Design of an Intelligent Location-Aware Architecture for Mobile Computing Environments

Santipong Thaiprayoon
Chair of Communication Networks
FernUniversität in Hagen
Hagen, Germany
santipong.thaiprayoon@fernuni-hagen.de

Herwig Unger
Chair of Communication Networks
FernUniversität in Hagen
Hagen, Germany
herwig.unger@fernuni-hagen.de

***Abstract*—Mobile devices have become ubiquitous in daily lives, especially for people who use their smartphones throughout the day while visiting different places in the real world. Providing valuable services and information that are tailored to a specific area or environment can make user interactions more seamless, personalized, and convenient. This paper proposes an intelligent location-aware architecture for mobile computing environments that interacts with individual mobile users by automatically providing appropriate local services, information, and personalized recommendations based on their current location and profiles while tracking their position and movement. Experimental results show that the proposed architecture is intelligent, scalable, reliable, and can be efficiently deployed in various real-world environments.**

***Keywords*—Location awareness; Personalized recommendations; User interactions; Mobile computing environments**

I. Introduction

The emergence of mobile technologies and advancements in wireless networking technologies have significantly transformed the natural way for users to access and search for information, interact with digital services, and navigate surrounding environments anytime and anywhere [1]–[3]. The convergence of these technologies has shifted the trend towards Location-Aware Systems (LASs) that provide users with personalized and contextually relevant information based on their current geographical position within mobile computing environments [4] [5].

The development of location-aware systems typically resides on a backend server as a web application that operates using a client-server architecture [6], enabling interaction, communication, and data exchange between clients and servers. It consists of various technologies and components to provide convenient services and information for users. In this architecture, a client, a computer, or other device uses a web browser to request data or services through the Internet, while a server then performs the request, generates the result of the requested task, and serves the task results to the client, which are then displayed in the web browser.

Based on their previous research [7]–[9], Santipong et al. introduced a conceptual framework for the future web that helped local mobile users directly access and contribute web content and services on the local and global webs without needing centralized servers through their mobile devices. Moreover, it automatically provided mobile users with tailored local information and personalized services in environmental contexts, which could address the issues of information overload, centralization, and data privacy. However, it needs a complete description of the communication and interaction with local services hosted on a local sandbox server within a given environment, enhancing user experiences and enabling seamless integration with real-world environments.

To bridge the gap, this paper proposes an intelligent location-aware architecture for mobile computing environments. The proposed architecture aims to enhance the overall user experience by providing personalized and relevant services, information, and suggestions tailored to their needs and preferences based on their current location. The use of location-based services and user profiles ensures that individual mobile users receive valuable real-time information about nearby businesses, events or attractions. Moreover, through push or pull mechanisms, mobile users can receive potential interests related to an environment so that they can actively engage with their surroundings and discover new opportunities that match their interests. Furthermore, the proposed architecture is designed as a client-side mobile application that gives mobile users more control and ownership over their personal data, with a focus on maintaining privacy and allowing mobile users to fully manage their data.

The contributions of this paper are summarized as follows:

- This paper provides valuable insights into the challenges and opportunities of designing an intelligent location-aware architecture in mobile computing environments based on the current location and user profiles.
- Extensive experiments are conducted to validate the robustness and effectiveness of the proposed architecture.

The structure of the rest of this paper is as follows: In Section II, a literature review and related work are conducted. In Section III, an architecture is proposed. Experimental details are presented in Section IV. Section V contains a discussion of experimental results. Section VI gives an example of a use-case scenario. Section VII presents opportunities and challenges. Finally, a conclusion and suggestions for future work are presented.

II. Literature Review and Related Work

In the last decade, various research studies have explored several application areas for location-aware systems. For example, Damianos et al. [10] focused on developing context-aware recommendation systems within intelligent location-aware platforms. Their study demonstrated that by considering user location and context, recommendation algorithms could provide highly personalized suggestions for nearby restaurants, shops, and attractions. Importantly, integrating machine learning techniques enabled the continuous refinement of recommendations based on user feedback and preferences. Building on this, Knijnenburg et al. [11] tackled the security and privacy issues inherent in location-aware platforms. Their research proposed a secure, privacy-preserving framework that robustly protected sensitive user location data against unauthorized access. Lastly, Shini et al. [12] presented a context-aware recommendation system for mobile users that leveraged location data and user preferences to provide personalized suggestions for nearby points of interest. Their research showed the capacity of intelligent location-aware platforms to enhance user experiences and decision-making processes.

Despite these significant advancements, there still needs to be a gap in designing a comprehensive location-aware system for mobile computing environments. Existing solutions regularly face challenges in accuracy, power consumption, privacy protection, scalability, and user experience. Therefore, the purpose of this paper is to address these limitations by designing an intelligent location-aware architecture that is optimized for mobile computing environments.

III. Architecture

This section describes the design of an intelligent, location-aware architecture for mobile computing environments. The main idea is to automatically provide individual mobile users in different environments with specific local services, information, and recommendations based on their current location and profile that meet the needs and preferences of the right users at the right time in an appropriate environment [13]. The proposed architecture is a computing model that divides tasks between a sandbox server and a mobile device, allowing mobile users to directly and securely access and contribute information and services through their mobile devices connected to a local Wi-Fi network. The sandbox server is responsible for hosting, managing, executing, and contributing services and information to the mobile device. The mobile device provides a user interface through which a mobile user can initiate requests for content or services and display results from the sandbox server. On the other hand, the sandbox server allows service providers to participate by developing their own services and applications, which they can then upload to the sandbox server. By uploading these services and applications to the sandbox server, service providers can extend the functionality and capabilities available to mobile users. This enables a dynamic and customizable user experience and the ability to access a wider variety of content and services through mobile devices. An architectural overview is given in Figure 1.

The proposed architecture also provides a solution to privacy and data protection issues by independently separating personal data stored on mobile devices from local services on a sandbox server. Each mobile user can control their own data through their mobile device, enabling direct interaction between mobile users and their local server without intermediaries. Additionally, mobile users can grant or revoke access to their personal data as needed and authorize external services, applications, or users to access it.

The proposed architecture consists of two main components: (1) a mobile device and (2) a sandbox server. This architecture plays important roles in serving a specific purpose, seamless communication, and engagement within a specific environment. Each part is explained, which describes how the sandbox server shares services and information and interacts with mobile users.

A. A Mobile Device

A mobile device includes the hardware and software components necessary for running applications. This includes the processor, memory, operating system, and any additional features or sensors present on the device. This mobile device is designed to support the execution of various applications while providing a user-friendly interface for interacting with them. Additionally, it ensures that the mobile device is capable of securely connecting to a sandbox server and transferring data between the two entities. Figure 2 shows a data flow diagram representing the flow of data through several processes between a mobile device and a mobile user.

Mobile users obtain a set of services and menus, depending on where they are, represented by local workflows through dialogues. These local workflows allow mobile users to easily navigate through various menus and access the services they need based on their location. As a whole, a mobile device can be divided into four primary layers:

1) Presentation Layer: This layer runs on a front-end device as a client-side system. A mobile user and a sandbox server interact and communicate primarily through a Graphical User Interface (GUI) with a compatible mobile application. A GUI consists of user interfaces and graphical elements, such as icons, buttons, and menus that allow mobile users to interact with them on dialogues. These dialogues are a communication process for exchanging data between a mobile user and a sandbox server that receives user input and displays information in response to user actions with dynamically updated content from the sandbox server, making them come alive with interactivity and adaptability. In a GUI, the visuals displayed in the user interface convey information relevant to the user and actions they can take. Users can usually interact with GUI elements by tapping a touch screen.

2) Business Layer: The layer is responsible for processing and managing the data received from the user interface. It handles the logic and rules of the application, ensuring that data is validated and processed correctly. Additionally, the business layer communicates with the data layer to retrieve and update information from databases. This layer plays a crucial

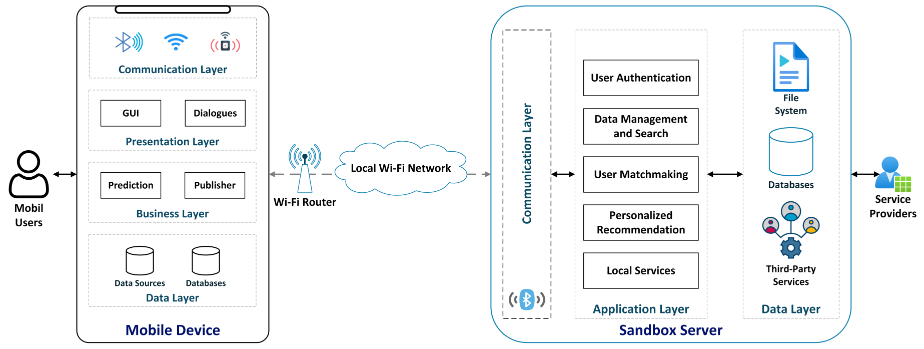

Figure 1. An architectural overview of the communication among a mobile device, a sandbox server, and service providers via a local Wi-Fi network

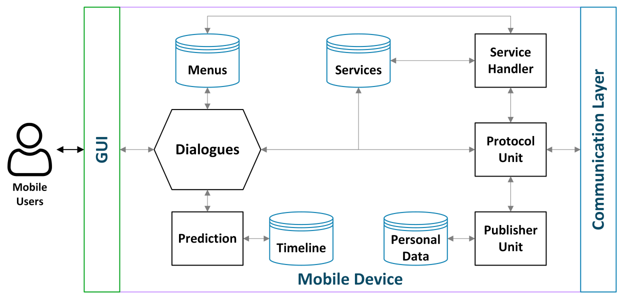

Figure 2. A data flow diagram of a mobile device

role in ensuring that the application functions properly and delivers the desired functionality to mobile users.

a) Service Handler: This is an instance that is responsible for downloading and uploading services and menus from local databases on a sandbox server. This instance ensures that the downloaded services and information are up-to-date by regularly checking for updates on the sandbox server. It also handles any errors or issues that may happen during the downloading or uploading process to ensure a smooth and uninterrupted flow of data.

b) Prediction Unit: The process attempts to automatically predict what a mobile user is doing next by learning sequential patterns from their local timeline or calendar. This is done using the Markov chain technique, which is a probabilistic model that has low computational resource requirements and can rapidly adapt to changing user behavior. This helps to optimize the overall efficiency and effectiveness of mobile applications and services. Additionally, it can be used to determine the possible next place or environment, which can help mobile users make better decisions on a daily basis.

c) Publisher Unit: This is responsible for automatically constructing a research-related profile with details that highlight their work, interests, and achievements from public academic databases. This profiling also includes extracting key information, such as authors, affiliations, abstracts, and keywords. This unit could be web crawlers or Application Programming Interfaces (APIs) to harvest the bibliographic data of a mobile user and store it in the personal database for further processing and analysis. It can then extract and share the required information with a sandbox server. For example, suppose mobile users have access to a conference. Then the publisher unit automatically shares their data and makes it accessible to the public on a sandbox server to make the conference location visible to other users.

d) Protocol Unit: This unit contains several custom protocols that are designed and implemented for particular applications or services to fit their specific needs. It is a set of rules, syntax, commands, and conventions that govern how different components within the application or service communicate and exchange information with each other.

3) Data Layer: This layer is responsible for storing and retrieving data from databases. It provides the necessary functionality for the business layer to access and manipulate data. This layer contains the following databases:

a) Personal Data: This database stores personal data that specifically identifies an individual mobile user, such as name, age, gender, education, contact details, expertise, biographies, and any other relevant data required for identification purposes. Additionally, it may include user preferences and settings to enhance the personalized recommendations.

b) Timeline Data: This database stores data related to the activities and interactions of mobile users, such as their browsing history, app usage, search queries, and social media interactions. This data is used to analyze user behavior, make personalized recommendations, provide next-item recommendations, and improve the overall user experience.

4) Communication Layer: This layer facilitates the secure exchange of data between a mobile device and a sandbox server. It acts as the first interface to manage the network connectivity of the mobile device, including incoming requests, validate them, and forward them to the appropriate components for processing, which enables a mobile user to directly and safely access information and services on the sandbox server through a local area network. There are a variety of communication technologies that enable a mobile device to establish connections with other devices or servers.

a) Bluetooth: Bluetooth is a short-range wireless transmission technology that is used for exchanging data between two different Bluetooth devices within a short distance using radio waves to communicate wirelessly. This Bluetooth technology is utilized for detecting the location information of mobile users, whether they are in radio coverage or not, and communicating with a sandbox server to request a Wi-Fi password before gaining access to resources or services on a local Wi-Fi network.

b) Wi-Fi: Wi-Fi is a wireless networking technology that allows mobile devices held by users to connect to a sandbox server. A Wi-Fi router assigns local IP addresses to connected devices, allowing them to communicate and exchange data with one another on a local Wi-Fi network. In addition, it uses the radio signal that the Wi-Fi router transmits to detect and identify the position and movements of a mobile user.

B. A Sandbox Server

A sandbox server, also referred to as a local server, runs on the backend system as a server-side component. It can be a physical or virtual machine operating on a powerful computer connected to a local area network within a specific environment. The sandbox server is responsible for receiving requests from mobile users. It processes these requests and returns the corresponding responses. A data flow diagram of a sandbox server is illustrated in Figure 3.

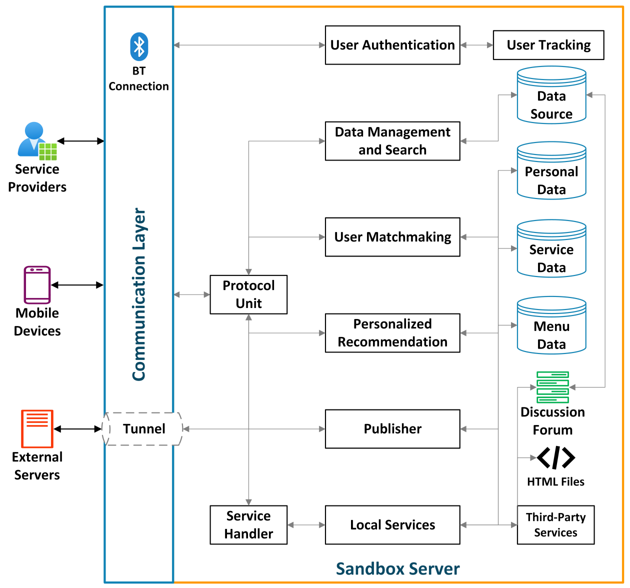

Figure 3. A data flow diagram of a sandbox server

A mobile device held by a user can request any services and information offered by a sandbox server and service providers. The sandbox server also creates a dynamic ecosystem where the service providers can continuously enhance and expand the range of information and services available on mobile devices. In addition, this sandbox server offers a direct connection to other local servers via a tunneling network, so that data transmitted across this network is encrypted and encapsulated within a secure communication channel. Overall, a sandbox server contains three fundamental layers:

1) Application Layer: This layer runs on a sandbox server that manages the logic and functionality of the proposed architecture. It processes user requests, performs necessary calculations or operations, and communicates with the data layer to retrieve or store information. Additionally, it handles user authentication and security measures to protect sensitive information. This layer facilitates the integration of local services, personalized recommendation services, and APIs, allowing for additional functionality and features to be incorporated into the proposed architecture. This application layer also communicates with the presentation layer to receive user input and deliver appropriate responses.

a) User Authentication: This is a security process to verify the identity of a mobile user attempting to access a local Wi-Fi network that prevents unauthorized users from accessing sensitive information. This can be accomplished through Wi-Fi authentication based on their presence or proximity. The proximity-based Wi-Fi authentication uses location-based BLE technology, enabling mobile users to authenticate using their mobile devices via Bluetooth connection. This means that it uses the distance between a mobile device and a sandbox server as a key measurement to verify the identity and determine that the mobile user is in close proximity to the sandbox server on a trusted mobile device. A mobile user needs to make a registration or check in by providing a phone number via Bluetooth connection to receive a Wi-Fi password according to the registered mobile phone via a Short Message Service (SMS). It makes sure that mobile users who access the local Wi-Fi network are authorized by the owner of the phone. By implementing this authentication process, the local Wi-Fi network can prevent unauthorized access and maintain a secure environment for its users. Additionally, this method allows the owner of the phone to have control over who can access their network, ensuring that only trusted individuals are granted access.

b) Data Management and Search: This offers efficient data management and search capabilities. Mobile users can easily find specific files or information from a database or data storage system when needed. Meanwhile, a sandbox server can manage and store their data.

c) User Matchmaking: The main objective of this application service is to produce a list of potential friends with similar interests, ranked according to a similarity score based on their personal information. The matchmaking algorithm compares a user profile with other profiles using a text similarity technique and suggests a suitable list of similar users. User interests, expertise, and biographies are combined to improve the accuracy of recommending similar users. The text similarity technique measures the similarity score between two pieces of personal information based on lexical and semantic similarity, covering both word level and context level using Natural Language Processing (NLP) techniques, word embeddings, and cosine similarity. Each user profile is cleaned up and transformed from unstructured textual data into an appreciable format. A word embedding technique encodes and converts textual data into a numeric format as a vector representation. Two vectors are compared using cosine similarity to extract semantically similar text from user profiles and return a similarity score.

d) Personalized Recommendation: This is personalized features that suggest relevant information or services based on the current situations and preferences of mobile users. The main task is to provide tailored recommendations, including related items and friends. By offering personalized suggestions, recommendation services aim to improve user satisfaction and engagement with the mobile application. The recommendation services utilize NLP and Machine Learning (ML) techniques to analyze and understand the contextual data of mobile users and generate accurate recommendations. They take into account factors, such as user preferences, browsing history, location history, purchase patterns, and social connections to deliver highly relevant suggestions.

e) Publisher Unit: This unit is responsible for managing scholarly profile information from mobile users so that they can share it with other individuals. In this way, they can increase the impact of their research by, for example, gaining opportunities to create connections, associations, and interactions with others.

f) Local Services: These services provide adaptation information and services to mobile users depending on their current location. This is responsible for hosting the services offered to be downloaded from mobile devices. Mobile users can move to a specific environment to get more services and information relating to the environment. It consists of two main categories: (1) standard services and (2) additional services. The standard services are a set of basic requirements that interact with users in order to meet their expectations of mobile users. Ideally, the standard services include discussion forums, chat, local web pages, announcements, games or puzzles, advertisements, campaigns, promotions, and useful information defined by service providers. The discussion forum enables users to write and share content and images with their community in real time. The additional services are specific services offered to mobile users according to their profiles or historical behaviors. These additional services can include alternative features by integrating with third-party applications. A local service can be conceptualized as a local navigational workflow comprising a series of sequential processes defining how a service flows or moves from one state to another to accomplish a task or make a decision within a given environment. A local workflow consists of a series of processes that need to be accomplished to complete a task step-by-step, from initiation to completion. It is also useful for users to understand their particular roles in a specific environment by visualizing the processes involved in a service. This means that a local workflow serves as a user guide, enabling smooth navigation through the processes of each service. Consequently, a well-defined local workflow reduces user effort, enhances user satisfaction, and supports service providers in achieving long-term loyalty.

2) Data Layer: This layer stores and manages all relevant information exchanged between a sandbox server and mobile devices. This includes user profiles, preferences, and any other data necessary for the functioning of the proposed architecture. This data layer ensures that the information is securely stored and can be accessed by the sandbox server when needed for processing user requests.

3) Communication Layer: This layer handles incoming requests from mobile users and manages the transmission of data between a sandbox server and mobile devices, ensuring that data is returned to mobile users in a timely manner. It provides secure and efficient communication via a local Wi-Fi network by implementing protocols that ensure all communications are encrypted, protected from unauthorized access, and properly protected during transmission. These protocols include Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), and WebSocket, which are utilized to send requests from the mobile user to the sandbox server and responses from the sandbox server to the mobile user. The WebSocket protocol especially provides a powerful, persistent connection for establishing bidirectional, real-time, and event-based communication channels between a mobile user, typically a web browser, and a sandbox server using WebSockets over a single Transmission Control Protocol (TCP) connection. It allows for frequent data updates and instant interaction. With this connection, the sandbox server can push data to the mobile user in real time while simultaneously receiving data from the mobile user. It eliminates the need for continuous polling or long-polling techniques commonly used in traditional HTTP-based communication. As a result, it significantly achieves scalability by reducing the overhead of HTTP requests and responses, facilitating low-latency data transfer, and improving the efficiency and speed of data transmission. This capability is particularly beneficial for applications that require real-time updates, such as chat applications, notification platforms, and multiplayer games. Additionally, it facilitates real-time synchronization of data between the mobile device and the sandbox server, ensuring that both systems stay updated with the latest information.

The sandbox server provides access over a local Wi-Fi network, allowing mobile users to interact with the application services from a specific area. This can help reduce the spread of misinformation and disinformation, as mobile users can access credible and relevant information specific to their locations and contexts.

In a typical scenario, when a mobile user comes within range of a sandbox server environment, the mobile user needs to authenticate before allowing them to establish the connection through a local Wi-Fi network. The mobile user initially verifies their identity by checking in to the sandbox server within the targeted environment using Bluetooth proximity to receive a Wi-Fi password directly sent to the registered mobile number via SMS. The mobile user then enters the received password to gain access to the resources and services through the local Wi-Fi network. The mobile user can set up profiles with some basic information that allows the sandbox server to provide them with more personalized recommendations and information. Once the mobile user has completed their profile settings, the sandbox server interacts with the mobile user by offering standard services. These standard services enable the mobile user to obtain useful information and communicate with others who share their interests, questions, and discussions. The mobile user can also access additional services. Then, specific services, personalization recommendations, and information will be integrated based on their profile and current situation in the form of user interfaces and dialogues. The mobile user utilizes a local workflow, which displays the navigational paths the mobile user takes in the environment from their entry point through a set of processes to reach a successful outcome or final interaction. In the situation where a new mobile user lacks historical and personal data, the proposed architecture initially suggests related information and services based on the current community, such as recently viewed items, frequently purchased items, best sellers, trending items, most viewed items, popular items, and featured items. When mobile users exist in the environment, their personal and behavioral data are automatically recorded since they started interacting in the recent environment. These data are saved as a daily schedule in a highly secure database in order to protect user privacy while working on the proposed architecture. It can also help the proposed architecture deeply understand the habits and preferences of mobile users and accurately tailor more personalized information and recommendations to them.

IV. Experiments

In this section, a series of experiments are conducted to validate the robustness and effectiveness of the proposed architecture, ensure that it can support the expected number of concurrent users, and work smoothly in real-world environments. In general, the experiments conducted aim to answer the following Research Questions (RQs):

- **RQ1**: Can the web pages hosted on a sandbox server handle a large number of concurrent users?
- **RQ2**: What maximum user load can web pages handle before the response time exceeds an acceptable threshold?
- **RQ3**: Can the APIs hosted on a sandbox server support a certain number of requests per second?
- **RQ4**: What is the maximum request rate that APIs can handle before the response time exceeds an acceptable threshold?

The detailed experimental procedures are shown in the following subsections:

A. Experimental Strategies

The experimental strategies aim to create a realistic simulation of user behavior under various load conditions, divided into two categories: (1) Web page load testing and (2) API load testing. The following strategies are explained below.

1) Web Page Load Testing: This strategy involves testing the load and performance of individual web pages under various scenarios to ensure smooth flow and an optimal user experience. The experiment simulates multiple users accessing web pages simultaneously and measuring various factors, such as server response time, concurrent user capacity, and page load time. The goal is to ensure that web pages can handle high traffic and load without any failures, because web pages that are slow to load or fail under heavy traffic can lead to a poor user experience and the loss of users or customers.

2) API Load Testing: This strategy involves sending a high volume of requests to the API endpoint concurrently to evaluate the scalability and reliability of APIs used in a web application. In contrast, instead of simulating user interactions with a website, API load testing works by sending requests directly to an API and measuring how it performs. API load testing is important for applications that rely heavily on APIs, such as microservice architectures, mobile applications, and modern web applications. The main objective is to examine its ability to handle concurrent users, analyzing response times and latency under different network conditions that could impact the functionalities and user experience.

Overall, the mentioned strategies are used to assess the performance of the proposed architecture under various conditions, such as different user loads, and network speeds. This can assist in identifying any performance issues for the purpose of overall improvement.

B. Experimental Settings

To show that the proposed architecture is proofed and achieved, all experiments were conducted on a Personal Computer (PC) with an Intel (R) Core (TM) i5-4570 CPU at 3.20 GHz and 8 GB of RAM as a sandbox server placed in an independent environment. The Xiaomi Redmi Note 11 Pro, based on the Android 11 operating system, was used as a mobile device in all of the experimental testing. Apache JMeter version 5.6.1 was used to simulate various realistic scenarios and implement load testing solutions. Nginx version 1.23.3 was set up as a web server to serve dynamic web pages and web applications written in Python and Java. The backend data was stored in MySQL Server version 8.0.31, which was running on an Ubuntu 22.04 LTS Linux server.

The proposed architecture was evaluated based on the two strategies mentioned above. In web page load testing, a test scenario was set up for evaluating the performance of web pages under the peak load of 500 virtual users. It simulated virtual users performing a complete action on two individual web pages that reflect different types of user interactions, including a landing page labeled as *WebPage1* and a contact page labeled as *WebPage2*, within 60 seconds. The ramp-up period was set to 10 seconds, meaning that the test would start with a few virtual users and increase gradually over 10 seconds until it reached 500 virtual users. In API load testing, a test scenario was created to identify the maximum load of APIs hosted on a sandbox server and potential bottleneck issues. It simulated 1,000 virtual users to execute different scenarios of APIs, including a topic recommendation API labeled as *API1* and a registration API labeled as *API2*, within 100 seconds. The ramp-up period was set to 5 seconds. In addition, the expected response time was less than 200 milliseconds for 95% of requests.

V. Experimental Results and Discussion

To test how well the proposed architecture works, the experiments were run on two main strategies. The results show

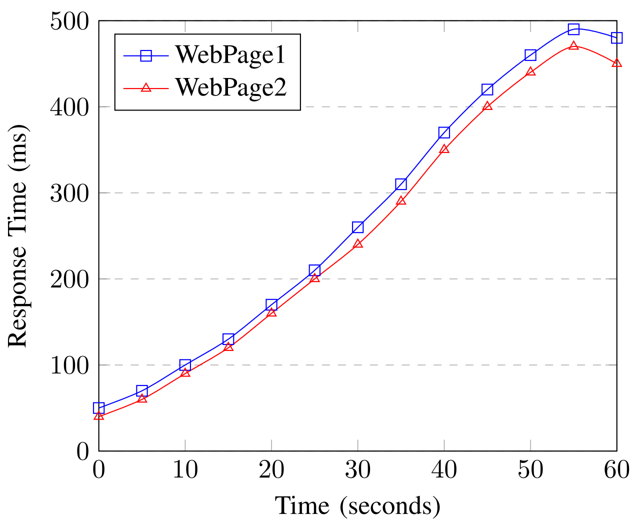

Figure 4. Web page load testing results

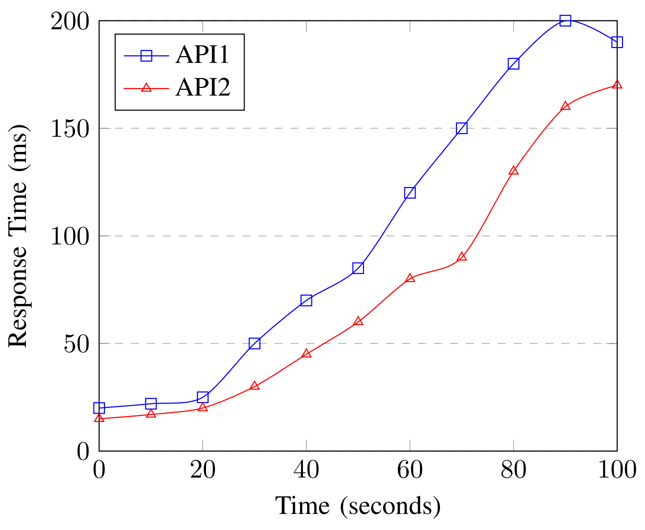

Figure 5. API load testing results

how effectively it works in different situations, indicating that it is good at optimizing performance.

A. *The Results of Web Page Load Testing*

The experimental results of the proposed architecture in terms of web page load testing were reported to address RQ1 and RQ2. The response time measurement was an important metric to assess the performance of both web pages.

Figure 4 demonstrated the graph representing the result of web page load testing over time. The response time values corresponded to different times ranging from 0 to 60 seconds. The blue line represented *WebPage1*, which started with a lower response time of 50 milliseconds at the beginning, but its response time dramatically increased, reaching a peak of around 490 milliseconds at 55 seconds. After this point, the response time slightly decreased to nearly 480 milliseconds at 60 seconds. On the other hand, the red line represented *Web-Page2*, which had a response time of roughly 40 milliseconds at the start of the test. It reached a peak of approximately 470 milliseconds at 55 seconds. The response time then reduced slightly to about 450 milliseconds by the end of the test.

In comparison with the two web pages, *WebPage1* always had a slightly longer response time than *WebPage2* because it is the landing page, has a larger data size, and embeds more complex functionalities, images, videos, and audio content, causing it to respond slower under the same load conditions.

This graph indicated that the response times of both web pages increased as the number of concurrent users grew. Another point was that both web pages would be in trouble after about 30 seconds because their response times exceeded the acceptable threshold.

B. *The Results of API Load Testing*

The experimental results of the proposed architecture in terms of API load testing were reported to address RQ3 and RQ4. The response time measurement was a crucial metric for comparing the performance of the two APIs.

The graph in Figure 5 illustrates the result of API load testing, which was tested over a period of 100 seconds. The response time was in milliseconds, ranging from 0 to 200 milliseconds. Each point on a line represented the response times of APIs at a particular second of the test. The blue line represented the performance of *API1*, whose response time started at 20 milliseconds and gradually increased. There was a significant shift in the response time from 60 seconds to 90 seconds, with a peak at around 200 milliseconds before slightly dropping to 190 milliseconds at the end of the test. The red line represented the performance of *API2*, which started with a low response time of 15 milliseconds. The peak response time was around 170 milliseconds at 100 seconds.

According to these results, *API2* outperformed *API1* under load because *API1* was a recommendation engine, which ran more complicated algorithms and processes than the algorithm of *API2*.

From the graph, it could be observed that the response time gradually increased as the number of concurrent users grew. Moreover, both APIs achieved a good response time because it was under 200 milliseconds.

In summary, the proposed architecture achieved remarkable performance for web pages and APIs. It could be deployed in a production environment because it can satisfactorily maintain several factors at acceptable levels, including reliability, scalability, robustness, and efficiency. Additionally, the response time of both web pages and APIs remained consistently below the desirable threshold of 200 milliseconds, ensuring a positive user experience. This could involve optimizing the backend, such as improving database queries, increasing server resources, using connection pooling to leverage load balancing techniques, or deploying on cloud platforms, and the frontend, such as minimizing JavaScript or optimizing images.

VI. A Use Case

In this section, an example of a local conference scenario located in a specific area is considered to describe the storytelling and planning processes of the proposed architecture. For instance, imagine that hundreds of participants enter a conference room and also open a mobile application connected to a local Wi-Fi network. They will receive necessary services, information, and recommendations in real time, such as a list of participants with profiles, scheduled programs, presentations, documents, videos, and any other relevant material, which will be reflected immediately in the mobile application for participants who are granted access to the network. These services and information generated by a local sandbox server are sent proactively and reactively as instant notifications when participants enter the conference room in proximity. Moreover, the mobile application can be customized to allow participants to collaborate with each other who are in the same geographical location, meet new friends, make comments, and share information while displaying relevant information, providing a personalized experience for each participant.

Finally, when participants enter the conference environment, they can explore all of the services and information they need. In addition, the full integration between a local sandbox server and a mobile application should be designed to create a comprehensive and engaging experience, making it easy for participants to seamlessly navigate through the conference digitally and identify what topics or items interest them the most.

VII. Opportunities and Challenges

The proposed architecture has great potential for use in various smart environments within geographical areas, such as smart homes, innovative universities, smart cities, and smart industries. In the context of smart cities, it utilizes real-time data and location-based intelligence to create digital landscapes of urban infrastructure, in some cases resulting in a digital twin of a particular city. The proposed architecture is designed as a generic concept that could provide opportunities for full integration into smart cities, towns, or villages via a mobile application to enhance the quality of life for citizens and make their lives more efficient and convenient.

VIII. Conclusion

This paper proposes an intelligent, location-aware architecture for mobile computing environments. The main idea is to attempt to interact with individual mobile users by automatically offering suitable local services, information, and recommendations that meet their needs and expectations based on their current location and profile when mobile users appear in the vicinity of a sandbox server and keep track of their position, presence, and movement. It also deals with crucial aspects, including user privacy, data security, and scalability. According to the experimental results, the proposed architecture is smart, scalable, and reliable enough for practical deployment in various real-world environments.

Based on this research, future work will involve simulating the proposed architecture in real-world scenarios to examine its performance. This can help identify issues that might not appear in laboratory testing. Real mobile users will also assess the proposed architecture to gather feedback on usability, responsiveness, and the overall user experience. Future research could also investigate integrating emerging technologies, such as 5G networks and edge computing, to enhance the capabilities of intelligent location-aware platforms.

References

[1] A. Dix, T. Rodden, N. Davies, J. Trevor, A. Friday, and K. Palfreyman, "Exploiting space and location as a design framework for interactive mobile systems," *ACM Trans. Comput.-Hum. Interact.*, vol. 7, no. 3, p. 285–321, sep 2000. [Online]. Available: https://doi.org/10.1145/355324.355325

[2] C. K. Georgiadis, "Mobile commerce application development: Implementing location-aware information services," in *2009 Fifth Advanced International Conference on Telecommunications*, May 2009, pp. 333–338.

[3] G. Yovanof and G. Hazapis, "An architectural framework and enabling wireless technologies for digital cities & intelligent urban environments," *Wireless Personal Communications*, vol. 49, pp. 445–463, 05 2009.

[4] S. H. Jang and C. W. Lee, "The impact of location-based service factors on usage intentions for technology acceptance: The moderating effect of innovativeness," *Sustainability*, vol. 10, no. 6, 2018. [Online]. Available: https://www.mdpi.com/2071-1050/10/6/1876

[5] H. R. Schmidtke, "Location-aware systems or location-based services: a survey with applications to covid-19 contact tracking," p. 191–214, Sep 2020. [Online]. Available: http://dx.doi.org/10.1007/s40860-020-00111-4

[6] J. Shang, S. Yu, F. Gu, Z. Xu, and L. Zhu, "A mobile guide system framework for museums based on local location-aware approach," in *2011 International Conference on Computer Science and Service System (CSSS)*, 2011, pp. 1935–1940.

[7] S. Thaiprayoon and H. Unger, "Towards personalized context-aware recommendation agent in mobile social networks," in *AFIN 2022, The Fourteenth International Conference on Advances in Future Internet*. IARIA, 2022, pp. 1–8. [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=afin_2022_1_10_40004

[8] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, and T. Berners-Lee, "A demonstration of the solid platform for social web applications," in *Proceedings of the 25th International Conference Companion on World Wide Web*, ser. WWW '16 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, p. 223–226. [Online]. Available: https://doi.org/10.1145/2872518.2890529

[9] S. Thaiprayoon and H. Unger, "Towards design and implementation of the breakthrough web," *International Journal on Advances in Networks and Services*, vol. 16, no. 1&2, pp. 14–26, 2023.

[10] Q. Fang, C. Xu, M. S. Hossain, and G. Muhammad, "Stcaplrs: A spatial-temporal context-aware personalized location recommendation system," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 4, pp. 1–30, mar 2016. [Online]. Available: https://doi.org/10.1145/2842631

[11] B. P. Knijnenburg and A. Kobsa, "Making decisions about privacy: Information disclosure in context-aware recommender systems," *ACM Trans. Interact. Intell. Syst.*, vol. 3, no. 3, oct 2013. [Online]. Available: https://doi.org/10.1145/2499670

[12] S. Renjith, A. Sreekumar, and M. Jathavedan, "An extensive study on the evolution of context-aware personalized travel recommender systems," *Information Processing & Management*, vol. 57, no. 1, p. 102078, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457319300111

[13] A. Sunikka and J. Bragge, "Applying text-mining to personalization and customization research literature – who, what and where?" *Expert Systems with Applications*, vol. 39, no. 11, pp. 10 049–10 058, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417412002862