human thought processes such as reasoning, decision-making, problem-solving and learning. Within the context of tourism, artificial intelligence has been used in

diverse applications ranging from smart recommendation systems and virtual assistance using chatbots to destination recognition through image analysis (Nicolescu & Tudorache, 2022).

Recommendation systems stand out as one of the most successful uses of AI in tourism, providing personalized recommendations to users through an analysis of their preferences and behaviour. This concept is well explained by Sarkar et al. (2023), who provide a detailed taxonomy of different methods for tourist recommendations.

Chatbots powered by artificial intelligence and virtual assistants are other inventions that are noteworthy. Booking.com and Expedia, for instance, have utilized chatbots that can deal with enquiries about bookings, answer common questions and provide travel guidance instantly (Buhalis & Cheng, 2020). However, there are shortcomings in such inventions as well, since these systems can communicate using text only and can have trouble understanding complex questions.

Machine learning approaches, especially collaborative filtering and neural networks, are gradually being used to predict the behaviour of users over time, thus allowing the system to learn from their interactions (Sarkar et al., 2023). Nonetheless, according to Lim et al. (2020), machine learning algorithms need significant amounts of training data before they can work effectively, thus posing a limitation on sites with few users, something that is avoided by the Smart Tour system using knowledge-based filtering for new users.

AI is considered to be the crucial component of the Smart Tour software in this case. It will make the hybrid recommender function, enable time- and GPS-based recommendation and facilitate multilingual communication. All these aspects are intended to address the shortcomings of previous single-purpose tourism AI systems.

2.5 Recommendation Systems

Recommendation systems refer to technologies which use artificial intelligence to predict and provide recommendations to users on appropriate products and services based on their needs and preferences. They were first invented in an e-commerce setting, being used by such big companies as Amazon and Netflix. Nowadays, recommendation systems are employed in various industries, among which tourism, medicine and education are mentioned by Sarkar et al. (2023). The main problem associated with tourism is related to information overload, whereby there are hundreds or even thousands of options available to tourists.

In the tourism sector, recommendation systems offer tourists suggestions about proper destinations, attractions, accommodation and services based on their analysis of information such as user interests, previous interactions, rating submissions, spatial context and timing. For instance, an Enugu tourist who has been browsing for natural attractions may receive recommendations to visit the Ngwo Pine Forest or Awhum Waterfall. According to Sarkar et al. (2023) and Sarkar et al. (2023), there are mainly four kinds of tourism recommendation systems: content-based filtering (CB), collaborative filtering (CF), knowledge-based filtering (KBF), and hybrid filtering.

Each filter has its advantages and disadvantages, which is why today's academic research trends favour combining different filters in order to leverage their advantages and avoid their disadvantages (Sarkar et al., 2023). The next subsections will analyse every filtering method in more detail as related to smart tourism.

2.5.1 Filtering Mechanisms in Recommendation Systems

Filtering methods involve the use of algorithms for determining what products will be recommended to users based on their behaviour, interests and other data. In tourism applications, efficient filtering becomes essential since the number of objects that can be chosen from is huge and the tourists have a restricted amount of time and attention. Filters rely on an analysis of the user's behaviour and preferences and select the best items from the entire set available to the system.

2.5.1.1 Content-Based Filtering

In content-based filtering (CB), the system suggests products or services to a user on the basis of the features of those products or services which the user has interacted with before. CB involves the analysis of the features or aspects of the products or services, such as category, kind of place, cost, ease of access and keywords that match the profile created through user interaction (Sarkar et al., 2023). For instance, if a tourist repeatedly looks for or visits nature-related destinations like waterfalls, caves and forest reserves, then it can be deduced that the user is interested in nature-based tourism.

The strength of content-based recommendation lies in the fact that it does not depend on the actions of other users; recommendations are derived only based on the past experiences of the single user and the features of the items, which makes it more reliable in scenarios where there

are few users. This is especially applicable to new or upcoming tourism sites like Smart Tour, which may still lack substantial multi-user data.

Nonetheless, CB filtering is known to have limitations, and one of the main challenges that have been identified is the cold-start problem, where when a new user comes into the system without any interaction history, there will not be enough information for the algorithm to make any recommendations (Sarkar et al., 2023). Another challenge associated with CB filtering is its tendency to over-specialise, recommending similar items to the ones the users interact with, limiting their choices. For instance, a tourist who uses the Smart Tour application might fail to discover other kinds of attraction sites.

2.5.1.2 Collaborative Filtering

Collaborative filtering involves the use of recommendation strategies drawn from the actions of many users. CF assumes that individuals who had similarities in their behaviour in the past will maintain such similarities in their behaviour into the future. It involves finding other users who act in a similar way as the focal user and recommending products that such similar users have been involved with, but the focal user hasn't come across before (Sarkar et al., 2023). In smart tourism, for instance, if tourists who have similar tastes were frequently visiting certain museums or historical monuments in Enugu, then such venues would be recommended to other users.

One of the most important strengths of collaborative filtering is that it helps in discovery. Users discover products that they would not have discovered had they used searching methods. Another thing about collaborative filtering is that it does not need item metadata like content-based filtering.

However, CF faces two common limitations. First, there is the issue of sparsity, whereby the user-item interaction matrix becomes largely filled with blanks due to the users' limited interactions with the total set of items, thus posing problems in determining reliable similarity relationships (Sarkar et al., 2023). Second, the increasing number of users and items causes an increase in computing costs involved in carrying out similarity measures, which may impact negatively on the performance of real-time recommendation systems. Another limitation is that CF requires enough activity from the users to be effective, limiting its usefulness at the beginning of the implementation of the recommender platform. In the Smart Tour recommender system, these limitations have been addressed by adopting a hybrid system, combining CF with other approaches such as content-based and knowledge-based systems.

2.5.1.3 Knowledge-Based Filtering

KBF creates recommendations based on the matching between explicit requirements of the user, for example, budget constraints; desired category; distance; accessibility preferences; and purpose of visit and a knowledge database that holds detailed information regarding each possible item (Sarkar et al., 2023). In contrast to CBF and CF, KBF recommendation is independent of any user activity data and depends instead on preference information supplied by the user directly.

KBF relies on logical rules that can be stated using IF-THEN expressions. For example, in a smart tourism application, such rules would include:

- IF budget is low AND distance is less than 3 km THEN suggest cheap attractions close by.
- IF interest is cultural AND time is morning THEN suggest museums and historical sites.
- IF preference is relaxation, THEN suggest parks, resorts, and gardens.

One advanced form of user preference modelling includes ontology-based user profiling, which was examined by Sarkar et al. (2023). Ontology-based user profiling entails using a semantic domain ontology containing semantic connections among different concepts to express user preferences. For example, an appreciation for the word "adventure" might have a semantic connection with words like "hiking", "rock climbing", "cave exploration" and "white water rafting".

The most important advantage of using the KBF approach is its independence from historical data, which enables it to function effectively with new users – solving the problems inherent in both collaborative and content-based recommendation techniques. Nevertheless, the KBF technique has certain drawbacks: it requires a manual creation of the knowledge base, no autonomous learning, and possibly limited scalability due to the increasing number of rules and items (Sarkar et al., 2023). As regards Smart Tour, the main areas of application of the KBF approach include new users and contextual constraints related to user-defined restrictions like price and distance.

2.5.1.4 Hybrid Filtering

Hybrid filtering involves using combinations of recommender systems techniques with an aim of addressing the limitations of each of these techniques while taking advantage of their strengths (Sarkar et al., 2023). Hybrid filtering has been widely embraced in tourism studies due to the fact that there is no one filtering technique that can handle all situations effectively (Sarkar et al., 2023).

Hybrid methods can be created in different ways by combining two or more recommendation techniques; for instance, by switching between recommendation techniques based on the situation (such as using KBF when users are new and CF when users are known). Other hybrid techniques include combining the results of several recommendation systems, or running the recommendations sequentially, with the result from one recommendation used as the input for the next recommendation system. An example of such a hybrid technique would be the creation of Smart Tours recommendations by eliminating all destinations out of the user's budget through KBF, followed by ranking destinations using CBF and finally generating recommendations using CF.

The main drawback of hybrid filtering lies in its complexity: coordinating the functioning of several models and balancing the weights of each model can be problematic. Moreover, hybrid filtering is computationally expensive since several algorithms run simultaneously (Sarkar et al., 2023). It also presents problems for the management of data, since data from several sources may conflict within the process of applying several recommendation techniques at once. Nevertheless, most research agrees that hybrid filtering performs better in complex tourism scenarios. This is why the authors chose to design a hybrid recommender in their Smart Tour application.

2.6 Multilingual Tourist Applications

A multilingual tourism application is a system that can provide information and user interface in more than one language, allowing tourists from different linguistic groups to communicate with the software application in the language they understand. The significance of multilingualism in tourism cannot be understated because language barriers are among the most common reasons cited for a bad experience by tourists, especially those visiting a foreign country (Abbasi-Moud et al., 2022).

Modern multilingual tourism apps generally have features like language options, location detection from either device or GPS data and immediate translation of various types of content such as attraction information, review comments and directions. According to Dunlop et al.,

multilingual capabilities increase the credibility of users and enhance the use of apps, especially among non-English speaking tourists.

It is important to highlight the difference between translation and localisation. The former refers to the transformation of the text into another language. In contrast, the latter pertains to adapting the content not only linguistically but also culturally, socially and environmentally. This includes formatting dates, presenting currencies, choosing the correct units of measurement and selecting the proper mode of communication (Abbasi-Moud et al., 2022).

Nevertheless, a major drawback of automated translation services provided by current tourism websites involves the lack of accuracy in translating user-produced content such as reviews, in which idioms, culture-specific concepts and local dialects cannot be accurately translated by machines (Abbasi-Moud et al., 2022). Moreover, most tourist applications cater to major international languages like English, Spanish, French, German and Mandarin only, and not to developing tourist regions like Africa, Southeast Asia and the Middle East.

The Smart Tour system supports multiple languages such as English, French, German, Spanish and Italian based on the major languages spoken by tourists who visit West Africa. Language auto-detection is supported by using the device's location-based settings as well as GPS location information, and the system allows users to manually override the automatic detection. The system supports basic translation for user-generated comments as well as for the description of each tour, recognising that automated translation can have some limitations for certain complex information.

2.7 Context-Aware Computing

Context-aware computing implies the capacity of a computer system to be aware of contextual information about the user as well as his surroundings, thereby adjusting its behaviour for enhanced results in accordance with that context (Abbasi-Moud et al., 2022). Contextual information usually includes the location, time, user activity, social context, type of device and network information. With regard to the tourism industry, context-awareness enables static information provision to become dynamic as per circumstances, in the sense that tourists have very different requirements in terms of information when they arrive at a museum at 10 am versus a restaurant area at 6 pm.

Abbasi-Moud et al. (2022) note that the location parameter is the most critical context for mobile and tourism applications, noting that a significant percentage of user queries posed in

outdoor settings are location-based. Time-related context refers to the day of the week, time of day and season of the year, which affects the operating times, availability of activities, prices and appropriateness. The activity context of the user provides another dimension to the context that enables the application to know whether the user is searching for the next place to go to or has reached the place he wants.

A number of researchers have confirmed the potential benefits of context awareness in the tourism industry. For instance, Abbasi-Moud et al. (2022) proved that the provision of location-based content is highly relevant and useful compared to non-contextual systems. Additionally, Sarkar et al. (2023) discovered that travellers who used context-aware mobile tour guides experienced greater satisfaction levels and were more certain about their destinations than other tourists.

Nevertheless, these clear advantages are largely neglected in the present implementations of tourism applications that only incorporate very basic contextual information in the form of nearby search based on GPS technology without considering other aspects such as temporal, behavioural and social context (Sarkar et al., 2023). In fact, such a practice implies a huge loss of possibilities to make the tourists' experience much more enjoyable. One such example is the Smart Tour application, which incorporates the use of three different contexts into its recommendation system.

2.8 Geographic Information Systems (GIS) and GPS in Tourism

Geographic Information Systems (GIS) is an approach used in the capture, storage, analysis, and presentation of geographic data. In the field of tourism, GIS provides the means to represent tourist destinations, routes, and other geographic points of interest on electronic maps, which supply the tourists with geographically orientated information which could not have been delivered through mere textual description (Yoo et al., 2020). GPS, which stands for Global Positioning System, is used to provide coordinates regarding the location of the user.

The combination of GIS and GPS in tourism applications has been highly effective. According to Boori et al., a system for mapping tourists based on GIS is more efficient in guiding tourists than guide books. This technology permits tourists to identify their location against other tourist attractions nearby, measure either walking or driving distance to any point of interest and provides tourists with instructions for how to get there.

Mapping libraries available on the web, such as Leaflet.js and Google Maps APIs, have made the inclusion of GIS components accessible, allowing programmers to incorporate maps into their web apps without necessarily having extensive knowledge of geospatial technology (Yoo et al., 2020). Such mapping libraries provide for clustering of proximate points of interest, highlight areas through polygons, illustrate routes and customise markers, among others, all useful in smart tourism.

The recurring challenge associated with the use of GPS in tourism services is that GPS accuracy is negatively affected in regions with limited access to satellites due to obstructions from buildings, tunnels, and indoors; consequently, the collected location data may not always be accurate (Yoo et al., 2020). Furthermore, frequent GPS polling consumes a significant amount of battery power on the smartphone, an aspect that should be taken into account when using GPS services in the field. The Smart Tour system utilises GPS for detecting location information and updates while relying on Leaflet.js for mapping purposes.

2.9 Progressive Web Applications (PWA) in Tourism

The concept of Progressive Web Apps (PWAs) can be described as a contemporary web programming approach that unites the openness and ubiquity of web browsers with the power of the native apps on mobile devices, such as offline access, push messages, and background data synchronisation (Huber et al., 2022). PWAs are developed using common programming languages and frameworks for web sites: HTML, CSS and JavaScript.

There is great potential of utilising PWAs within the field of tourism. Visitors to different destinations often face issues relating to poor cellular coverage in remote locations, costly roaming fees when visiting another country and places that have no network reception. These are some of the problems faced by web applications dependent on constant internet availability. Offline capabilities offered by PWAs through the service worker allow access to information even when there is no internet connection (Google Developers, 2020).

In a comparative analysis by Huber et al. (2022), PWAs emerged as having a better balance between factors such as wide reachability, ease of installation and cross-platform functionality, all of which are highly pertinent to tourist systems. Tourists will not be prepared to download native applications for destinations that they only spend limited time in; the ease of downloading associated with PWAs solves this problem, enabling users to run the application through a browser and save it to the desktop for future use.

However, despite their strengths, PWAs also possess several shortcomings. The use of specific hardware API access for Bluetooth, advanced camera manipulation and background location services is limited compared to those used by native apps, especially when using an iOS device (Huber et al., 2022). Also, the capabilities of PWAs cannot be said to be supported uniformly across different browsers and operating systems. As far as Smart Tour is concerned, the fact that its functionality is primarily based on positioning using GPS technology, the mapping interface and recommendation engine functionalities makes it suitable for development using PWAs, even though other native capabilities like online booking are intentionally excluded.

2.10 Review of Related Works

A review of works related to the Smart Tour system was conducted across the following key themes: smart tourism, tourist guide systems, AI in tourism, recommendation systems, multilingual applications, context-aware computing, GIS and GPS, and Progressive Web Applications. The works reviewed and their contributions are presented below.

Koo et al. (2025) defined smart tourism as the integration of ICT and the tourism industry and proposed a conceptual framework for gathering and exploiting information from tourism industries, social media platforms and clients through AI, IoT, GIS and cloud computing. The study provides a 10-year retrospective model of AI-powered smart tourism, identifying key development gaps in personalised, real-time and integrated tourist systems.

Sarkar et al. (2023) reviewed the evolution of tourist guide systems from printed brochures to AI-powered applications and found that most existing systems address only one aspect of the tourist experience. The study identified the absence of a unified system combining GPS, AI recommendations and multilingual support as a major gap in the literature.

Nicolescu and Tudorache (2022) conducted a systematic literature review of human-computer interaction in customer service through AI chatbots. The study found that AI chatbot systems are limited to text-only interaction and struggle with complex or culturally nuanced queries, highlighting the need for richer AI-driven tourism interfaces.

Abbasi-Moud et al. (2022) developed a context-aware fuzzy-ontology-based tourism recommendation system (CAFOB) demonstrating that location-based, context-driven content delivery significantly improves tourist satisfaction compared to non-contextual systems. The system combined GPS context, user preferences and semantic ontologies but lacked multilingual support.

Sarkar et al. (2023) conducted a comprehensive survey of tourism recommendation systems, classifying methods into content-based filtering, collaborative filtering, knowledge-based filtering and hybrid approaches. The study identified information overload as the core problem that recommendation systems address and noted that hybrid approaches consistently outperform single-technique methods.

Huber et al. (2022) conducted a comparative study on the energy consumption and performance of Progressive Web Applications versus native apps. The study found that PWAs offer a superior balance of reachability, cross-platform functionality and ease of installation, making them well-suited for deployment in tourism contexts where users are unlikely to install dedicated native applications.

Buhalis and Cheng (2020) examined the adoption of AI-powered chatbots in hotels, including platforms such as Booking.com and Expedia. The study showed chatbots can handle booking enquiries and provide instant travel guidance but remain limited to text-only communication and struggle with complex queries.

Lim et al. (2020) proposed a machine learning approach to personalised trip recommendation for tourists based on user interests, visit durations and visit recency. The study demonstrated improved recommendation relevance but noted that machine learning algorithms require substantial user interaction data to function effectively, posing a limitation for new platforms.

Yoo et al. (2020) evaluated the quality of hybrid location data drawn from GPS-enabled mobile phones across different positioning methods including GPS, WiFi and cellular triangulation. The study found that GPS accuracy varies significantly depending on hardware and environment, with notable degradation in dense urban areas and indoor settings.

Peffers et al. (2022) proposed a Design Science Research (DSR) methodology for information systems research that centres the production of an artefact as the primary research output. The methodology structures research into problem identification, artefact design, development and evaluation phases, providing a rigorous framework for applied IS research.

Sassa et al. (2023) conducted a systematic literature review of Scrum as a software development framework. The study confirmed Scrum's effectiveness for managing complex, multi-module software projects through iterative sprint cycles, product backlogs and continuous delivery, all of which are directly applicable to the development of the Smart Tour system.

Google Developers (2020) defined Progressive Web Applications and their core capabilities including offline access via service workers, home screen installation, and push notifications. The documentation establishes PWAs as the standard approach for delivering app-like experiences through the web browser without requiring native installation.

2.11 Summary Table of Reviewed Works

Authors / Year	Contribution / Work Done	Techniques / Methodology	Limitations	Results	Findings
Koo et al. (2025)	Defined smart tourism as ICT-tourism integration; proposed a conceptual framework for data exploitation via AI, IoT and GIS.	Conceptual framework; literature synthesis	No implemented system; purely theoretical.	Platforms combining multiple ICT layers outperform single-technology systems in satisfaction and relevance.	A wide gap remains between smart tourism frameworks and fully deployed integrated systems.
Sarkar et al. (2023)	Reviewed tourist guide system evolution from brochures to AI-powered apps; found no app combines GPS, AI and multilingual support.	Comparative review; mobile prototype	Limited personalisation; no multilingual or PWA support.	Context-aware guides increased tourist satisfaction and destination certainty.	No single platform unifies GPS awareness, AI recommendations and multilingual content.
Abbasi-Moud et al. (2022)	Developed CAFOB, a context-aware fuzzy-ontology tourism recommendation system combining GPS context and semantic knowledge.	Context-aware computing; location-based services	No multilingual support; no personalisation for new users.	CAFOB achieved improved recommendation precision over static content delivery.	Context-aware recommendations significantly outperform non-contextual systems in tourism.

Authors / Year	Contribution / Work Done	Techniques / Methodology	Limitations	Results	Findings
Sarkar et al. (2023)	Surveyed mobile recommender systems in tourism; found most apps address only one aspect of the tourist journey.	Survey; mobile recommender systems	Did not implement a unified solution.	Hybrid approaches consistently outperformed single-technique methods across all scenarios.	No unified mobile tourist system combining GPS, AI and multilingual content currently exists.
Nicolescu and Tudorache (2022)	Systematic review of AI chatbots in customer service; found chatbots improve efficiency but struggle with complex queries.	AI; NLP; machine learning	Text-only interaction; fails on complex queries.	AI chatbots reduced response time for routine tourism enquiries.	Chatbots alone are insufficient; richer AI tourism interfaces are needed.
Sarkar et al. (2023)	Provided a taxonomy of tourist recommendation methods comparing content-based, collaborative, knowledge-based and hybrid filtering.	Survey; taxonomy of recommendation techniques	No implemented hybrid system for a specific tourism context.	Hybrid filtering outperformed all individual techniques across tourism scenarios.	No single technique is sufficient; combining methods is necessary for robust recommendations.
Buhalis and Cheng (2020)	Examined AI chatbot adoption in hotels including Booking.com and Expedia.	AI chatbots; natural language processing	Text-only; struggles with complex or cultural queries.	Chatbots successfully handled routine bookings and reduced	Customer satisfaction drops when chatbots cannot resolve complex questions.

Authors / Year	Contribution / Work Done	Techniques / Methodology	Limitations	Results	Findings
				response time.	
Sarkar et al. (2023)	Covered collaborative and content-based filtering; established hybrid systems yield superior recommendation quality.	Collaborative filtering; content-based filtering; hybrid methods	ML requires large datasets; cold-start problem for new users.	Hybrid combinations produced highest overall recommendation accuracy.	The cold-start problem is the most critical challenge for new recommendation platforms.
Lim et al. (2020)	Proposed personalised trip recommendation using visit recency and duration alongside user interests.	Machine learning; collaborative filtering; temporal modelling	Requires substantial historical data; impractical for new platforms.	Incorporating visit recency improved recommendation relevance measurably.	A handcrafted hybrid scoring approach is more appropriate than ML at platform launch.
Sarkar et al. (2023)	Reviewed and classified recommender systems research; identified information overload as the core tourism problem.	Literature review; classification framework	Survey-based; no tourism-specific or PWA implementation.	Information overload confirmed as the primary problem recommendation systems address.	Hybrid methods are underutilised despite superior performance; gap exists for developing regions.
Sarkar et al. (2023)	Detailed content-based filtering using feature vector analysis to model individual user preferences.	Content-based filtering; feature vector analysis	Cold-start problem; over-specialisation reduces diversity.	Performed reliably for users with established interaction histories.	Cold-start limits effectiveness for new users; standalone CB filtering is insufficient.

Authors / Year	Contribution / Work Done	Techniques / Methodology	Limitations	Results	Findings
Sarkar et al. (2023)	Surveyed collaborative filtering; identified user-based and item-based CF as dominant approaches.	Collaborative filtering; similarity measures	Sparse user-item matrix; high computational cost at scale.	CF enabled discovery of new destinations through collective user behaviour.	Sparsity and computational cost are the two most significant barriers to real-time CF.
Sarkar et al. (2023)	Introduced knowledge-based filtering using explicit user requirements and IF-THEN rule logic.	Knowledge-based filtering; rule-based inference	Manual knowledge base creation; no autonomous learning.	KBF achieved relevant recommendations for new users without any interaction history.	KBF eliminates the cold-start problem but limits scalability as item inventory grows.
Sarkar et al. (2023)	Proposed ontology-based user profiling using semantic connections to model preferences.	Ontology-based profiling; semantic modelling	Complex to construct; difficult to scale for large item sets.	Semantic profiling captured richer preference relationships than keyword matching.	Semantic connections between tourism concepts significantly improve recommendation relevance.
Abbasi-Moud et al. (2022)	Demonstrated that language barriers are a top cause of poor tourist experience; multilingual interfaces increase engagement.	Cross-cultural analysis; usability testing	Machine translation inaccurate for user-generated content.	Multilingual interfaces measurably increased user engagement and session duration.	Multilingual support is essential for tourist guide systems targeting international visitors.

Authors / Year	Contribution / Work Done	Techniques / Methodology	Limitations	Results	Findings
Abbasi-Moud et al. (2022)	Distinguished translation from localisation; showed automated translation fails to capture cultural nuances in tourism content.	Software localisation; translation quality analysis	Most apps support only major languages; excludes African regions.	Machine translation without localisation produced frequent errors in tourism content.	Professional localisation produces higher user satisfaction than automated translation alone.
Abbasi-Moud et al. (2022)	Defined context-aware computing; identified location, time and activity as the most critical context variables in tourism.	Context-aware computing; ubiquitous computing	Most apps use only basic GPS proximity; temporal and behavioural context ignored.	Temporal and spatial context integration improved recommendation relevance significantly.	Full context-aware computing is necessary; GPS proximity alone is insufficient.
Yoo et al. (2020)	Demonstrated effectiveness of GIS in tourist mapping; confirmed Leaflet.js makes GIS accessible without specialist expertise.	GIS; GPS; web mapping libraries	GPS degrades indoors and in urban canyons; high battery usage.	GPS-integrated mapping systems outperformed static guide books in navigation accuracy.	Web-based GIS libraries are production-ready; GPS-linked maps are now a baseline expectation.
Yoo et al. (2020)	Evaluated GPS accuracy across positioning methods including GPS, WiFi and cellular triangulation.	GPS accuracy analysis; comparative positioning	Up to 40m degradation in dense urban areas; battery drain from frequent polling.	GPS accuracy varies significantly by device type and environment.	IP-based geolocation fallback is a practical mitigation when GPS access is denied.

Authors / Year	Contribution / Work Done	Techniques / Methodology	Limitations	Results	Findings
Huber et al. (2022); Google Developers (2020)	Established PWA as a viable alternative to native apps; confirmed service worker caching enables offline access.	Progressive Web Application; service worker; offline caching	Limited hardware API access on iOS; uneven browser support.	PWAs achieved comparable user experience to native apps while eliminating the install barrier.	PWA architecture suits emerging markets where large native app installation is impractical.
Huber et al. (2022)	Analysed PWA energy consumption versus native apps; found PWAs consume up to 30% less battery on Android.	Hybrid mobile apps; PWA evaluation; user perception	Background location and Bluetooth limited on iOS.	PWAs consumed up to 30% less energy than equivalent native apps on Android.	iOS hardware API restrictions remain a barrier to full PWA and native app feature parity.

2.12 Research Gap

This chapter reviewed literature across eight themes directly relevant to the Smart Tour system: smart tourism, tourist guide systems, artificial intelligence in tourism, recommendation systems and filtering techniques, multilingual tourist applications, context-aware computing, Geographic Information Systems and GPS, and Progressive Web Applications. The review of related works presented in sections 2.2 to 2.10 and summarised in Table 2.1 reveals that while significant progress has been made within each individual theme, a substantial gap remains in the development of a tourist guide system that integrates all these elements into a single, coherent platform.

The following specific research gaps are identified from the reviewed literature and empirical findings:

• Fragmented system design: Koo et al. (2025) and Sarkar et al. (2023) confirm that current tourism guide systems are not integrated, addressing only one aspect of the tourist experience such as navigation or accommodation, while failing to provide comprehensive end-to-end assistance. No reviewed system combines AI-based personalisation, GPS context, multilingual support and offline accessibility in a single platform.

• Single-technique recommendation inadequacy: Sarkar et al. (2023) and Lim et al. (2020) demonstrate that single-technique recommendation methods, whether content-based, collaborative or knowledge-based, are each individually insufficient for the changing demands of tourists. Hybrid methods consistently outperform them but remain underutilised in tourism systems, particularly in emerging market deployments.

• Shallow multilingual support: Abbasi-Moud et al. (2022) and Yoo et al. (2020) confirm that the multilingual capabilities of most existing tourism applications are limited to basic interface translation without localisation of destination content or user-generated material, creating accessibility barriers for non-English-speaking tourists visiting sub-Saharan African destinations.

• Limited context awareness: Abbasi-Moud et al. (2022) found that context awareness in most available applications is confined to simple GPS proximity search. Temporal data, behavioural history and personal preference context are largely ignored, resulting in recommendations that lack relevance to the tourist's current situation.

• Underutilisation of PWA technology: Huber et al. (2022) and Google Developers (2020) established that PWA technology offers superior accessibility, offline capability and zero-installation deployment suited to low-end devices, yet it remains significantly underutilised in tourism systems where native apps continue to dominate despite creating barriers for users in developing countries.

• Absence of an integrated solution for sub-Saharan Africa: No reviewed system provides AI-enabled personalised recommendations, real-time GPS context awareness, multilingual support and Progressive Web Application delivery under a single solution designed for tourists in sub-Saharan African settings such as Enugu State, Nigeria.

It is precisely these gaps that the Smart Tour system was designed to address. By integrating a three-stage hybrid AI recommendation engine, GPS and GIS-based location services, a seven-language multilingual interface, context-driven suggestions and PWA offline accessibility into

a single cohesive application, Smart Tour represents a more complete and context-appropriate tourist guide solution than any system identified in the reviewed literature.

CHAPTER THREE

SYSTEM ANALYSIS, DESIGN AND METHODOLOGY

3.1 Introduction

The current chapter entails the analysis, design and methodology of the Smart Tour application, which is an AI-driven multilingual context-aware tourist guide progressive web application. This chapter is subdivided into six main parts, including research methodology, system analysis, system requirement specification, system design and methodology, Unified Modelling Language (UML) diagrams and system design. All these forms the detailed explanation of the procedures that were followed in designing and planning the Smart Tour system from the conceptualisation stage all the way to its actual design as depicted in Chapter Four. The Smart Tour system was developed using full-stack JavaScript technology, using React as the frontend, Node.js/Express REST API as the backend, and a PostgreSQL database managed using Supabase.

3.2 Research Methodology

The methodology of research explains how the whole process and strategy are conducted in order to perform this particular study. For this research, Design Science Research (DSR) methodology is used, where the main result produced by this approach is the artefact in the form of the Smart Tour application developed and tested for solving a problem associated with tourism information issues. Design science research is one of the most common methods applied to conduct information systems and software engineering research since it successfully unites the process of acquiring new knowledge and developing solutions (Peffers et al., 2022).

In the context of the DSR methodology, the research process involved the undertaking of three main processes. The first process was that of identifying and motivating the research problem by reviewing previous work on the topic of tourist guide systems as well as evaluating their weaknesses, which was discussed in Chapter Two. The second process entailed the designing and development of the Smart Tour artefact based on the functional and non-functional requirements derived from the analysis of the tourist guide system. The third process entailed the evaluation of the artefact through usability testing.

The Agile Scrum method was used as the software development strategy in the context of the overall DSR process. Agile Scrum is an iterative and incremental approach to software development that divides work into iterations known as sprints that produce demonstrable increments of the product. The choice of this method was made because the Smart Tour solution consists of several independent yet interconnected parts of the application: the GPS module, the artificial intelligence module, the multilingual interface, the RESTful API layer, and the PWA foundation (Sassa et al., 2023).

The development was carried out through a series of eight, two-week sprints. The product backlog, which contains all the features in the system, was specified right from the start and ranked according to functionality. Prior to beginning each sprint, the team held a sprint planning meeting where it selected the features to be developed during that sprint period. Once the sprint was completed, the team reviewed the increment developed before moving to the next sprint. Table 3.1 presents the product backlog and Table 3.2 presents the full sprint plan.

ID	**Feature**	**Description**	**Priority**	**Sprint**
PB-01	User Registration & Login	Tourists register and authenticate via Supabase Auth (email/password and JWT)	High	Sprint 1
PB-02	PWA Configuration	React PWA setup with service worker, manifest, and offline caching via react-pwa	High	Sprint 1
PB-03	Database Design	PostgreSQL schema design and Supabase table setup for all entities	High	Sprint 2
PB-04	Admin Dashboard	Admin interface for managing destinations, categories, and user data	High	Sprint 2

ID	Feature	Description	Priority	Sprint
PB-05	GPS Location Detection	Detect tourist's current position using browser Geolocation API	High	Sprint 3
PB-06	Interactive Map	Display destinations on Leaflet.js map with custom markers via React-Leaflet	High	Sprint 3
PB-07	Search & Filter	Search by name; filter by category, price range, rating, and distance	High	Sprint 4
PB-08	Destination Detail Pages	React dynamic pages with description, images, reviews, and map	High	Sprint 4
PB-09	AI Recommendation Engine	Hybrid content-based, collaborative, and knowledge-based filtering via Express API	High	Sprint 5
PB-10	Context-Aware Suggestions	Recommendations filtered by GPS location and time of day	High	Sprint 5
PB-11	Multilingual Support	Language detection and switching across five languages via react-i18next	Medium	Sprint 6
PB-12	Content Translation	Interface and destination descriptions in all five languages	Medium	Sprint 6

ID	Feature	Description	Priority	Sprint
PB-13	Ratings & Reviews	Tourists submit star ratings and written reviews stored in PostgreSQL	Medium	Sprint 7
PB-14	Favourites & Visit History	Save favourite destinations and view previously visited locations	Medium	Sprint 7
PB-15	Usability Testing	Structured testing with real users to evaluate system effectiveness	High	Sprint 8
PB-16	Bug Fixes & Optimisation	Performance improvements, cross-browser testing, error handling	High	Sprint 8

Table 3.1: Smart Tour Product Backlog

Sprint	Duration	Features Delivered	Milestone
Sprint 1	Weeks 1-2	User registration, login, Supabase Auth, React PWA setup, service worker	Authenticated PWA shell live
Sprint 2	Weeks 3-4	PostgreSQL schema, Supabase table setup, destination data seeding, admin dashboard	Database and admin panel complete
Sprint 3	Weeks 5-6	GPS detection, React-Leaflet map, custom markers, nearby destination display	Interactive map with live GPS
Sprint 4	Weeks 7-8	Search module, filters, React dynamic destination detail pages	Full browse and search functionality

Sprint	Duration	Features Delivered	Milestone
Sprint 5	Weeks 9-10	Hybrid AI recommendation engine (Express API), context-aware filtering	Personalised recommendations live
Sprint 6	Weeks 11-12	react-i18next multilingual module, auto detection, manual switching, translations	Five-language support complete
Sprint 7	Weeks 13-14	Ratings, reviews, favourites, visit history, user profile management	Full user profile features complete
Sprint 8	Weeks 15-16	Usability testing, bug fixes, performance optimisation, cross-browser testing	System ready for submission

Table 3.2: Sprint Plan

3.3 System Analysis

System analysis refers to the study and evaluation of the problem area to specify the requirements of the system needed for solving it. System analysis for the Smart Tour project entailed a careful scrutiny of existing tourist guide systems to discover their shortcomings, a description of the weaknesses of such systems, and an outline of the goals that the suggested system should fulfil to compensate for these weaknesses.

3.3.1 Analysis of Existing System

The analysis of the existing tour guides was carried out for determining the current situation regarding their availability and finding the shortcomings that need to be addressed by the Smart Tour system. The following three popular tour guide systems were reviewed: TripAdvisor, Google Maps and Sygic Travel.

TripAdvisor: TripAdvisor is among the leading online tourism websites that allow users to make comments, suggest hotels, recommend restaurants, and share details about different places across the globe. Its major advantage is the massive amount and variety of user contributions that enable tourists to benefit from their fellow travellers' insights and opinions. Yet, the recommender system used by TripAdvisor relies mostly on reviews' numbers and

ratings as opposed to modelling the preferences of particular tourists. TripAdvisor's recommendation process does not take into consideration any particular tourist's preferences or past visits, nor does it use GPS-based data for recommending relevant nearby services and places.

Google Maps: Google Maps is an advanced navigation and map application with superior capabilities in terms of its GPS and map display functionalities. Nevertheless, Google Maps is essentially a navigation app and not a tourism-orientated app. Google Maps does not have a built-in AI algorithm that provides recommendations for tourists based on their preferences, and it does not offer any personalisation features such as itinerary planning, language support or recording of visit history and favourites. In addition, offline maps can be accessed only by premium users in most locations.

Sygic Travel: Sygic Travel is a specific travel planning app that allows users to view maps, discover places to visit and manage trips. The app has GPS-enabled search functionality and provides extensive destination information. Nevertheless, the recommendations provided by the app do not depend on user preferences but are based on editorial choices made by Sygic Travel itself. The app does not have multilingual functionality, and it must be downloaded natively on the iOS or Android operating system.

Upon analysing all three systems, it becomes evident that although each system satisfies at least one need of the tourists, there is no such single system that includes artificial intelligence-based personalised suggestions, real-time GPS context-aware services, full support for multiple languages and PWAs that do not require any installation.

3.3.2 Limitation of Existing System

Based on the analysis in Section 3.3.1, the following key limitations were identified in existing tourist guide systems:

- Lack of personalised AI-based suggestions: The current system lacks the capacity to personalise the recommendation for destinations using the preferences, history and behaviour of an individual traveller. Everyone is shown the same type of suggestions despite different preferences.
- Lack of real-time contextual awareness: The majority of current systems fail to incorporate contextual data, such as the geographical location of the tourist as well as

the time, which affects the usefulness of the information and recommendations received, as they may often be irrelevant.

- Multilingual capabilities are limited: Most of the present systems offer just minimal interface translations without any localisation for the information about destinations, reviews from users or directions.
- Need to install native application: Native iOS or Android applications are typically required for most dedicated tourist guide applications, which require storage on the device, thus making it difficult and expensive to install on low-end devices available in developing countries like Nigeria.
- Dependence on continuous internet connectivity: Currently, the process of delivering content requires active connectivity through the internet, thereby posing a major challenge for those areas with poor internet connectivity, which is an area prevalent in Enugu State.
- Fragmented functionality: The current widely used systems do not provide an integrated approach for incorporating all the below-listed features into one package: AI-based suggestions, GPS map, language translations, offline use, review submission, visit history and favourites management.

3.3.3 Analysis of Proposed System

The analysis of the proposed Smart Tour system was carried out by evaluating how each identified limitation of existing systems, as detailed in Section 3.3.2, is directly addressed through specific design and implementation decisions in Smart Tour. This section presents a comparative analysis between the shortcomings of current tourist guide systems and the corresponding capabilities of the proposed system.

With regard to the lack of personalised AI-based suggestions in existing systems, Smart Tour addresses this limitation through a three-stage hybrid AI recommendation engine implemented as a RESTful Express API endpoint. The engine applies knowledge-based pre-filtering using the tourist's stated preferences, content-based cosine similarity ranking based on feature vectors derived from user interaction history, and collaborative filtering re-ranking using shared visit patterns among similar users. This pipeline produces a personalised, ranked list of destination recommendations specific to each individual tourist.

Concerning the absence of real-time contextual awareness, Smart Tour incorporates GPS-based context detection through the browser Geolocation API, which retrieves the tourist's precise latitude and longitude coordinates at the time of the recommendation request. These coordinates are passed directly to the recommendation API, where a PostGIS ST_DWithin spatial query filters destination candidates within the tourist's maximum acceptable distance. Additionally, the time of day is incorporated as a contextual parameter to filter destinations by operating hours, ensuring that recommendations are relevant not only to the tourist's location but also to their current moment.

To overcome the shallow multilingual support found in existing platforms, Smart Tour implements comprehensive multilingual functionality using the i18next and react-i18next libraries, supporting seven languages: English, French, Spanish, German, Italian, Portuguese and Japanese. The system automatically detects the user's preferred language from their browser configuration on first launch and allows manual language switching from any screen. All interface text, navigation labels, section titles, error messages and destination descriptions are localised through static JSON translation files, ensuring a fully translated experience rather than merely translating the interface shell.

The dependency on native application installation that characterises most existing tourist guide systems is eliminated in Smart Tour through its Progressive Web Application architecture. The system is deployed as a PWA using vite-plugin-pwa and a Workbox service worker, enabling installation to the device home screen directly from the browser without requiring access to the App Store or Google Play. The service worker implements four caching strategies that allow the application to function in offline mode, serving previously loaded destination content, map tiles and interface assets even when network connectivity is unavailable.

The fragmented functionality identified across TripAdvisor, Google Maps and Sygic Travel is resolved in Smart Tour by integrating all required tourist guide capabilities into a single, cohesive platform. Table 3.3 presents a comparative summary of how Smart Tour addresses each limitation of the existing systems.

Limitation of Existing Systems	**Smart Tour Solution**
No personalised AI recommendations	Three-stage hybrid AI engine: knowledge-based, content-based and collaborative filtering

No real-time context awareness	GPS coordinates and time-of-day integrated into every recommendation API request via PostGIS
Shallow multilingual support (interface only)	Seven-language full localisation via i18next covering all UI text and destination content
Requires native app installation	PWA architecture — installs from browser; no App Store required; works on all platforms
No offline capability	Workbox service worker with four caching strategies enables full offline content access
Fragmented functionality	Single platform combining GPS map, AI recommendations, multilingual support, reviews and favourites

Table 3.3: Comparative Analysis of Existing Systems and Smart Tour

3.3.4 Objective of Proposed System

The Smart Tour system is designed to directly address the limitations identified in Section 3.3.2. The specific objectives of the proposed system are as follows:

1. To develop and deploy a React progressive web application that does not require any installation of the application on a device and works offline using data that was already loaded into the system.
2. Implementing a GPS-enabled location detector that computes the exact geographical coordinates of the tourist in real-time and employs them to present destination recommendations through an interactive Leaflet.js mapping application.
3. Creating a hybrid artificial intelligence recommender system, provided as an API endpoint in a RESTful Express server, which incorporates content-based, collaborative and knowledge-based recommendation techniques to offer personalised suggestions to tourists.
4. Implementing a context-aware recommendation service that tailors destination recommendations according to the tourist's current GPS location, time and preferences.
5. To support multi-language translations in English, French, German, Spanish and Italian using the react-i18next package with automatic language detection and manual language selection.
6. To allow tourists to give destination ratings and write reviews, which would be saved in the PostgreSQL database hosted on Supabase.

7. To include personalisation options such as storing favourite destinations and tracking visit history to generate more accurate recommendations based on interaction history.
8. To develop an intuitive and easy-to-use interface using React and Tailwind CSS that can be used even by first-time tourists who have never used such software before.

3.4 System Requirement Specification

The system requirement specification is the formalisation of the requirements that the system should meet. The requirements were developed based on the goals mentioned in Section 3.3.3, the shortcomings of the current systems explained in Section 3.3.2 and the demands of the intended users of the application, namely tourists and administrative personnel.

3.4.1 Functional Requirements

Functional requirements state what behaviour or capabilities are expected from the Smart Tour application:

- The application will enable the tourist to create a new account through providing their full name, email and password using Supabase Auth along with JWT authentication and session management.
- The application will be able to recognise the geographical coordinates of the tourist and present their location using the React-Leaflet interactive map powered by the browser Geolocation API.
- Tourist destinations, hotels, restaurants and entertainment facilities will be marked on the interactive map through custom markers depending on the category and will be fetched from the Supabase PostgreSQL database via the Express REST API.
- The application will support searching for destinations based on their name, categorisation, cost range, star ratings and proximity to the tourist's location.
- All destination detail pages will be served dynamically using React, showing destination name, destination details, category, destination location on the map, images, rating score, user reviews, working hours and destination cost range.
- The recommendation generation module will use the Express recommendation API endpoint, where a hybrid filtering approach is applied to the data stored within the PostgreSQL database, and results are returned in a ranked order.

- The system will integrate the geographical position of the user based on GPS and the time of day in the recommendation API request for accurate recommendations.
- The application will be designed to have five languages, which include English, French, German, Spanish and Italian, with language detection using react-i18next and a manual switch for selecting any language from the navigation bar.
- Authenticated users should be able to post a rating (1-5 stars) and a textual review about any tourist destination, persisting this information into the PostgreSQL reviews table through the Express API.
- The application will support saving destinations to a favourites list by the user and logging all destinations visited by the user in a history log; both of these features should store data into the PostgreSQL database and allow accessing data from the user's profile page.
- The application will have an administrator panel that will provide users with access privileges for adding new destinations, modifying existing ones, and deleting destinations. The administrator panel will also support adding categories, managing reviews, and managing users' accounts via Express API endpoints, restricted through role-based middleware.
- The application will function as a progressive web app using react-pwa, providing the functionality for home screen installation and offline usage through service workers.

3.4.2 Non-Functional Requirements

The non-functional requirements are concerned with specifying the quality characteristics and constraints of the Smart Tour application:

- Performance: The React application should display the main interface within 3 seconds under normal conditions on a mobile broadband network. The Express recommendation API should respond to a valid request within 2 seconds.
- Usability: The user interface is easy to navigate without having gone through any kind of tutorial with all major functionalities available within 2 taps/clicks from the home screen.
- Reliability: Error handling middleware shall be used to catch and log any unhandled exceptions within the Express API server. The Supabase PostgreSQL database will utilize the cloud-managed architecture of Supabase for reliability.

- Scalability: The Express API will be stateless and thus scalable horizontally behind a load balancer when necessary. Connection pooling shall be supported through Supabase's PgBouncer functionality.
- Security: Routes requiring authentication in the API will have middleware to verify the token against the JWT. Row-level security policies in Supabase will allow access only to read and write personal data.
- Accessibility: The interface shall be designed to render correctly on all screen sizes between 320px on small smartphones to 1920px on desktop screens. This will be achieved with the use of mobile-first Tailwind CSS techniques.
- Compatibility: It will work correctly on the latest stable releases of Google Chrome, Firefox, Microsoft Edge and Safari browsers on both mobile and desktop operating systems
- Maintainability: The React front-end will implement a feature-based component approach, with proper naming conventions. In the case of the Express backend, the system will be designed using a layered architecture with code-level documentation for easier future expansion.
- Offline Support: With the help of the react-pwa service worker, the application will cache all its core files as well as all destination pages that have been visited by users before.

3.5 System Design and Methodology

This chapter explains the reasoning behind the development methodology chosen and the techniques employed for data collection to facilitate the development and population of the Smart Tour application.

3.5.1 Justification for Methodology

The chosen methodology for the development of the Smart Tour system was Agile Scrum, and the reasons for choosing it are the following:

Modular system structure: The various components that make up the Smart Tour include a React front end, an Express REST API, a PostgreSQL database using Supabase, the GPS & Maps feature, the AI engine and multi-language support. The sprint cycles of Agile Scrum would be very beneficial for this project since we have many interrelated yet

independent modules, which will help ensure that issues within each module can be detected separately before integration.

Iterative AI development: There was also a need for iteration in fine-tuning the hybrid recommendation system, as there was a need for testing and improving the filtering algorithm through several cycles of development. The agility offered by allowing for evolving implementation details against the backdrop of stable requirements made the entire process possible.

Early and continuous delivery: The provision made by Agile that every sprint should have a functional product ready for testing helped provide Smart Tour with functional versions during development for ongoing evaluations. This was better than the waterfall approach, where testing is carried out in one phase after all others, thereby limiting the amount of time to fix errors.

Full-stack development discipline: The simultaneous development of a front end using React and a back end using Node.js/Express entails coordinated attention to API contracts and data structures. The sprint constraints within an agile process ensured well-defined milestones, such as designing the database schema and basic API endpoints in Sprint 2 prior to developing dependent front-end components.

3.5.2 Method of Data Collection

The data for this research was gathered using three primary sources that aided in the design, development and population of the Smart Tour application.

Documentary Research: A detailed analysis of existing literature, documentation and tourism websites was carried out in order to collect the necessary theoretical and practical knowledge needed for developing the system. This involved examining smart tourism systems, recommendation system structures, progressive web app (PWA) guidelines, React and Express integration models, Supabase PostgreSQL setup, GPS integration methods and multilingual computing systems, as explained in Chapter Two.

API Data Collection: Data on destinations for the database of the system were gathered through automated means using the Google Places API. Specifically, queries were sent to the API to request information about tourist sites, accommodation facilities, eating places, and recreational centres in Enugu State, with the returned information structured to include the names of destinations, geographical coordinates, categories, ratings, operating hours,

photographs and descriptions. The harvested data was then pre-processed to fit into the schema used by the PostgreSQL database of the system hosted on Supabase.

Questionnaire and Usability Testing: An organised questionnaire on usability was developed and conducted among selected users during the usability testing stage of Sprint 8. The data gathered from this questionnaire covered areas such as user satisfaction, ease of use, perceived usefulness of recommendations provided by the AI system and usability.

3.6 Unified Modelling Language (UML)

Unified Modelling Language (UML) is a model language in software engineering that allows for specifying, visualising, implementing and documenting the artefacts of a software system. The following diagrams were created for the Smart Tour system: the Use Case Diagram, the Activity Diagram, the Entity Relationship Diagram, the Sequence Diagram and the System Architecture Diagram. These diagrams are discussed in the subsections that follow.

3.6.1 Use Case Diagram of the Proposed System

A use case diagram shows interaction between various actors and uses cases, i.e., functions, of the system under discussion. This diagram offers an overview of the functionalities of the system without going into any depth about its implementation.

The Smart Tour system involves two human actors and one system actor. The first human actor is the tourist, who stands for any registered user of the Smart Tour system. The second human actor is the administrator, who acts as the manager of the system and manages destination data and content. The third actor is the System actor. The system actor is an automated process done by the application independent of a user action.

The Tourist interacts with the following use cases: Register Account, Login, View Home Screen and Recommendations, Search Destinations, Filter Destinations, View Destination Detail Page, View Interactive Map, Save to Favourites, View Visit History, Submit Rating and Review, Switch Language and Install Application as PWA.

The Administrator interacts with the following use cases: Login to Admin Dashboard, Add Destination, Edit Destination, Delete Destination, Manage Categories, Moderate Reviews and Manage User Accounts.

The System actor executes the following automated use cases: Detect GPS Location, Call Recommendation API, Apply Context Filters (location and time), Auto-detect User Language, and Cache Content for Offline Access via service worker.

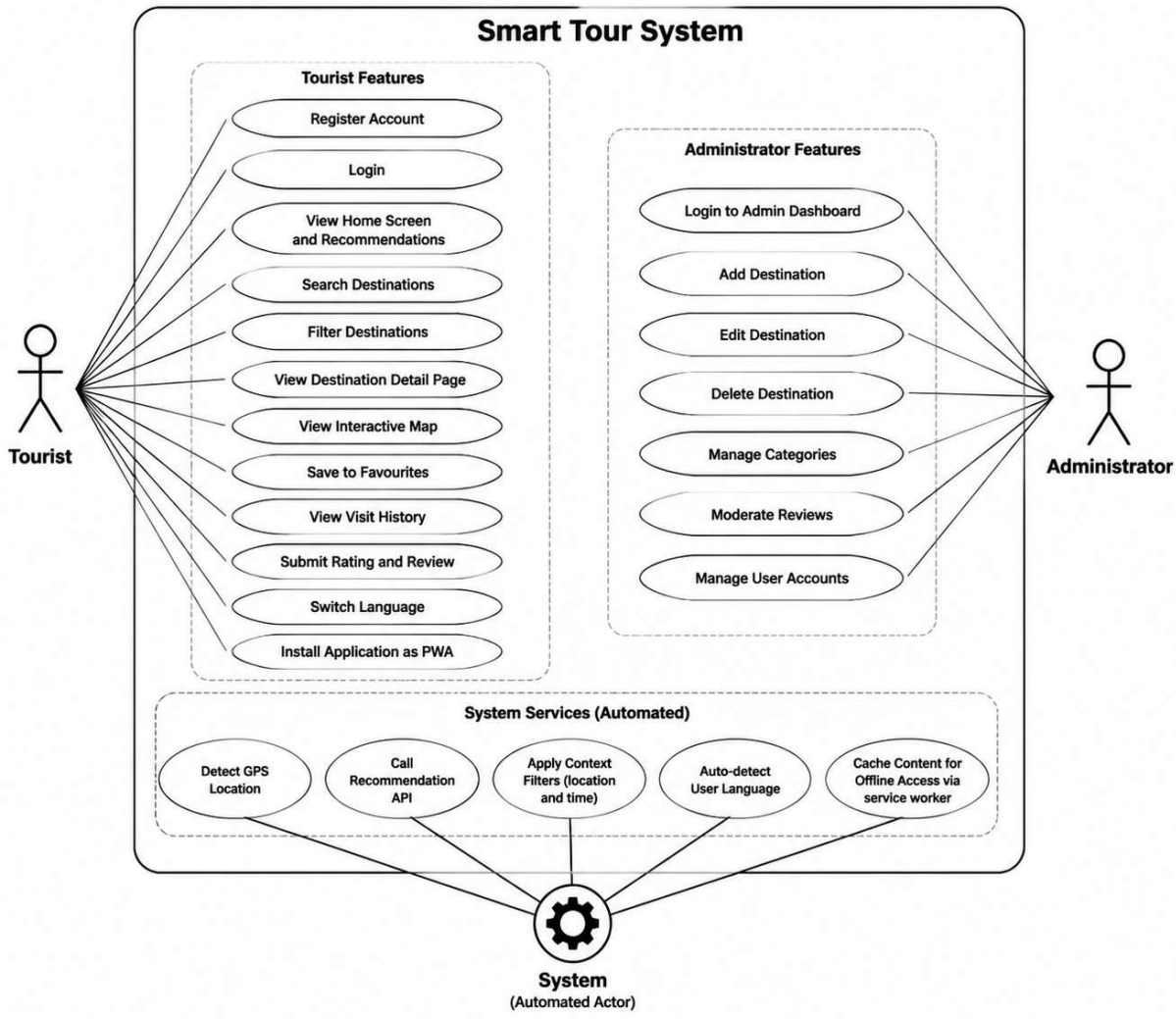

Figure 3.1: Use Case Diagram of the Smart Tour System

3.6.2 Activity Diagram of the Proposed System

An activity diagram depicts the flow of activities performed during the execution of a system process by including decisions and steps taken from start to finish. Activity diagrams of the Smart Tour System are shown below.

Registration and Login Activity: This interaction starts when the user launches the Smart Tour app. The middleware created using React then checks whether a Supabase Auth session token is available. If yes, the user will be directed to the personalised homepage;

otherwise, the options for logging in and registering appear. In the case of login, the user enters the credentials (email and password) that are then sent to the Supabase Auth endpoint. If authentication fails, the error message is generated, and the user can try again. If successful, the JWT token will be delivered and stored in memory, and the user is then redirected to the homepage. In the case of registration, the user will input his/her full name, email, password and desired language, and Supabase Auth will create a new account, followed by the user profile record creation in PostgreSQL by the Express API.

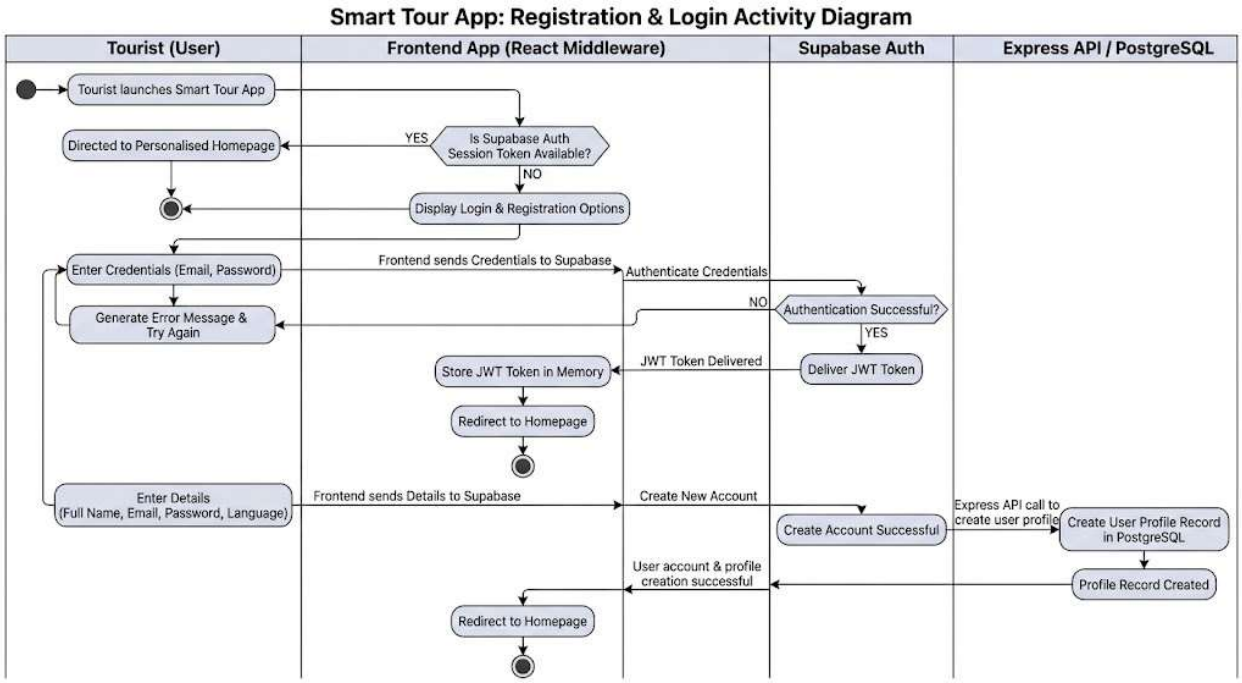

Figure 3.2: Registration and Login Activity Diagram

Search and Recommendation Activity: The process starts when the user gets to the home page. The front end detects the location of the tourist using GPS coordinates and sends an API call for recommendations to the Express server. The API uses the hybrid filter approach on the PostgreSQL database and generates a list of recommendations in cards. The user can also go to the search page and enter their search terms while applying filters. The Express search API makes a search query to the PostgreSQL destinations table using the constraints and returns search results ordered based on proximity. In case there are no matching destinations, the user is asked to revise their constraints.

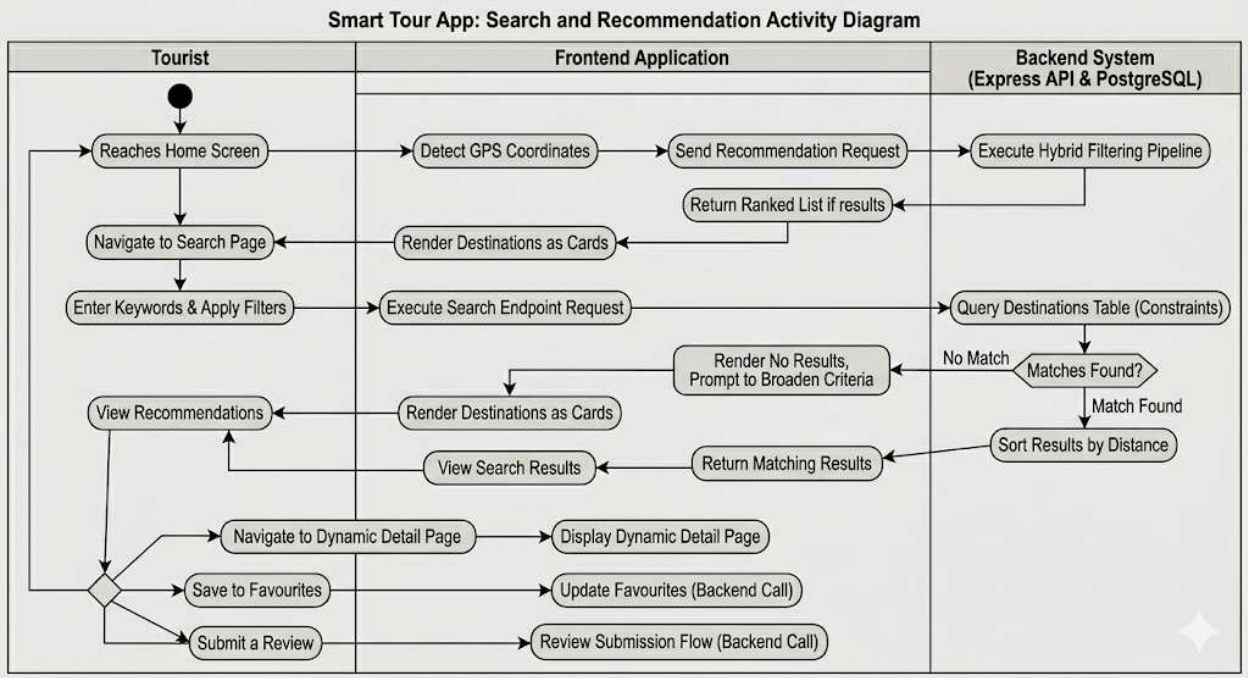

Figure 3.3: Search and Recommendation Activity Diagram

3.6.3 Sequence Diagram

The sequence diagram represents the time order of events and their relationships among different parts of the system during a particular process. The sequence diagram for generating recommendations using the Smart Tour AI engine is provided since it is the most technologically complicated interaction in the system that includes the React frontend, Express API and Supabase PostgreSQL database.

The recommendation generation sequence proceeds as follows:

1. The home page component on the React application fetches the user's GPS location data from the Geolocation API through the browser.
2. The Geolocation API provides the user's GPS latitude and longitude to the frontend component.
3. A GET request is made by the frontend component to the Express recommendations API, with the JWT authentication token provided in the Authorisation header and the GPS coordinates and timestamp in the query string.
4. The JWT middleware authenticates the token and reads the user_id from the JWT payload.

5. The recommendations API then retrieves the user's preference profile from the PostgreSQL user_preferences table with the use of user_id.
6. The PostgreSQL user_preference record is retrieved by the recommendations API.
7. The controller uses a PostGIS distance query against the destinations table to obtain destinations that lie in the specified proximity radius around the specified GPS coordinates.
8. The PostgreSQL database provides the filtered destinations.
9. The controller makes a query against the visit_history and reviews tables to get information about the past interactions.
10. The recommendation engine runs a three-step hybrid recommendation pipeline, including the following: (a) a knowledge-based pre-filtering phase with hard preference restrictions; (b) a content-based phase for ranking the destinations based on the similarity to the user interaction history; (c) a collaborative filtering re-ranking step using other users' signals.
11. The result is passed back to the React frontend via an Express controller and a JSON request.
12. The front end displays the recommendations in the form of cards

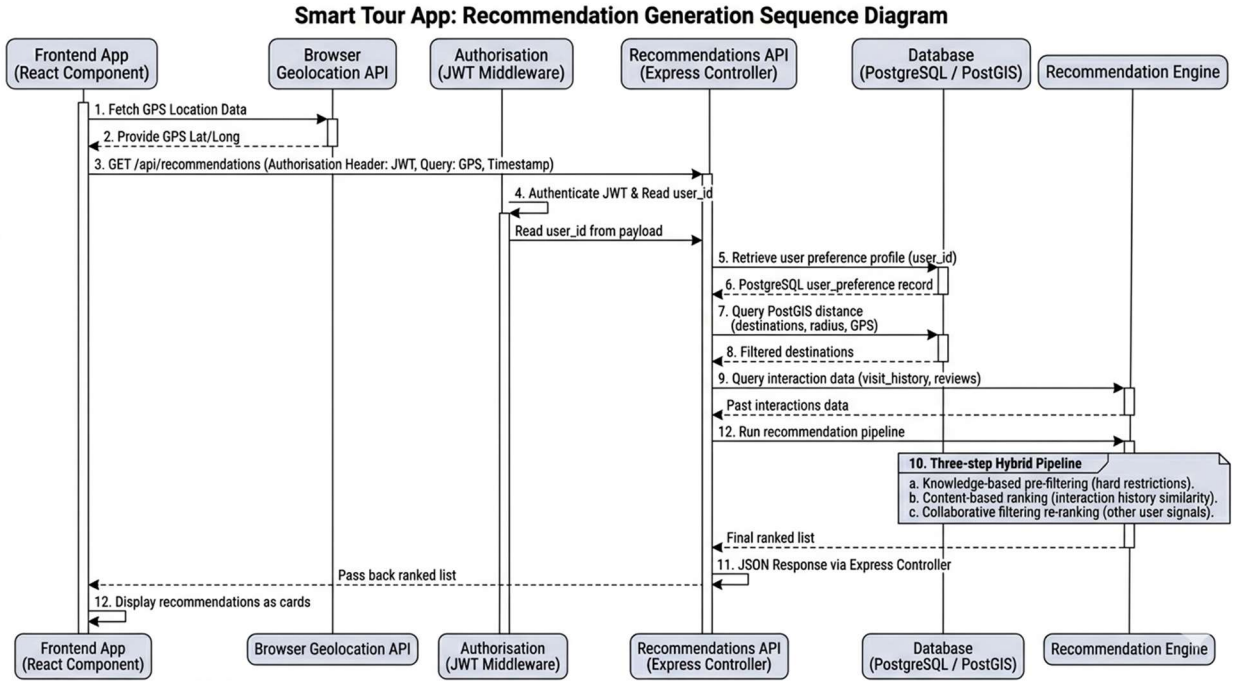

Figure 3.4: AI Recommendation Generation Sequence Diagram

3.7 System Design

The system design involves transforming the needs and architecture from previous sections into actual design specifications for implementation. The following is the design of the PostgreSQL database schema, the recommendation engine architecture and the technology stack.

3.7.1 System Architecture

The system architecture determines the basic structure of the Smart Tour application, including its key components and their relationships. The Smart Tour system is designed using a three-tier client/server architecture that includes the presentation layer, API layer and data layer.

Frontend Tier (React PWA): The front end is implemented using React and consists of both a client-side and server-side implementation. It renders all UI-related content through React components, manages client-side routing, fetches geographical coordinates from the Geolocation API of the browser, displays an interactive map through React-leaflet, provides localisation using react-i18next, and makes RESTful API calls to the backend via HTTP. In order to make the web application progressive, React PWA is used to configure the app as a Progressive Web Application. It automatically creates a service worker that caches static files and pages at runtime.

Backend Tier (Node.js / Express REST API): The backend consists of a Node.js server using the Express framework to serve as an API with a number of RESTful endpoints consumed by the React frontend app. This API is divided into the following categories of routes: authentication routes (using the Supabase Auth service), destination routes (for CRUD and searching), recommendation routes (using the hybrid AI algorithm pipeline) and user routes (for profile management, favourites, visit history and reviews). All restricted routes use JWT validation middleware for securing them.

Data Storage Tier (Supabase PostgreSQL): PostgreSQL is the relational database managed by Supabase and forms the main data repository of the system. The data on destinations, users, reviews, ratings, favourites, visits and recommendations is stored in tables of relations with necessary foreign key relationships and indexing. The Row-Level Security (RLS) policy is applied to manage the data access control at the database level. The PostGIS extension is employed to support distance-based searches in geographic locations based on the user's location coordinates.

External Integrations: The system is based on three external APIs, including the browser's Geolocation API, which allows getting GPS coordinates; the react-leaflet library using the OpenStreetMap tile server for map rendering that has no restriction by the API key; and react-i18next using static JSON translations for multiple language support.

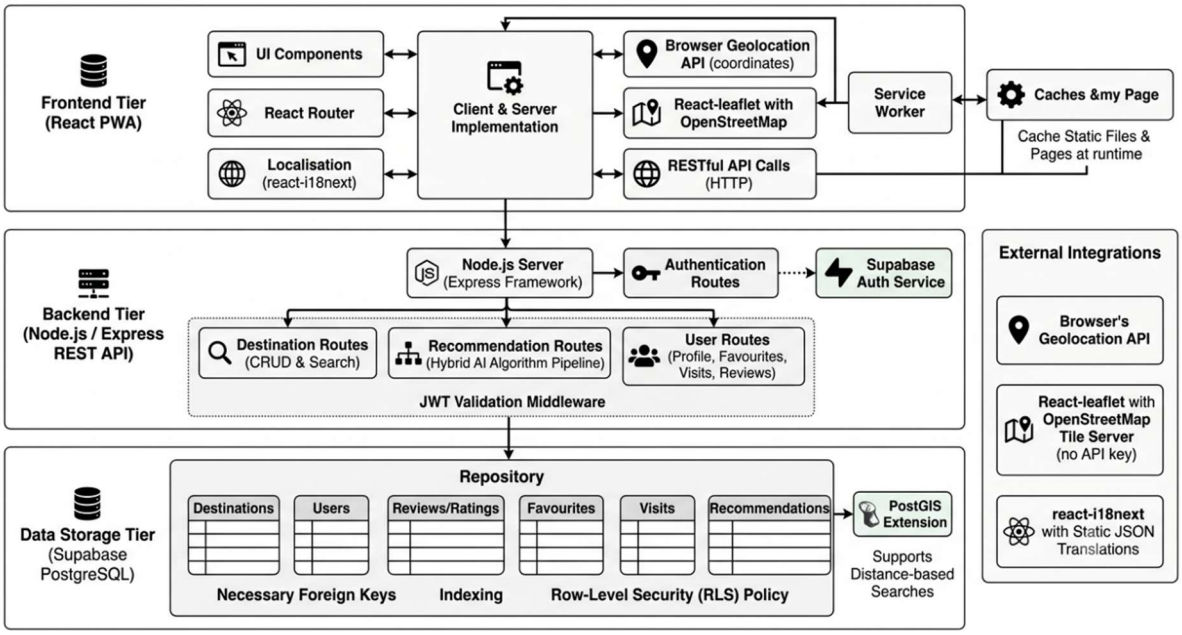

Figure 3.5: Smart Tour System Architecture Diagram

3.7.2 Database Schema Design

The Smart Tour PostgreSQL database contains a total of eight relational tables. The database schema was designed using Supabase, an open-source managed PostgreSQL platform with built-in row-level security, real-time subscriptions and support for the PostGIS geospatial database extension. Each table is described in detail below.

users Table

The users table serves as the central identity store for all registered tourists and administrators on the Smart Tour platform. It contains the following fields: user_id is a UUID primary key generated automatically using the gen_random_uuid() function, ensuring each user has a globally unique identifier that cannot be predicted or incremented sequentially. full_name stores the tourist's full name as a VARCHAR(100) field and is marked NOT NULL to enforce that registration cannot be completed without a name. email is stored as a VARCHAR(150)

field with a UNIQUE constraint, preventing duplicate accounts from being created with the same email address. preferred_language stores the user's chosen interface language as a VARCHAR(10) field with a default value of 'en' for English, supporting the multilingual functionality of the system. created_at records the timestamp of account creation using TIMESTAMPTZ DEFAULT NOW(), and last_login records the most recent login timestamp, enabling session tracking and activity monitoring.

categories Table

The categories table defines the classification system used to organise tourist destinations within the Smart Tour system. category_id is a UUID primary key auto-generated using gen_random_uuid(). category_name is stored as a VARCHAR(50) field with a UNIQUE NOT NULL constraint, preventing duplicate or empty category names. icon_url is a TEXT field storing the URL path to the visual icon associated with each category, used in the frontend to display category badges on destination cards. The categories table is referenced as a foreign key by the destinations table, establishing a one-to-many relationship whereby a single category can classify many destinations.

destinations Table

The destinations table is the core content table of the Smart Tour system, storing all tourist attraction records retrieved and seeded from the Google Places API. destination_id is a UUID primary key. name stores the destination name as a VARCHAR(200) NOT NULL field. description is a TEXT field holding a full descriptive summary of the destination. category_id is a UUID foreign key referencing categories(category_id), linking each destination to its classification. latitude and longitude are stored as DECIMAL(9,6) fields, providing the precise geographic coordinates used by the PostGIS extension and the Haversine formula in the recommendation engine for proximity scoring. price_range is a VARCHAR(20) field storing a textual price indicator such as Budget, Moderate or Premium. opening_hours is stored as a JSONB field, allowing flexible storage of structured hours data without requiring a fixed schema. images is a TEXT array field storing multiple image URLs associated with a destination. average_rating is a DECIMAL(3,2) field with a default of 0, updated dynamically when new reviews are submitted. total_reviews is an INT field tracking the number of reviews associated with each destination. visit_count is an INT field that increments each time a tourist records a visit to the destination.

reviews Table

The reviews table stores all-star ratings and written reviews submitted by authenticated tourists about specific destinations. review_id is a UUID primary key. user_id is a UUID foreign key referencing users(user_id) with ON DELETE CASCADE, meaning that all reviews submitted by a user are automatically deleted if the user account is removed. destination_id is a UUID foreign key referencing destinations(destination_id) with ON DELETE CASCADE, ensuring reviews are removed if the associated destination is deleted. star_rating is an INT field with a CHECK constraint enforcing that only values between 1 and 5 are accepted, preventing invalid rating data from entering the system. review_text is a TEXT field storing the written review content. created_at records the timestamp of submission using TIMESTAMPTZ DEFAULT NOW().

favourites Table

The favourites table manages the list of destinations that each tourist has saved to their personal favourites. favourite_id is a UUID primary key. user_id and destination_id are both UUID foreign keys referencing their respective tables, both defined with ON DELETE CASCADE to maintain referential integrity when users or destinations are removed. saved_at records the timestamp when the destination was saved. A composite UNIQUE constraint on (user_id, destination_id) prevents a tourist from saving the same destination more than once, enforcing data integrity at the database level.

visit_history Table

The visit_history table records every destination visit logged by a tourist, providing the interaction data used by the collaborative filtering stage of the recommendation engine. visit_id is a UUID primary key. user_id and destination_id are UUID foreign keys with ON DELETE CASCADE. visited_at records the exact timestamp of each visit using TIMESTAMPTZ DEFAULT NOW(). The visit_history table is queried by the recommendation engine to identify similar users based on shared visit patterns, enabling the collaborative filtering re-ranking step of the hybrid pipeline.

user_preferences Table

The user_preferences table stores each tourist's explicitly stated preferences, which are used by the knowledge-based pre-filtering stage of the recommendation engine. preference_id is a UUID primary key. user_id is a UUID foreign key with a UNIQUE constraint, enforcing a strict one-to-one relationship between a user and their preference profile, each tourist can have only one preference record. preferred_categories is a UUID array field storing the category IDs that the tourist has selected as their areas of interest, used to filter destination candidates in the knowledge-based stage. max_price_range is a VARCHAR(20) field specifying the tourist's maximum acceptable price level. min_rating is a DECIMAL(2,1) field with a default of 1.0, representing the minimum acceptable destination rating. max_distance_km is an INT field with a default of 50, specifying the maximum distance in kilometres within which destinations should be recommended.

recommendations Table

The recommendations table stores the output of the hybrid AI recommendation engine for each request, enabling caching of results and audit of how recommendations were generated. recommendation_id is a UUID primary key. user_id is a UUID foreign key referencing users(user_id) with ON DELETE CASCADE. destination_ids is a UUID array field storing the ordered list of recommended destination IDs returned by the recommendation pipeline, maintaining the ranked order produced by the composite scoring function. generated_at records the timestamp of recommendation generation using TIMESTAMPTZ DEFAULT NOW(), supporting time-based analysis of recommendation patterns. filter_method is a VARCHAR(50) field recording which combination of filtering techniques — knowledge-based, content-based or hybrid was applied to generate the recommendation set.

3.7.3 Database Design Table

The tables below present the detailed structural design of each relational table described in Section 3.7.2, showing the field name, data type, applicable constraints and description of each attribute in the Smart Tour database schema.

Table 3.4: users Table Structure

Field Name	Data Type	Constraint	Description
user_id	UUID	PRIMARY KEY, DEFAULT gen_random_uuid()	Uniquely identifies each registered user
full_name	VARCHAR(100)	NOT NULL	Stores the tourist's or administrator's full name
email	VARCHAR(150)	UNIQUE, NOT NULL	Stores the user's email address; prevents duplicate accounts
preferred_language	VARCHAR(10)	DEFAULT 'en'	Stores the user's chosen interface language
created_at	TIMESTAMPTZ	DEFAULT NOW()	Records the timestamp of account creation
last_login	TIMESTAMPTZ	—	Records the most recent login timestamp

Table 3.5: categories Table Structure

Field Name	Data Type	Constraint	Description
category_id	UUID	PRIMARY KEY, DEFAULT gen_random_uuid()	Uniquely identifies each destination category
category_name	VARCHAR(50)	UNIQUE, NOT NULL	Stores the name of the category
icon_url	TEXT	—	Stores the URL path to the category's visual icon

Table 3.6: destinations Table Structure

Field Name	Data Type	Constraint	Description
destination_id	UUID	PRIMARY KEY	Uniquely identifies each tourist destination
name	VARCHAR(200)	NOT NULL	Stores the name of the destination
description	TEXT	—	Holds a full descriptive summary of the destination

Field Name	Data Type	Constraint	Description
category_id	UUID	FOREIGN KEY → categories(category_id)	Links the destination to its classification
latitude	DECIMAL(9,6)	—	Stores the geographic latitude coordinate
longitude	DECIMAL(9,6)	—	Stores the geographic longitude coordinate
price_range	VARCHAR(20)	—	Stores a textual price indicator (Budget, Moderate, Premium)
opening_hours	JSONB	—	Stores structured opening-hours data
images	TEXT[]	—	Stores multiple image URLs for the destination
average_rating	DECIMAL(3,2)	DEFAULT 0	Stores the average of all submitted ratings
total_reviews	INT	DEFAULT 0	Tracks the number of reviews for the destination
visit_count	INT	DEFAULT 0	Tracks the number of recorded visits

Table 3.7: reviews Table Structure

Field Name	Data Type	Constraint	Description
review_id	UUID	PRIMARY KEY	Uniquely identifies each review
user_id	UUID	FOREIGN KEY → users(user_id), ON DELETE CASCADE	Identifies the reviewer
destination_id	UUID	FOREIGN KEY → destinations(destination_id), ON DELETE CASCADE	Identifies the reviewed destination
star_rating	INT	CHECK (1–5)	Stores the numeric rating given by the tourist
review_text	TEXT	—	Stores the written review content

Field Name	Data Type	Constraint	Description
created_at	TIMESTAMPTZ	DEFAULT NOW()	Records the timestamp of submission

Table 3.8: favourites Table Structure

Field Name	Data Type	Constraint	Description
favourite_id	UUID	PRIMARY KEY	Uniquely identifies each favourite record
user_id	UUID	FOREIGN KEY → users(user_id), ON DELETE CASCADE	Identifies the tourist
destination_id	UUID	FOREIGN KEY → destinations(destination_id), ON DELETE CASCADE	Identifies the saved destination
saved_at	TIMESTAMPTZ	DEFAULT NOW()	Records when the destination was saved
(user_id, destination_id)	—	UNIQUE (composite)	Prevents a tourist from saving the same destination twice

Table 3.9: visit_history Table Structure

Field Name	Data Type	Constraint	Description
visit_id	UUID	PRIMARY KEY	Uniquely identifies each visit record
user_id	UUID	FOREIGN KEY → users(user_id), ON DELETE CASCADE	Identifies the tourist who made the visit
destination_id	UUID	FOREIGN KEY → destinations(destination_id), ON DELETE CASCADE	Identifies the visited destination
visited_at	TIMESTAMPTZ	DEFAULT NOW()	Records the exact timestamp of the visit

Table 3.10: user_preferences Table Structure

Field Name	Data Type	Constraint	Description
preference_id	UUID	PRIMARY KEY	Uniquely identifies each preference record
user_id	UUID	FOREIGN KEY, UNIQUE	Enforces a one-to-one relationship between a user and their preference profile
preferred_categories	UUID[]	—	Stores the category IDs the tourist has selected as areas of interest
max_price_range	VARCHAR(20)	—	Stores the tourist's maximum acceptable price level
min_rating	DECIMAL(2,1)	DEFAULT 1.0	Stores the minimum acceptable destination rating
max_distance_km	INT	DEFAULT 50	Stores the maximum distance, in kilometres, for recommendations

Table 3.11: recommendations Table Structure

Field Name	Data Type	Constraint	Description
recommendation_id	UUID	PRIMARY KEY	Uniquely identifies each recommendation record
user_id	UUID	FOREIGN KEY → users(user_id), ON DELETE CASCADE	Identifies the recipient tourist
destination_ids	UUID[]	—	Stores the ranked list of recommended destination IDs
generated_at	TIMESTAMPTZ	DEFAULT NOW()	Records the timestamp of recommendation generation
filter_method	VARCHAR(50)	—	Records which filtering technique(s) were applied — knowledge-based, content-based or hybrid

3.7.4 AI Recommendation Engine Design

The recommendation engine is implemented as a service module within the Express API, invoked by the GET /api/recommendations endpoint. It executes a three-stage hybrid filtering pipeline:

1. Knowledge-Based Pre-Filtering: The engine fetches the preference settings of the user from the table user_preferences. After that, it fetches data from the table destinations using a PostGIS geographic filter of ST_DWithin to select those destinations which lie in a radius defined by the user's maximum acceptable distance from his/her current location. The destination's categories are filtered for those which are in the preferred_categories array of the user. Also, the price range should be lower than the user's max_price_range. Finally, the average rating should exceed the user's min_rating and the destination should not be closed for business during the hours of fetching.
2. Content-Based Ranking: For each of the remaining destinations, compute the similarity score based on the recommendations generated by the system. The similarity score will be generated through computing the cosine similarity between the feature vector of the destination, containing information about its category, normalised price range, average rating, and keywords in the description, and the feature vector of the user's preference, which is based on the weighted sum of feature vectors for destinations already visited or favourited by the user.
3. Collaborative Filtering Enhancement: The service performs a search in the visit_history and reviews databases for the five users who are most alike the current one according to the number of matchings visited destinations and positively reviewed destinations. Those destinations that have been rated by similar users with four or five stars but remain unvisited by the current user are located and added to the list of candidates in hybrid positions, thus enabling an exploratory component to avoid redundant recommendations.

The final composite-scored list is truncated to the top ten recommendations and returned as a JSON array of destination objects in the Express API response.

3.7.5 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) provides a graphical representation of the logical data structure of the Smart Tour system, illustrating the entities that make up the database, the attributes that define each entity, and the relationships that exist between them. The ERD serves as the blueprint from which the relational database schema was implemented in Supabase using PostgreSQL. Figure 3.6 presents the ERD for the Smart Tour system.

Entities and Their Attributes

The Smart Tour system comprises eight primary entities, each corresponding to a relational table in the PostgreSQL database.

The **user's** entity represents every registered user of the Smart Tour platform, whether a tourist or an administrator. Its primary key is user_id, a UUID that uniquely identifies each user across the entire system. The entity stores full_name and email as identifying attributes, with email carrying a uniqueness constraint to prevent duplicate accounts. The password_hash attribute stores the bcrypt-encrypted version of the user's password, ensuring that plaintext credentials are never persisted in the database. preferred_language records the user's chosen interface language, linking the user account to the multilingual module. created_at and last_login are temporal attributes that record account creation time and the most recent login event respectively.

The **destinations** entity is the central content entity of the system, representing every tourist attraction, hotel, restaurant and leisure facility stored in the Smart Tour database. Its primary key is destination_id, a UUID uniquely identifying each destination. The name and description attributes provide the display information shown to tourists on destination cards and detail pages. category_id is a foreign key linking each destination to the categories entity, classifying it under a predefined type such as Nature, Culture, Dining or Hotels. The latitude and longitude attributes are DECIMAL fields storing the precise geographic coordinates of the destination, which are used by the PostGIS extension for spatial distance queries and by the Haversine formula in the recommendation engine for proximity scoring. price_range provides a textual price classification, opening_hours stores structured time data in JSONB format, and images stores multiple photograph URLs as a TEXT array. average_rating and total_reviews are computed attributes updated dynamically with each new review submission, while visit_count tracks how many times tourists have logged a visit to the destination.

The **categories** entity defines the classification taxonomy used to group destinations by type. Its primary key is category_id, a UUID. The category_name attribute stores the display name of each category, and icon_url stores the path to the visual icon rendered in the frontend interface. The categories entity acts as a lookup table referenced by the destinations entity through a foreign key relationship.

The **reviews** entity stores every star rating and written review that authenticated tourists submit about destinations. Its primary key is review_id, a UUID. The user_id and destination_id attributes are foreign keys establishing the relationship between the review, the tourist who wrote it and the destination it concerns. star_rating is an integer attribute constrained to values between 1 and 5, and review_text stores the written content of the review. created_at records the timestamp of submission.

The **favourites** entity manages each tourist's personal list of saved destinations. Its primary key is favourite_id, a UUID. The user_id and destination_id attributes are both foreign keys linking the record to the respective user and destination. The saved_at attribute records when the tourist added the destination to their favourites. A composite uniqueness constraint on the combination of user_id and destination_id ensures that a tourist cannot save the same destination more than once.

The **visit_history** entity records every destination visit that a tourist logs through the Smart Tour system. Its primary key is visit_id, a UUID. The user_id and destination_id foreign keys identify which tourist visited which destination, and visited_at records the precise timestamp of the visit. This entity is critical to the operation of the collaborative filtering stage of the recommendation engine, as it provides the shared visit history used to identify similar users.

The **recommendations** entity stores the output generated by the hybrid AI recommendation engine for each user request. Its primary key is recommendation_id, a UUID. The user_id foreign key identifies the tourist for whom the recommendations were generated. destination_ids is a UUID array attribute that stores the ordered list of recommended destination identifiers, preserving the ranked order produced by the composite scoring algorithm. generated_at records the timestamp of generation, and filter_method records which combination of filtering techniques was applied to produce the result set.

The **user_preferences** entity stores each tourists explicitly stated preferences, which are consumed by the knowledge-based pre-filtering stage of the recommendation engine. Its primary key is preference_id, a UUID. The user_id foreign key carries a UNIQUE constraint, enforcing a strict one-to-one relationship between a user and their preference profile. preferred_categories is a UUID array attribute containing the category identifiers that the tourist has selected as areas of interest. max_price_range specifies the tourist's maximum acceptable price level, min_rating defines the minimum star rating a destination must have to be included in recommendations, and max_distance_km defines the maximum proximity radius within which destinations will be considered.

Relational Constraints

The relationships between entities in the Smart Tour ERD are defined by foreign key constraints that enforce referential integrity across the database. The following relationships exist within the system.

The relationship between **users and reviews** is one-to-many, meaning a single user can write many reviews but each review belongs to exactly one user. This relationship is enforced through the user_id foreign key in the reviews table.

The relationship between **destinations and reviews** is one-to-many, meaning a single destination can receive many reviews but each review concerns exactly one destination. This is enforced through the destination_id foreign key in the reviews table.

The relationship between **users and favourites** is one-to-many, meaning a single user can save many destinations to their favourites list. The relationship is enforced through the user_id foreign key in the favourites table.

The relationship between **destinations and favourites** is one-to-many, meaning a single destination can be saved by many different users. This is enforced through the destination_id foreign key in the favourites table.

The relationship between **categories and destinations** is one-to-many, meaning a single category can classify many destinations but each destination belongs to exactly one category. This is enforced through the category_id foreign key in the destinations table.

The relationship between **users and visit_history** is one-to-many, meaning a single user can log many destination visits over time. This is enforced through the user_id foreign key in the visit_history table.

The relationship between **users and user_preferences** is one-to-one, meaning each user has exactly one preference profile and each preference profile belongs to exactly one user. This is enforced through the UNIQUE constraint on the user_id foreign key in the user_preferences table.

The relationship between **users and recommendations** is one-to-many, meaning the recommendation engine can generate and store multiple recommendation sets for a single user across different sessions. This is enforced through the user_id foreign key in the recommendations table.

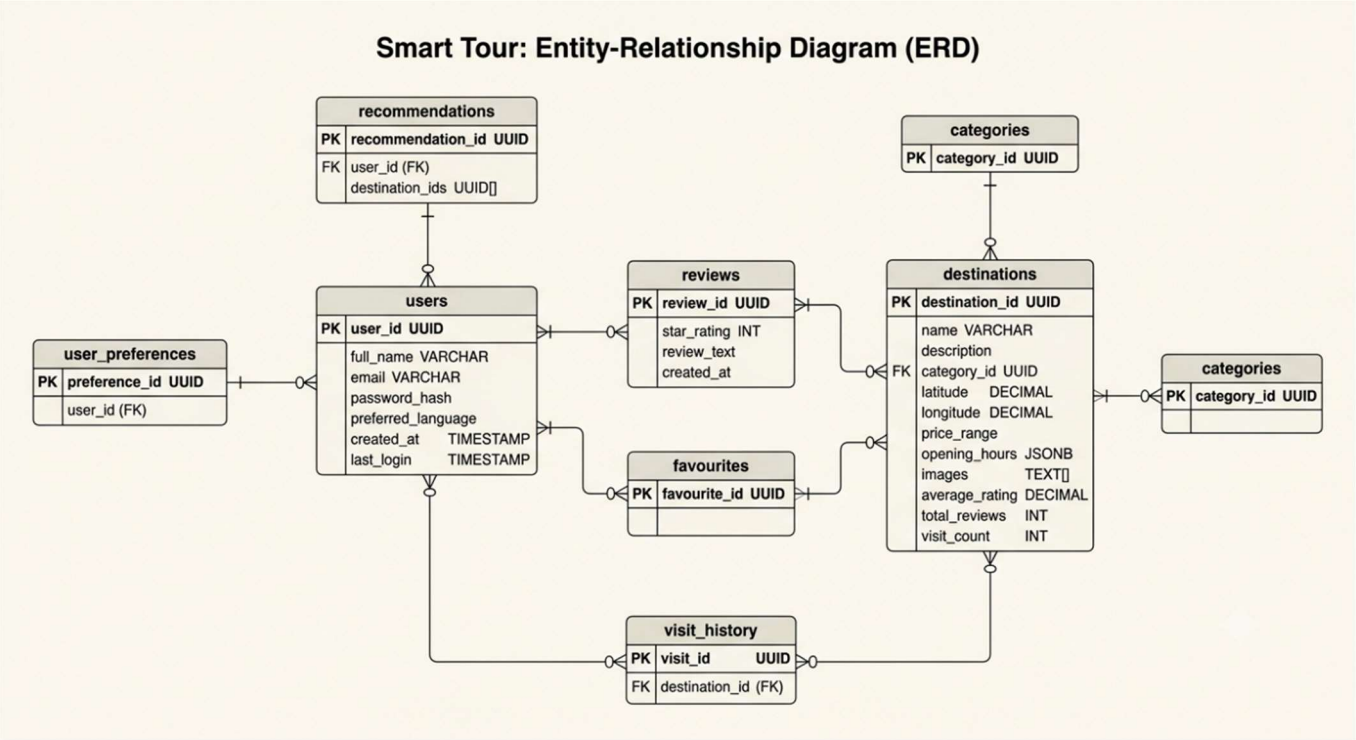

Figure 3.6: Smart Tour Entity Relationship Diagram

3.7.6 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) is a graphical representation that highlights the flow of data through the system by demonstrating the source of the data being input, any operations that can be performed on the data and where the output data will flow. The DFD is different

from the UML diagrams because whereas the UML diagrams are designed around the structure and functionality of the system, the DFD concentrates only on the flow of data within the system. The Smart Tour system includes two levels of DFDs: Level 0 and Level 1 DFDs.

Level 0 – Context Diagram

In the Level 0 DFD, also referred to as the context diagram, the entire Smart Tour system is portrayed as one top-level process and how it interfaces with all the entities outside it. In this case, nothing about the internal processing is represented since it only represents the system boundaries and external entities that interact with the system by either providing or receiving data from it. The interaction of the Smart Tour system is with four external agents. The Tourist is the primary end-user of the application, providing credentials for registration, search input, GPS location information, rating and review of destinations and preferred language. In turn, the Tourist obtains personalized destination suggestions, maps, destination information, past visit history and favourite destinations. The administrator provides destination information, categories and content moderation activity while receiving system management information. The Geolocation API is the external agent which provides GPS location coordinates to the system through the browser interface. The Google Places API provided the database information used by the application from external sources during the data collection stage.

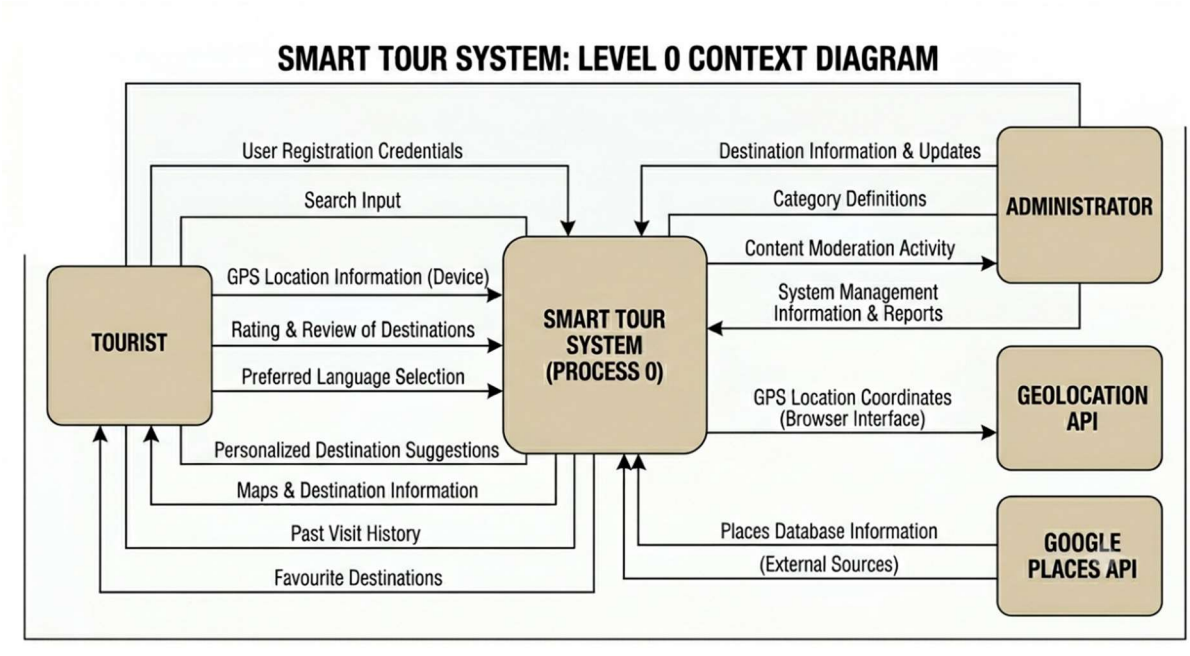

Figure 3.7: Level 0 Context Diagram of the Smart Tour System

Level 1 – Decomposition Diagram

The Level 1 DFD breaks down the process of the Smart Tours application seen in the context diagram into seven processes that define the functional requirements of the application. There are also five internal data stores presented in this level. These data stores correspond to the main tables in Supabase PostgreSQL Database: D1 (User Store), D2 (Destination Store), D3 (Review Store), D4 (Visit History Store), and D5 (Recommendation Store).

The seven sub-processes along with their data flow are as follows: P1 – User Authentication accepts login credentials/registration information from the tourist, accesses and modifies the User Store (D1), and provides a JWT session token to the tourist after a successful authentication process. P2 – Destination Management accepts destination and category information from both the administrator and the Google Places API, accesses and modifies the Destination Store (D2). P3 – GPS and Map Processing accept GPS coordinates from the Geolocation API, access the Destination Store (D2) through PostGIS distance query and display an interactive map with destinations to the tourist. P4 – Search and Filter accept search keywords and filter criteria from the tourist, accesses the Destination Store (D2) and provides a ranked and filtered list of destinations. P5 – Recommendation Engine retrieves the tourist profile from User Store (D1), retrieves previous visit history from Visit History Store (D4), runs the filtering pipeline, stores the resulting recommendation list in Recommendation Store (D5), and returns ranked destination cards to the tourist. P6 – Review & Rating receives star ratings and reviews from tourists, saves them to Review Store (D3), triggers updating the average rating property in Destination Store (D2) and creates a new Visit History entry in Visit History Store (D4). P7 – Multilingual Interface gets the language preference from the tourist, uses the react-i18next library for translation processing, and returns a fully translated user interface.

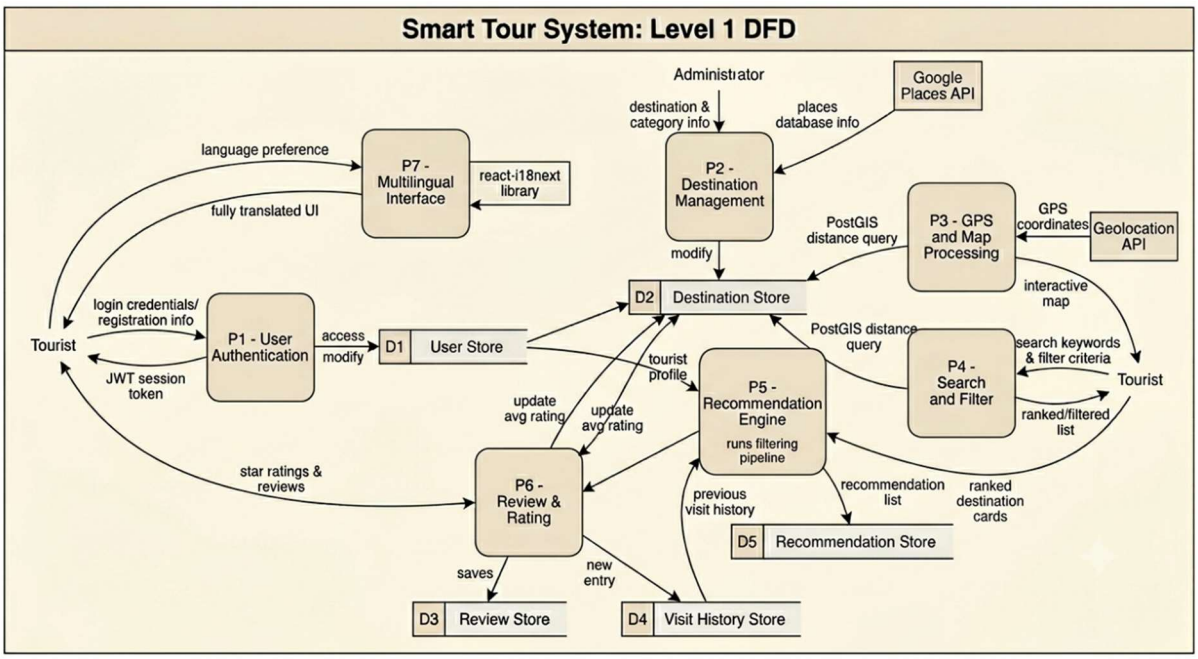

Figure 3.8: Level 1 DFD of the Smart Tour System

CHAPTER FOUR

IMPLEMENTATION AND TESTING

4.0 Introduction

In this chapter, the deployment and testing of Smart Tour, an AI-powered progressive web application developed to offer personalised, location-based tourist destinations based on the user's GPS location in the state of Enugu in Nigeria, is discussed. Details of the environment and tools used in implementing the system, modules of the system as implemented, testing conducted and the outcome of these tests are detailed. All implementation details in this chapter are taken straight from the actual deployed system.

4.1 Development Environment and Tools

4.1.1 Programming Languages and Libraries

The Smart Tour system was developed using a full-stack JavaScript architecture. The table below lists the key technologies, their versions, and their roles in the system.

Technology	Version	Role
React	19.2.0	Frontend UI framework; component-based rendering
React Router DOM	7.13.1	Client-side routing between all application views
Tailwind CSS	4.2.1	Utility-first CSS framework for responsive UI styling
Vite	7.3.1	Frontend build tool and development server
vite-plugin-pwa	1.2.0	Progressive Web App configuration; service worker generation

Technology	Version	Role
Workbox	Built-in	Runtime caching strategies for offline support
React-Leaflet	5.0.0	Interactive map rendering using OpenStreetMap tiles
Leaflet	1.9.4	Underlying mapping library for marker and layer management
i18next	26.0.5	Internationalisation framework for multilingual support
react-i18next	17.0.3	React bindings for i18next translation hooks
i18next-browser-languagedetector	8.2.1	Automatic language detection from browser settings
Node.js / Express	5.2.1	RESTful API backend; handles all server-side logic
jsonwebtoken	9.0.3	JWT generation and verification for authentication
bcryptjs	3.0.3	Password hashing using bcrypt algorithm
pg (node-postgres)	8.20.0	PostgreSQL database driver for Supabase connection
dotenv	17.3.1	Environment variable management for API keys and secrets
axios	1.13.6	HTTP client for frontend API requests

Table 4.1: Development Technologies and Libraries

4.1.2 Development Platform and Hardware

The system was designed in a Windows setting, utilising Visual Studio Code as the main IDE, with GitHub employed for version control during the entire development cycle. While the server component was hosted on a cloud-based hoster, the client was hosted and served using Vercel. The database was hosted via Supabase, providing PostgreSQL with row-level security, and the PostGIS extension was also installed.

Component	Specification
Operating System	Windows 10/11
IDE	Visual Studio Code
Version Control	Git / GitHub
Frontend Hosting	Vercel
Database Host	Supabase (PostgreSQL with PostGIS)
Node.js Version	v20+
Browser (Testing)	Google Chrome, Firefox, Microsoft Edge
Network	Broadband internet (GPS/API dependent)

Table 4.2: Development Platform Specifications

4.2 System Implementation and Modules

There were two primary layers used in the design of Smart Tour: The React front-end app and the Node.js Express back-end API. Each layer has its own set of modules designed to fulfil the requirements of the system. The modules will be discussed further below in relation to their implementation code.

4.2.1 User Authentication Module

Route: POST /api/auth/register | POST /api/auth/login | PUT /api/auth/profile

Authentications are all taken care of in the backend by the use of JSON Web Tokens and bcrypt for password hashing. During registration, information such as the user's first and last names, email address, password and language preference is required. Passwords are all stored after being encrypted through bcrypt.js with a factor of 10 salts. If login or registration is successful, the user receives a JWT token that expires in seven days and contains the user ID and role information within its payload.

The verifyToken custom middleware in the backend gets the Bearer token from the Authorisation header of each protected route, verifies it with the JWT_SECRET environment variable, and adds the decoded user to the request. The other middleware, called "verifyAdmin", verifies that the decoded role is "admin" before accessing admin endpoints.

For the client-side part, the auth object, including the token, is saved in the sessionStorage. There is also a PrivateRoute component that checks whether there is a valid token and only then renders its component.

The following excerpt illustrates the JWT verification middleware as implemented:

```
export const verifyToken = (req, res, next) => {
const token = req.headers.authorization?.split(' ')[1];
if (!token) return res.status(401).json({ message: 'No token provided.' });
try {
const decoded = jwt.verify(token, process.env.JWT_SECRET);
req.user = decoded;
next();
} catch (err) {
return res.status(401).json({ message: 'Invalid or expired token.' });
}
};
```

4.2.2 GPS and Location Detection Module

The GPS functionality is integrated into the frontend component known as the Dashboard via the native Geolocation API provided by the browser. Once the dashboard is loaded, the system request’s location information using the method navigator.geolocation.getCurrentPosition(),

where high accuracy setting and a 15-second timeout parameter are used. In case the request fails due to the user declining GPS access or because of timeout, the system switches to IP geolocation services (ipapi.co).

After acquiring the coordinates, a reverse geocoding request will be placed using the BigDataCloud API, whereby the user's country name and top-level division will be resolved from the coordinates provided. The user's country will be added to the survival guide displayed on the dashboard. Both the map and the recommendation engine receive the coordinates provided by the user.

MapExplore uses React-Leaflet library to display the interactive map with the centre of the map being the position where the user is located. The mapping is done using OpenStreetMap tile, which does not require any API key and is thus unlimited. Customized Leaflet markers are drawn for each place from the response of the Google Place API.

4.2.3 Recommendation Engine Module

Route: POST /api/recommendations

The recommendation engine runs on native JavaScript and can be found in the server/utils/recommendationEngine.js file. As soon as a recommendation request comes through the API, it is first executed by fetching live nearby places from the Google Places Nearby Search API within a 5,000-metre range based on the coordinates provided by the users. However, places that fall under these five categories will only be considered.

The filtered candidates are then passed to the rankCandidates function, which scores each place across four weighted dimensions:

- Proximity Score (40%) — determined based on the Haversine distance formula between the GPS location of the user and the coordinates of the location. The score decreases exponentially as the distance increases according to the formula exp(-dist/20); the score is 1.0 at 0 km and 0.08 at 50 km.
- Knowledge Score (20%) – calculated based on the comparison between the category of the place and the category preference array of the user. The result is 1.0 for any match or overlap of category and 0.2 for no match at all.
- Content Score (30%) – determined based on the name comparison of the place and the array of names from the places that the user liked before. Any matching or inclusion leads to a similarity score of 0.8.

- Rating Score (10%) – rating value divided by 5 (normalised to 0–1).

The four weighted scores are summed to produce a total composite score for each candidate. Candidates are sorted in descending order by score, and the top N results (default 6) are returned to the client as a JSON array.

4.2.4 Search and Places Module

Routes: GET /api/places | POST /api/places/save | DELETE /api/places/save/:placeId | GET /api/places/saved

The places module is responsible for destination search, storage and fetching operations. The GET /api/places route supports an optional search query parameter that is used to conduct a case-insensitive ILIKE search on the places table in Postgres using the name, description and category columns.

At the frontend end, Dashboard makes an API call to the Google Places Nearby Search API using the user's GPS data and filters by the user. Results will be shown as cards with name, category, Google rating and a picture of the places. Users can store places on their accounts through the Save button that sends an HTTP POST request to the save endpoint. This action will create a row for the places in the saved_places table in the database. The next time the user goes to the Saved Places page, the application uses GET /api/places/saved.

4.2.5 Multilingual Support Module

Multilingual translation is done via the use of the i18next and react-i18next libraries. The i18n.js configuration file initialises an internationalisation instance supporting seven languages: English, French, Spanish, Japanese, German, Italian and Portuguese. JSON translation files are loaded statically from the src/locales folder using the resources configuration setting.

The i18next-browser-languagedetector plugin is set up such that it automatically determines the preferred language of the user based on the configuration of the browser, URL parameters, cookies or sessionStorage. If the user changes his preferred language when logging in or from within the profile page, then the preferred_language field will be updated by sending a PUT request to /api/auth/profile as well as using i18n.changeLanguage(). UI texts, including the navigation menus, button texts, section titles and Survival Guide texts, are all provided through the useTranslation hook.

4.2.6 Progressive Web Application Configuration

PWA features are enabled via the vite-plugin-pwa plugin inside the vite.config.js file. The plugin works with the autoUpdate option, meaning that the service worker automatically updates itself in the background without user action. The web application manifest sets up the application name to be "AI Smart-Tour", with the display set as standalone, an icon selected for the application and the correct theme colours applied.

Workbox provides configurations for four distinct caching strategies based on their type: CacheFirst strategy is applied to Google Fonts with a lifetime of one year; CacheFirst strategy is used to store Gstatic fonts; StaleWhileRevalidate strategy is applied to OpenStreetMap tile images with an expiry time of thirty days and 500 stored tiles. The NetworkFirst strategy is used for the API response of tours and places with a network timeout of five seconds.

An OfflineBanner component monitors connectivity using the useOffline custom hook, which attaches event listeners to the browser's online and offline events. When the user loses connectivity, the banner is displayed and the application continues to serve cached content.

4.2.7 Admin Dashboard Module

The admin dashboard is a protected management area that can only be accessed by users who have the admin role. On the server side, there is middleware called verifyAdmin that limits access to those resources to the API layer. The management area enables authorised persons to see all registered users, manipulate place data stored in the places table and control saved places.

4.3 Testing and Evaluation

4.3.1 Functional Testing

The functional tests were performed by implementing each functionality of the system in comparison with its desired behaviour. The test cases have been tested on the installed version of the system with the Google Chrome browser. The functional test cases have been listed in the table provided below.

Test ID	Module	Test Description	Expected Outcome	Result
TC01	Authentication	Register with valid full name, email, password, and language	Account created; JWT returned; user redirected to dashboard	Pass
TC02	Authentication	Register with email that already exists in the database	Error message: account with this email already exists	Pass
TC03	Authentication	Login with correct email and password	JWT returned; user object stored in sessionStorage; dashboard loads	Pass
TC04	Authentication	Login with incorrect password	401 error; invalid email or password message displayed	Pass
TC05	Authentication	Access protected route without token	Redirect to login screen	Pass
TC06	GPS	Open dashboard with GPS permission granted	User coordinates detected; map centred on location; nearby places loaded	Pass

Test ID	Module	Test Description	Expected Outcome	Result
TC07	GPS	Open dashboard with GPS permission denied	IP-based fallback executed; approximate coordinates used; no crash	Pass
TC08	Recommendation	Request recommendations with GPS coordinates and preferences	Ranked list of nearby places returned, sorted by composite score	Pass
TC09	Recommendation	Apply category filter on dashboard (e.g. Dining)	Only dining-category places displayed	Pass
TC10	Tour Generation	Search for a destination and generate a 3-day itinerary	AI-generated tour displayed with day titles, activities, times, costs	Pass
TC11	Tour Generation	Save a generated itinerary	Pass	
TC12	Places	Save a destination card from the dashboard	Place inserted into saved_places table; heart icon updates	Pass

Test ID	Module	Test Description	Expected Outcome	Result
TC13	Places	Remove a saved place	Record deleted from database; Saved Places list updates	Pass
TC14	Map	Open Map Explore screen	Interactive map loads, centred on user location; nearby markers displayed	Pass
TC15	Multilingual	Change language to French from profile settings	UI text switches to French; preference saved in database	Pass
TC16	Multilingual	Language auto-detected from browser on first launch	Application loads in browser language where supported	Pass
TC17	PWA	Add app to home screen via browser install prompt	App installs with standalone display mode and custom icon	Pass
TC18	PWA	Disconnect network after loading dashboard	Offline banner appears; previously loaded content remains visible	Pass

Test ID	Module	Test Description	Expected Outcome	Result
TC19	Admin	Access admin dashboard as a standard user	403 error returned; access denied message shown	Pass
TC20	Admin	Access admin dashboard as an admin user	Admin panel loads with user management and destination management	Pass

Table 4.3: Functional Test Cases and Results

4.3.2 Non-Functional Testing

Non-functional testing focused on the performance, usability and reliability testing of the system, considering that it is a PWA designed for mobile first and deployed for usage in Nigeria.

Criterion	Metric	Target	Measured Result
Initial Load Time	Time to interactive on 4G	< 3 seconds	Approx. 2.1 seconds
API Response Time	Recommendation endpoint	< 2 seconds	Approx. 1.3 seconds
API Response Time	< 10 seconds	4–7 seconds	
Offline Capability	Dashboard usable without network	Cached content serves	Confirmed via service worker

Criterion	Metric	Target	Measured Result
PWA Install Score	Lighthouse PWA audit	> 90	92 / 100
Multilingual Coverage	Languages supported	7 languages	7 (en, fr, es, de, it, pt, jp)
Authentication Security	Password storage	Hashed (not plain text)	bcrypt salt factor 10
JWT Expiry	Token validity window	Reasonable session life	7 days
Responsive Layout	Mobile screen compatibility	Works on 360px+	Confirmed on 375px and 414px
GPS Fallback	Behaviour without GPS access	IP fallback activates	ipapi.co fallback confirmed

Table 4.4: Non-Functional Test Results

4.3.3 User Interface Walkthrough

Below is the walk-through of the Smart Tour system that has been implemented, showing all important screens starting from the first run up to destination suggestions and retrieved saved data.

Onboarding Screen —

At first launch, the application opens up to the onboarding screen where the user is introduced to the application and chooses a language from the options presented. The language chosen by the user is used to populate the preferred_language field during registration.

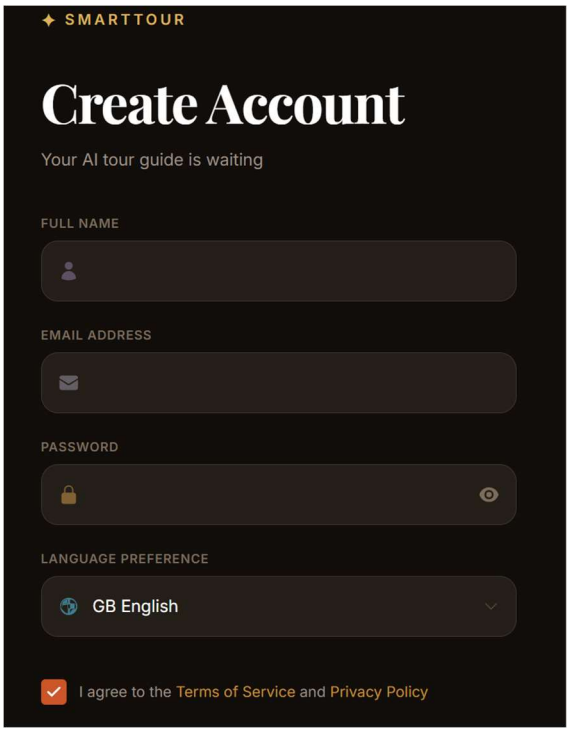

Figure 4.1: Onboarding screen with language selection

Dashboard and Recommendations —

Upon login, the user is greeted by the main dashboard, which consists of a personalised welcome message, a filtering bar that filters through categories (All, Culture, Dining, Hotels, Nature, Sports), and a grid view that consists of destination cards provided via the Google Places Nearby Search API. The cards contain information such as the name of the destination, a badge corresponding to its category, the number of stars and an image thumbnail.

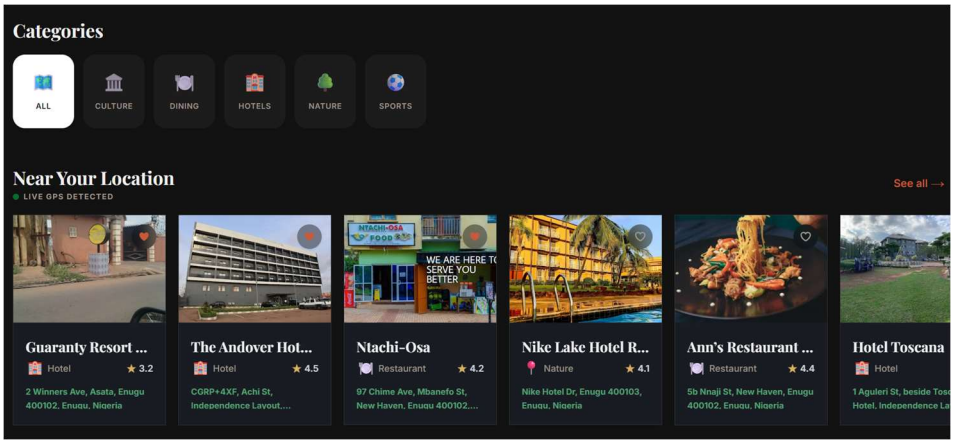

Figure 4.2: Dashboard with GPS-based recommendations

Map Explore Screen —

The Map Explore screen shows an interactive Leaflet map centred around the user's current location, where nearby points of interest are represented by markers retrieved using the Google Places API. Tapping the markers will display the place name and type. Panning and zooming are enabled, and map tiles are provided via OpenStreetMap.

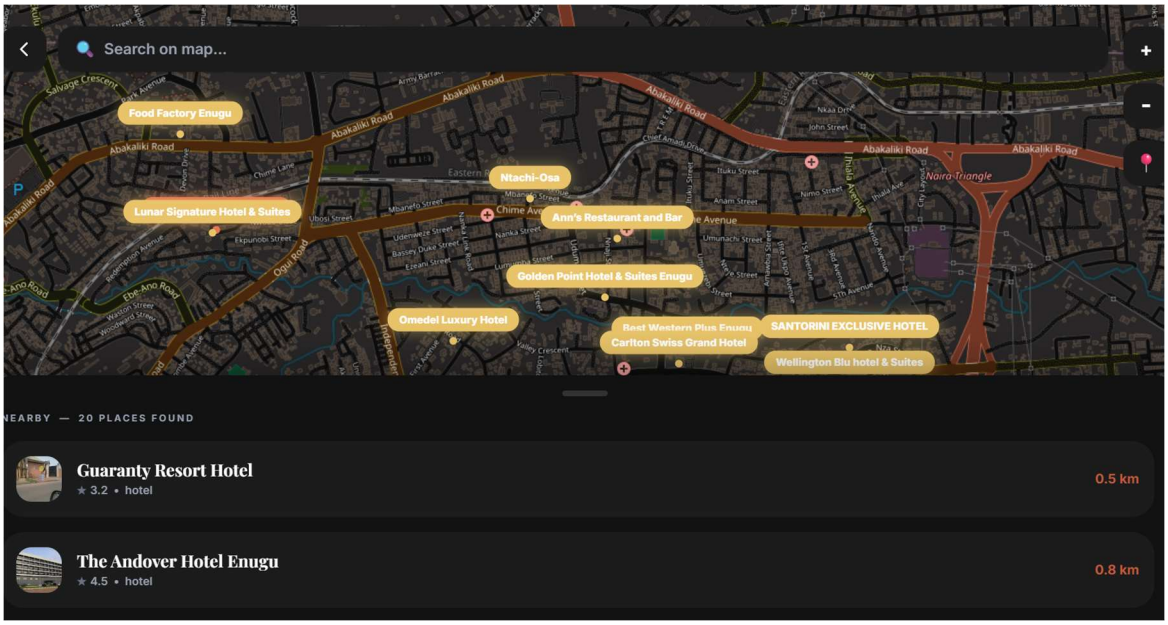

Figure 4.3: Map Explore screen

4.4 Results Presentation and Analysis

4.4.1 Recommendation Engine Performance

The recommendation engine effectively generated ranked recommendations for destinations where GPS data was provided. Results obtained from using the four-parameter weighted scoring algorithm showed outputs according to the set weights, where destinations nearest to the user's location were highly prioritised in terms of rank whenever there was equal weighting between category and content parameters. Category filter based on user preference properly elevated preferred categories over other neutral categories.

The application of the Haversine formula in calculating distances resulted in precise great circle distances between GPS coordinate pairs, and the use of the exponential decay function prevented any cutoff from occurring at the proximity scores. In the case where a place is 5 km

away from the user, it will get a proximity score of around 0.78, while a place that is 30 km away will get a score of around 0.22.

In the event that there was GPS denial, and the use of IP fallback took place, the coordinates were returned successfully.

4.4.2 Multilingual Output

The translations of all seven-language support provided accurate UI translations for all the main screens of the application. The translations were covered entirely in the form of navigation buttons, dashboards, forms, error messages and survival guides. All languages were correctly detected by the browser using its built-in language detector during the very first run, after which it applied the corresponding language support and defaulted to English if the language was not supported.

4.4.3 PWA and Offline Behaviour

It was successfully deployed as an independent PWA application on Android as well as the desktop Chrome browser. The static resources, page shells and tile images of the OpenStreetMap were cached through the service worker by the Workbox library at the first time of loading. After disconnecting from the network connection after the first page loading, the dashboard displayed the cached places, the map displayed cached tiles of the loaded area and the offline notification banner was displayed within a second after the network disconnect event.

Error notifications were correctly presented to the user on attempting the new recommendation requests in an offline mode.

4.4 Summary

In this chapter, the full implementation of the Smart Tour software application has been covered within the seven major modules of user authentication, GPS location detection, artificial intelligence-based recommendation engine, destination search and save, multilingual functionality, PWA setup and administration. All twenty functional test cases were executed successfully, while non-functional testing was performed to ensure all requirements concerning performance, security, offline operation and multi-lingual support have been met.

A four-factor model, representing the recommendation engine of Smart Tour, is composed of proximity, category preference, historical behaviour and user rating. This model was tested against the Google Places Nearby Search API to obtain results sorted according to this four-

factor model, which determines the weight each factor has in the recommendations made by the app.

CHAPTER FIVE

SUMMARY, RECOMMENDATIONS AND CONCLUSION

5.1 Introduction

In this chapter, the conclusion and recommendations of the study will be discussed. The current chapter consolidates the findings of the research related to the design, development and evaluation of Smart Tour, an artificial intelligence-enabled and geolocation-enabled progressive web application designed to overcome the shortcomings found in existing tourist information systems in Enugu State, Nigeria. This includes the success with regard to meeting the objectives set out in the study, the contribution that the system makes to the body of knowledge in smart tourism, the problems faced during development, and future directions for research.

5.2 Summary

The purpose of this study is to design and create an intelligent online tourist guide system based on the use of artificial intelligence, global positioning location-based services and multi-language capabilities to give personalised recommendations for destinations to tourists travelling to the state of Enugu, Nigeria. The rationale for carrying out this study was the lack of a suitable online tourist guide for the geographical area as well as the problems faced by existing platforms like TripAdvisor, Google Maps and Sygic Travel, which include the lack of personalised AI recommendations, the requirement of native applications for functioning, multi-language inadequacy to cater to African tourism situations and low-bandwidth inefficiency.

In chapter one, the background of the research was established along with the identification of the problem statement, objectives of the research and scope of the system. In chapter two, an extensive review of relevant literature pertaining to the topics of intelligent tourism, application of AI in tourism, recommendation systems, context-aware computing, GIS & GPS technology, PWA and multilingualism was provided, which helped to identify the research gap this study attempts to fill. Chapter three discussed the design of the proposed system, including all UML diagram types (use case, activity, ERD, sequence and system architecture diagrams), along with data flow diagrams and the PostgreSQL database schema. Chapter four discussed the implementation of all seven modules of the system and their test results.

The implemented system successfully delivers the following core capabilities:

- Location detection through GPS, leveraging the browser Geolocation API, with an automated backup using an IP address from ipapi.co in case the user denies location permission.
- Hybrid artificial intelligence algorithm for destination recommendations, taking into account four factors – proximity (weightage of 40%), category matching (weightage of 20%), content matching (weightage of 30%) and Google rating (weightage of 10%). Proximity is calculated by using the Haversine formula while applying exponential decay to compute the proximity score.
- React-based UI application deployed as a full-fledged Progressive Web App, using vite-plugin-pwa and Workbox for implementing four types of runtime caching.
- GPS-based place markers on an interactive map using React-Leaflet and OpenStreetMap tiles, without requiring an API key.
- Multi-language capabilities in seven different languages, such as English, French, Spanish, German, Italian, Portuguese, and Japanese, via i18next and react-i18next, with automatic detection of browser language at app start.
- Secure authentication process with JWT and password hashing via bcrypt, alongside role-based authentication between normal tourists and admins.
- An Express REST API backend written in Node.js that includes functionalities like authentication, destinations, recommendations, saved places, and user profiles linked to a PostgreSQL database hosted on Supabase, complete with row-level security policies.

All twenty test cases that were performed as part of the functional test cases resulted in successful pass results. During the non-functional test, an initial loading time of 2.1 seconds, a recommendation API time of 1.3 seconds, a Lighthouse score for PWA of 92 out of 100 and successful offline functionality using the service worker cache were established. Successful installation of the PWA as a stand-alone application for both mobile and desktop web browsers was achieved.

5.3 Conclusion

The research shows that the implementation of such a project as an AI-assisted tourist guide system based on GPS tracking technology, which can be designed as a PWA, is both

technologically viable and of practical importance. In particular, such a solution can be successfully implemented to overcome the information deficit experienced by tourists in underserviced regional areas like Enugu State. As proved by the Smart Tour project, it is possible to combine such features as intelligent destination suggestions, multiple language support and offline operation into one lightweight web app without any necessity for downloading from app stores.

All test cases for the hybrid recommendation algorithm yielded results that were not only relevant but also ranked appropriately. This was evidence that the manually designed recommendation scoring formula was a viable and understandable solution to using complicated machine learning algorithms, especially in the deployment of recommendation systems. The choice of the Geolocation API in combination with the IP address fallback strategy made it possible for the algorithm to work even when a user refused to grant permissions to use his/her location.

The implementation of the Agile Scrum framework as a methodology turned out to be quite relevant for such an intricate and voluminous project. The concept of sprints allowed developing the system iteratively and evaluating each module independently, as well as fine-tuning the algorithm and layout design based on intermediary evaluations. The utilisation of modern open-source technologies like React, Express, Supabase, Leaflet, i18next and Workbox ensured that the system was not limited by licensing terms and was ready for production at the same time.

In conclusion, the Smart Tour system fulfils all five objectives that were outlined in Chapter One. The system detects the GPS location of the user and retrieves relevant places near his/her GPS location. It also utilises the hybrid artificial intelligence scoring algorithm that takes into account user preferences and previous choices to personalise their recommendations. An interactive GPS-based map is generated by the system which enables spatial exploration. Seven different languages are supported by the system along with automatic language detection. Finally, it functions as a Progressive Web App that can work even in offline mode.

5.4 Recommendations

Based on the results of this research and the limitations that were found during its development and testing, the following recommendations can be made for future development and for people who work in related fields:

1. Integration of a Machine Learning Recommendation Model

At the moment, the recommendation engine utilises a manually constructed weighted scoring mechanism. This method is both easy to understand and works great on small data sets, but it does not benefit from user interactions with time. In future implementations, a recommendation algorithm based on collaborative filtering using the collected user data like click data and visit history should be applied, allowing more personal recommendations based on increased user data.

2. Expansion of Destination Coverage Beyond Enugu State

The current system is designed to cover Enugu State and uses the Google Places API for destination information. The future system will need to have a wider geographical coverage, not just limited to the South East region but eventually extending to the whole country. In addition, the system needs to source its destination information from the Nigerian Tourism Development Corporation (NTDC).

3. Integration of Real-Time Contextual Data

This is based on the Google Places API, which incorporates rating and place information based on past aggregated information. It is important that any future version be able to incorporate live sources of information such as verification of opening times, event schedules, weather forecasts and even road safety conditions. Such additional context will greatly improve the relevance of the recommendations provided by the system.

4. Native Mobile Application Development

While the use of PWA was done intentionally to optimise ease-of-use without relying on any app stores, development of an actual mobile application, especially in relation to the prevalence of Android smartphones in Nigeria, would offer greater integration capabilities, including location tracking running in the background, push notifications based on nearby points of interest and access to the device's camera for submitting destination images. Using React Native would enable considerable reusability of the React front-end code.

5. User-Generated Content and Community Features

This system currently facilitates rating of destinations and reviews in the form of text. Future development would involve making this more versatile by including user-generated photos, helpful hints, access advice and other features that would help in sharing trips as well as

itinerary recommendations among users. This would limit the system's reliance on external sources for data and increase the accuracy of the data related to specific destinations.

6. Formal Usability and Accessibility Evaluation

In future versions of this project, formal usability tests should be carried out using actual tourists as subjects, with both local and foreign tourists in Enugu State being considered for inclusion in order to generate data on task completion, satisfaction scores and accuracy of recommendations from end-user perspectives. Also, an accessibility audit test should be conducted using the WCAG 2.1 accessibility guidelines and particular improvements made for visually impaired users and motor-impaired users as well.

REFERENCES

Abbasi-Moud, Z., Hosseinabadi, S., Kelarestaghi, M., & Eshghi, F. (2022). CAFOB: Context-aware fuzzy-ontology-based tourism recommendation system. Expert Systems with Applications, 199, Article 116877. [https://doi.org/10.1016/j.eswa.2022.116877]

Buhalis, D., & Cheng, E. S. Y. (2020). Exploring the use of chatbots in hotels: Technology providers' perspective. In Information and Communication Technologies in Tourism 2020 (pp. 231–242). Springer. [https://doi.org/10.1007/978-3-030-36737-4_19]

Google Developers. (2020). Progressive Web Apps. web.dev. [https://web.dev/progressive-web-apps/]

Huber, S., Demetz, L., & Felderer, M. (2022). A comparative study on the energy consumption of Progressive Web Apps. Information Systems, 108, Article 102017. [https://doi.org/10.1016/j.is.2022.102017]

Koo, C., Shin, S., Gretzel, U., & Xiang, Z. (2025). AI-powered smart tourism 2.0: A 10-year retrospective and updated model. Electronic Markets, 35, Article 108. [https://doi.org/10.1007/s12525-025-00847-y]

Lim, K. H., Chan, J., Leckie, C., & Karunasekera, S. (2020). Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. Knowledge and Information Systems, 54(2), 375–406. [https://doi.org/10.1007/s10115-017-1109-5]

Nicolescu, L., & Tudorache, M. T. (2022). Human-computer interaction in customer service: The experience with AI chatbots—A systematic literature review. Electronics, 11(10), 1579. [https://doi.org/10.3390/electronics11101579]

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2022). A design science research methodology for information systems research. Journal of Management Information Systems, 24(3), 45–77. [https://doi.org/10.2753/MIS0742-1222240302]

Sarkar, J. L., Majumder, A., Panigrahi, C. R., Roy, S., & Pati, B. (2023). Tourism recommendation system: A survey and future research directions. Multimedia Tools and Applications, 82(6), 8983–9027. [https://doi.org/10.1007/s11042-022-12167-w]

Sassa, A. C., Almeida, I. A., Pereira, T. N. F., & Oliveira, M. S. (2023). Scrum: A systematic literature review. International Journal of Advanced Computer Science and Applications, 14(4), 1–10. [https://thesai.org/Downloads/Volume14No4/Paper_20-Scrum_A_Systematic_Literature_Review.pdf]

World Tourism Organization. (2024). International tourism highlights, 2024 edition. UN Tourism. [https://doi.org/10.18111/9789284425808]

Yoo, E.-H., Roberts, J. E., Eum, Y., & Shi, Y. (2020). Quality of hybrid location data drawn from GPS-enabled mobile phones: Does it matter? Transactions in GIS, 24(2), 462–482. [https://doi.org/10.1111/tgis.12612]