Implementation of markerless augmented reality and cyber-physical-social systems for smart tourism application

Ilham Firman Ashari[1], Fanesa Hadi Pramana[1], Muhammad Zainal Arifin[2], Purwono Prasetyawan[3]
[1]Department of Informatics Engineering, Faculty of Industrial Technology, Institut Teknologi Sumatera, South Lampung, Indonesia
[2]Department of Informatics Engineering, Faculty of Engineering, Universitas Muhammadiyah Malang, Malang, Indonesia
[3]Department of Electrical Engineering, Faculty of Industrial Technology, Institut Teknologi Sumatera, South Lampung, Indonesia

Article Info

Article history:

Received Jul 19, 2025
Revised Nov 19, 2025
Accepted Dec 8, 2025

Keywords:

Application
Cyber-physical-social systems
Hierarchical coordinate transformation architecture
Lampung province
Markerless augmented reality

ABSTRACT

Lampung province holds substantial tourism potential that remains underutilized due to fragmented information and limited promotional strategies. This study introduces a smart tourism application integrating markerless augmented reality (AR) with cyber-physical-social systems (CPSS), representing the first implementation of its kind for location-based tourism in the region. The novelty lies in the hierarchical coordinate transformation architecture (HCTA), a multi-layer computational framework employing the Haversine formula to achieve high-precision mapping of geographic coordinates into AR-optimized perceptual views. The system was evaluated for geolocation accuracy, resource utilization, backend scalability, AR rendering robustness, and user experience. Results show strong performance: geolocation tests across seven destinations yielded a mean error rate of 1.5%; AR operations remained efficient with 8–10% central processing unit (CPU) and 140–160 MB random access memory (RAM) usage; and rendering was stable across 360° device orientation. Backend tests confirmed scalability, sustaining 56 requests per second with zero failures under 100 concurrent users. A user study with 20 participants using the user experience questionnaire-short (UEQ-S) revealed highly positive outcomes, with overall scores 2.275, all within the Excellent benchmark. These findings confirm that the application is not only technically robust and efficient but also engaging and enjoyable, offering a scalable framework for immersive smart tourism ecosystems.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Ilham Firman Ashari
Informatics Engineering, Faculty of Industrial Technology
Institut Teknologi Sumatera
St. Terusan Ryacudu, South Lampung, Indonesia
Email: firman.ashari@if.itera.ac.id

1. INTRODUCTION

In the digital era, technology has become a critical driver of transformation in the global tourism industry. Mobile applications, web-based platforms, and immersive technologies such as augmented reality (AR) and cyber-physical-social systems (CPSS) enable more interactive, personalized, and sustainable travel experiences [1]–[3]. These innovations broaden access to tourism information while enhancing user engagement through dynamic and context-aware services [4], [5]. Within this broader trend, regions such as Lampung in Indonesia face both opportunities and challenges in leveraging digital technologies to strengthen destination competitiveness and visitor satisfaction [6]–[8].

Lampung province in Indonesia is renowned for its natural and cultural richness, offering substantial potential for tourism development [9], [10]. Popular destinations such as Marina Beach, Mutun Beach, Sari Ringgung Beach, Pahawang Island, and Way Kambas National Park attract local and domestic tourists [11], [12]. However, this potential has not been fully realized due to limited promotion, fragmented information, and weak community engagement [13], [14]. The absence of effective marketing strategies has contributed to Lampung's relatively low competitiveness in the national tourism landscape, with the province ranked only 12th in domestic tourist trips in 2024 [15], [16]. Existing tourism platforms remain restricted to static web content with little interactivity or personalization, while earlier AR-based applications are largely marker-dependent, limiting flexibility and, in some cases, disrupting the natural aesthetics of tourist sites. These shortcomings highlight the urgent need for an adaptive, immersive, and user-centered smart tourism solution, particularly through the integration of markerless AR and CPSS technologies.

CPSS integrate digital technologies with human and social interactions to create intelligent and adaptive environments [17]. In tourism, CPSS enable the development of smart, sustainable, and personalized travel experiences that improve visitor satisfaction and contribute to regional competitiveness [18]. As outlined by Ashari [19], CPSS consist of three key dimensions: cyber (connectivity and cloud computing), physical (mobile devices equipped with global positioning system (GPS) and sensors), and social (user-generated content and feedback). When combined with AR, CPSS can further enhance engagement by delivering immersive and context-aware services, positioning AR–CPSS integration as a cornerstone of smart tourism ecosystems [20].

AR has gained increasing attention in tourism for its ability to provide real-time overlays of cultural and natural attractions, thereby enriching the visitor experience [21]. Compared to marker-based systems, markerless AR offers greater flexibility and immersion by eliminating the need for static triggers such as quick response (QR) codes [22]. Studies in Indonesia have confirmed its usability, reporting accuracy rates above 90% and strong user satisfaction [23], although device compatibility remains a limitation [24]. In contrast, recent evaluations of marker-based AR highlight practical challenges, including sensitivity to lighting conditions, viewing distance, and marker durability, with usability scores rated only at a moderate level (system usability scale (SUS) = 71.58) [25]. In Lampung province, earlier AR applications have shown potential in promoting destinations and delivering interactive content to tourists [26]. Building upon these findings, this study proposes the integration of mobile and web-based markerless AR with CPSS to deliver dynamic, adaptive, and user-centered tourism services that overcome the constraints of conventional systems.

Despite these advantages, several limitations remain. Marker-based AR systems are inherently restricted by their reliance on predefined physical markers, which limits flexibility and applicability across diverse environments. Moreover, existing mobile-based implementations often demand relatively high hardware specifications to support real-time rendering, reducing accessibility for users with mid-range or low-end devices. These constraints pose challenges for scalability and widespread adoption of AR-based tourism applications.

Beyond marker dependency and hardware limitations, location-based AR in tourism faces fundamental challenges in coordinate transformation. Existing AR frameworks often fail to accurately render distant geographic points within perceptually constrained environments, causing destinations located hundreds of kilometers away to appear distorted or invisible [27], [28]. Such spatial mapping inadequacies undermine user experience, as virtual objects lose coherence and directional accuracy when projected into real-world settings [22], [29]–[31]. The absence of a systematic, multi-layered coordinate transformation architecture has resulted in inconsistent AR object placement, poor depth perception, and compromised spatial relationships between virtual tourism content and physical geographic references [32]. This highlights the need for a systematic, multi-layered computational framework capable of bridging real-world geodetic coordinates with AR-optimized perceptual positioning. To address this gap, this study introduces the hierarchical coordinate transformation architecture (HCTA), a novel approach designed to improve spatial accuracy and ensure immersive, consistent rendering of virtual tourism content.

Building on these challenges and previous research, there is a clear need for a technology-driven solution that can improve access to tourism information in Lampung province. Mobile and web-based applications are particularly well-suited to this purpose, as they provide on-demand access to services and greater flexibility for tourists regardless of time or location. Markerless AR offers a significant advantage in this context by eliminating the need for physical markers such as QR codes, thereby enabling broader exploration of tourist sites while preserving their natural aesthetics [33]–[35]. This approach enriches the visitor experience through interactive and immersive features. Among the available frameworks, AR.js stands out as a promising tool for developing markerless AR-based applications compatible with both Android and web platforms. When integrated with the CPSS paradigm, AR.js can deliver real-time visualizations of nearby attractions, enriched with ratings and user-generated content. By combining the cyber (digital), physical (environmental), and social (feedback) dimensions, this integration fosters innovative approaches for promoting Lampung's tourism while enhancing user engagement and information dissemination.

The novelty of this study lies in the integration of markerless AR and CPSS within mobile for tourism development in Lampung province. Unlike earlier works that primarily employed marker-based AR for destination promotion, this research adopts a holistic approach that combines digital infrastructure, geospatial accuracy, and community interaction to create an adaptive and user-centered smart tourism experience. This integration not only enhances engagement through immersive and flexible content delivery but also leverages real-time data and feedback to provide personalized services. Consequently, the proposed system strengthens Lampung's competitiveness in the national and international tourism landscape.

In the development of smart tourism applications, it is essential to conduct both functional and non-functional testing to ensure feasibility and optimal performance under real-world conditions [24], [36]. For AR-based systems, this includes evaluating overlay accuracy, geolocation precision, object readability, and application programming interface (API) response times as indicators of technical efficiency. However, beyond technical performance, assessing user experience is equally critical to determine whether the system provides practical value and engagement. To this end, the user experience questionnaire short (UEQ-S) was employed, as it offers a concise yet validated framework that effectively captures both pragmatic aspects (usefulness, efficiency, clarity) and hedonic aspects (stimulation, enjoyment, novelty). Compared to the full UEQ, the short version provides a more efficient and time-saving measurement of user satisfaction while reducing respondent burden [37].

2. METHOD

Figure 1 illustrates the five-phase research workflow. Phase 1 designs the CPSS architecture (cyber, physical, social aspects). Phase 2 outlines business processes, including user interaction, contribution, location search, and AR display. Phase 3 introduces the HCTA for accurate AR rendering through geographic, perceptual, and device coordinate systems. Phase 4 integrates CPSS logic with data flow, coordinate mapping, and real-time rendering. Phase 5 conducts evaluations on location accuracy, resource utilization, AR rendering, API performance, and user experience (UEQ-S).

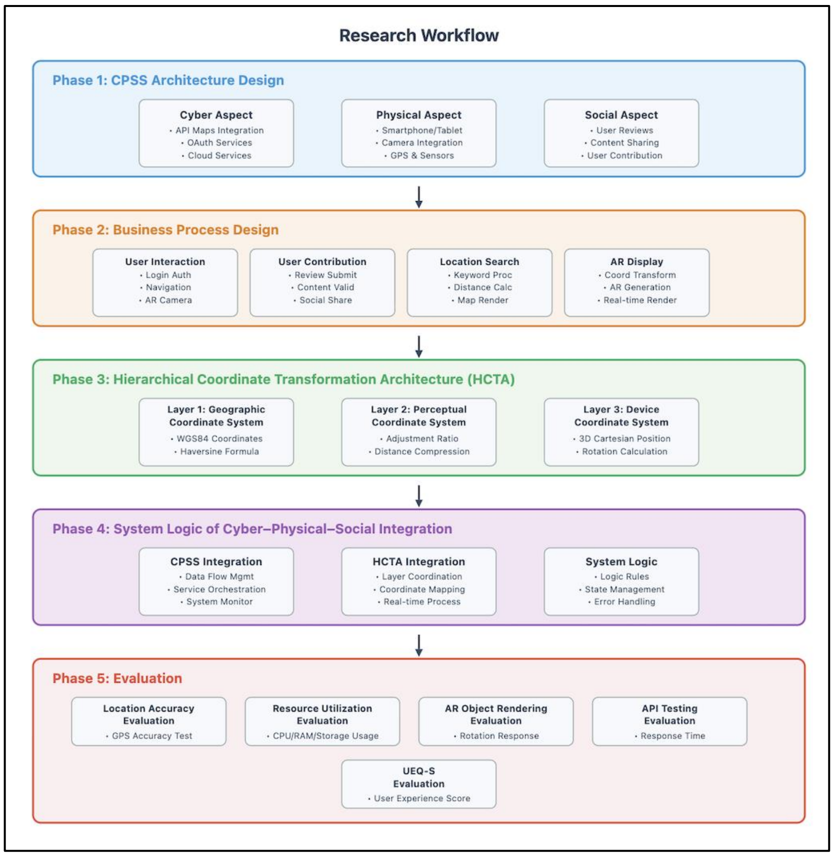


Figure 1. Research workflow diagram

2.1. Phase 1: CPSS architecture design

Figure 2 illustrates the development architecture of a markerless AR tourism application based on the Cyber–Physical–Social System (CPSS) paradigm. The architecture combines physical elements such as mobile devices and sensors with cyber components including servers, cloud services, and APIs to support real-time data processing. Social aspects are integrated through user interaction and content sharing to enhance contextual and interactive tourism experiences.

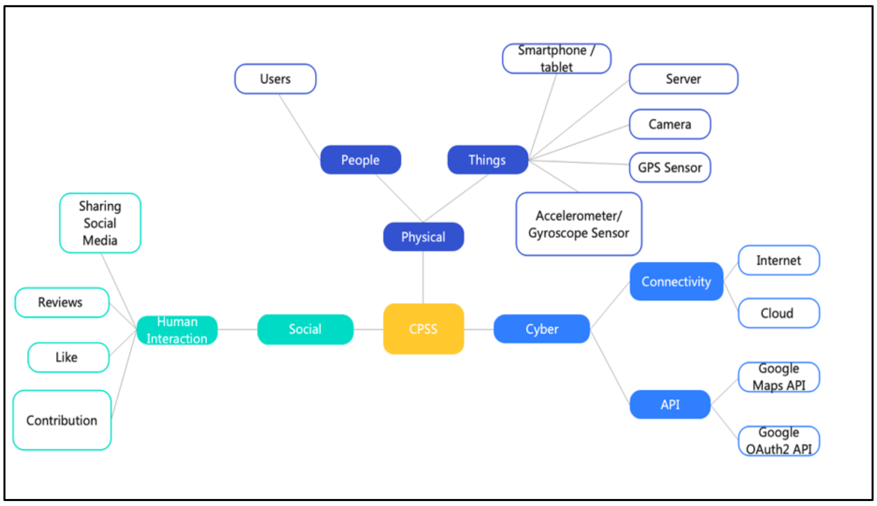


Figure 2. Cyber-physical-social system diagram

Based on Figure 2, the core components of the CPSS framework integrated into the developed system are outlined as follows.

2.1.1. Cyber aspect

The cyber aspect of CPSS refers to how data is collected and processed from users' smartphones via embedded sensors to enhance information delivery to other users [38]. The following outlines the cyber aspect of CPSS, location-based tourism applications, incorporating maps APIs, OAuth for secure access, and internet connectivity.

- Api maps: the application integrates the Google Maps API to determine user location and support navigation features such as optimal routing, traffic information, and estimated travel time [39], [40].
- OAuth services: by implementing OAuth, the application eliminates the need to store passwords or other sensitive credentials, thereby reducing the risk of security breaches [41].
- Internet: an active internet connection is essential for enabling access to cloud services and real-time maps APIs. it also facilitates integration with external information sources, including user reviews, destination ratings, and social media content, allowing users to share travel experiences, photos, and reviews directly through the platform.

2.1.2. Physical aspect

The physical aspect includes all hardware components that interact with the real world [42]. Some physical aspects developed in the system are:

- The user is a human actor who interacts with physical environments via smartphone or tablet devices.
- Smartphones or tablets serve as the primary devices for tourists to access information. They encompass critical hardware components such as screens, cameras, GPS, compasses, and gyroscopes, which collectively enable AR functionalities.
- The camera is used to captures the user's surroundings, which are then processed to overlay AR content aligned with the physical environment.
- GPS sensor provides real-time location data to accurately determine the user's position in the physical world.
- Compass and gyroscope sensors are used to help determine the orientation and direction of the device to display AR objects in the right position.

– The server as a backend component for storing and processing large-scale tourism data, which is transmitted to user devices as needed.

2.1.3. Social aspect

Each individual within the system contributes through various interactions that enhance the richness, relevance, and overall quality of the content presented [38], [43].

– Users can contribute by sharing their experiences through textual reviews and image uploads on various tourist destinations [19]. This feature enables the exchange of impressions, insights, and practical tips to assist other travelers. Shared content can also be distributed via integrated social media platforms such as Facebook, Instagram, X (formerly Twitter), and WhatsApp.
– Users are permitted to contribute new tourist destinations after submitting at least ten reviews on different locations. The "contribute" feature enables active users to provide detailed information about attractions, including location data, available facilities, and other relevant details.
– Service quality assessment is a key feature within the CPSS framework [44]. Users can express feedback through likes, dislikes, and star ratings as a form of evaluation for each tourist attraction. Metrics such as view counts, likes, and overall ratings are publicly displayed.

2.2. Phase 2: business process design

Some of the core business processes and functional activities implemented within the system are as follows.

2.2.1. User interaction with tourist attraction information

Figure 3 illustrates the user interaction process after logging in with a Google account. The dashboard provides multiple access points, including explore, maps, AR camera, and Bookmarks. From these, users can search for attractions, view details, and perform various actions such as saving to bookmarks, submitting reviews, liking/disliking, opening routes with Google Maps, or sharing information via social media platforms.

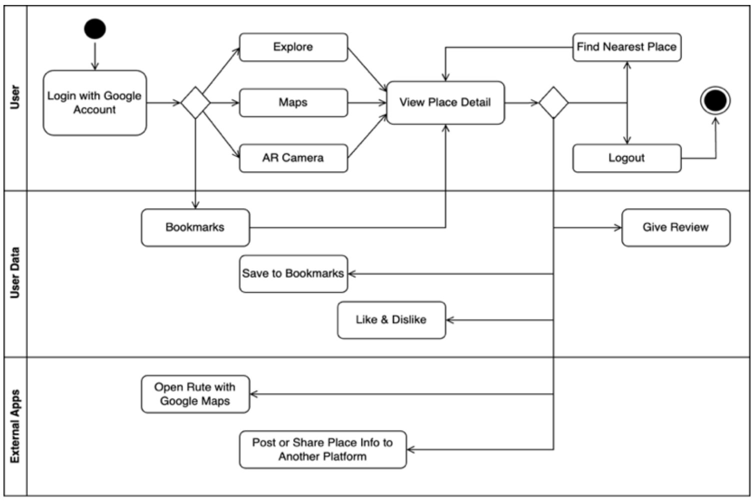


Figure 3. Workflow of user interactions in accessing tourist destination information

2.2.2. User contributor: adding new tourist attractions

As shown in Figure 4, new users cannot immediately add tourist destinations. To unlock this feature, they must first submit at least ten reviews across different locations. Once the requirement is met, the system grants access to the contribution page, allowing users to register and submit detailed information about new destinations.

2.2.3. Tourist location search process using maps

As shown in Figure 5, users enter keywords and their current coordinates into the maps interface, which are sent to the server. The system queries matching destinations, calculates distances, and filters results within a 500 km radius of Lampung province. The filtered locations are then rendered on the Google Maps canvas, with a dynamic zoom adjustment applied to ensure all markers are properly displayed.

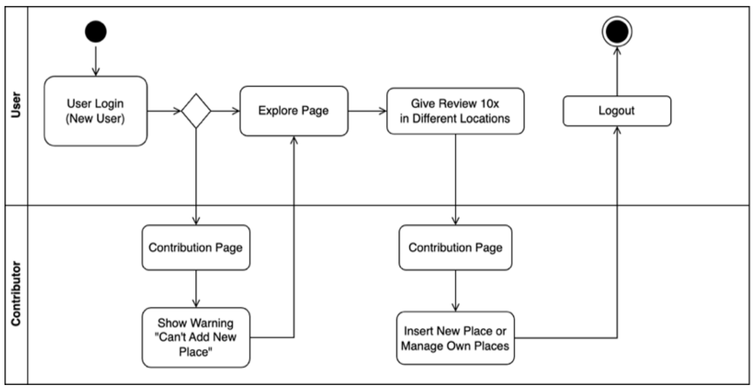

User
Contributor
User Login (New User)
Explore Page
Give Review 10x in Different Locations
Logout
Contribution Page
Show Warning "Can't Add New Place"
Contribution Page
Insert New Place or Manage Own Places

Figure 4. Workflow of user contribution in the tourism information system

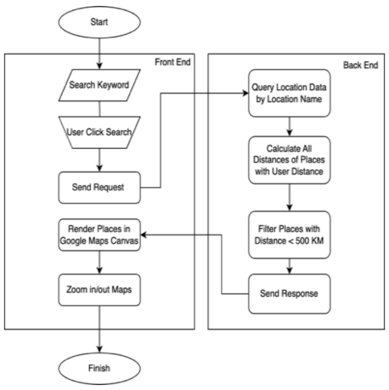

Start
Front End
Back End
Search Keyword
User Click Search
Send Request
Render Places in Google Maps Canvas
Zoom in/out Maps
Query Location Data by Location Name
Calculate All Distances of Places with User Distance
Filter Places with Distance < 500 KM
Send Response
Finish

Figure 5. Process flow for location-based tourist search

2.2.4. AR card display process based on tourist location

Figure 6 shows the AR card display workflow on the AR camera interface. When activated, the system sends the user's location to the server, which retrieves the three nearest tourist attractions. The AR.js module then adjusts their coordinates to 10, 15, and 20 meters to simulate depth and perspective before rendering them in real time as AR cards on the camera canvas.

Figure 7 illustrates the coordinate transformation applied when the AR camera is activated. The system adjusts distant tourist locations (100–250 km) to appear at 10, 15, and 20 meters from the user by recalculating their coordinates. This ensures AR objects remain visible and appropriately scaled, as distant points would otherwise appear too small or be invisible.

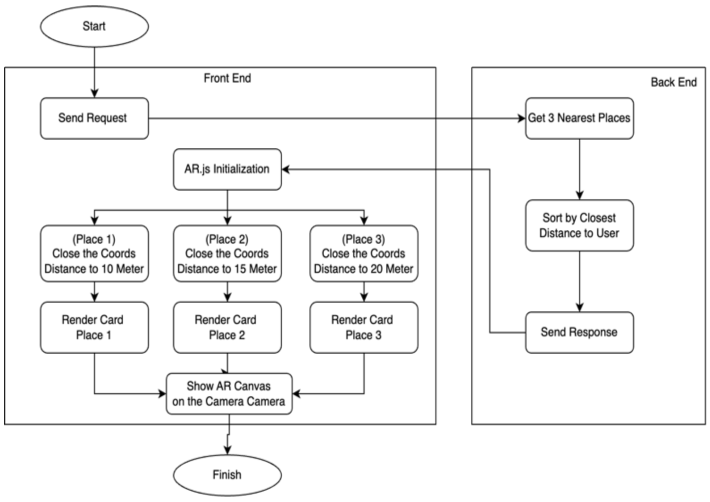

Start
Front End
Back End
Send Request
Get 3 Nearest Places
AR.js Initialization
Sort by Closest Distance to User
(Place 1) Close the Coords Distance to 10 Meter
(Place 2) Close the Coords Distance to 15 Meter
(Place 3) Close the Coords Distance to 20 Meter
Render Card Place 1
Render Card Place 2
Render Card Place 3
Send Response
Show AR Canvas on the Camera Camera
Finish

Figure 6. Process of displaying location on the AR card

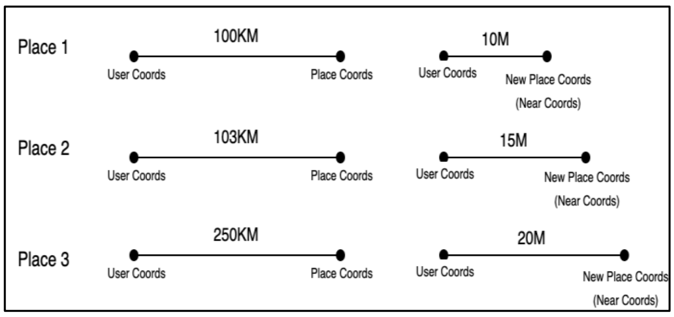
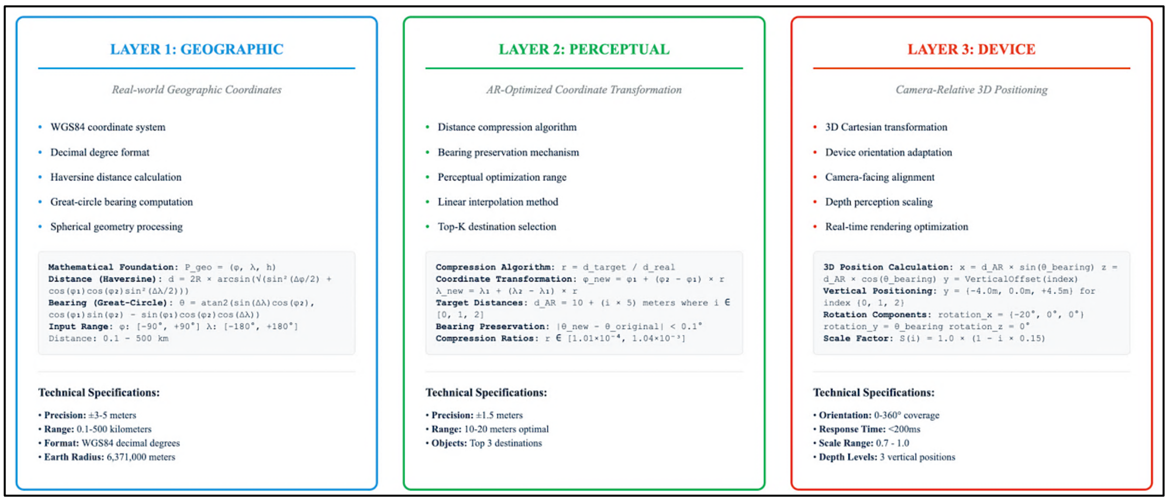


Figure 7. Distance transformation from actual location to AR-rendered position

2.3. Phase 3: HCTA

The widespread adoption of AR applications in tourism has highlighted the critical need for sophisticated coordinate transformation systems. These systems are essential for accurately bridging real-world geographic coordinates with perceptually optimized virtual object placements. This paper proposes (HCTA), illustrated in Figure 8. This novel three-layer computational framework aims to solve the fundamental problem of displaying distant tourist destinations within AR environments while maintaining spatial accuracy and optimizing user experience. The HCTA operates via a systematic transformation pipeline, converting WGS84 geographic coordinates through distance compression algorithms, ultimately yielding camera-relative 3D positioning suitable for real-time AR rendering.



LAYER 1: GEOGRAPHIC

Real-world Geographic Coordinates

- WGS84 coordinate system
- Decimal degree format
- Haversine distance calculation
- Great-circle bearing computation
- Spherical geometry processing

```
Mathematical Foundation: P_geo = (φ, λ, h)
Distance (Haversine): d = 2R × arcsin(√(sin²(Δφ/2) +
cos(φ₁)cos(φ₂)sin²(Δλ/2)))
Bearing (Great-Circle): θ = atan2(sin(Δλ)cos(φ₂),
cos(φ₁)sin(φ₂) - sin(φ₁)cos(φ₂)cos(Δλ))
Input Range: φ: [-90°, +90°] λ: [-180°, +180°]
Distance: 0.1 - 500 km
```

Technical Specifications:

- **Precision:** ±3-5 meters
- **Range:** 0.1-500 kilometers
- **Format:** WGS84 decimal degrees
- **Earth Radius:** 6,371,000 meters

LAYER 2: PERCEPTUAL

AR-Optimized Coordinate Transformation

- Distance compression algorithm
- Bearing preservation mechanism
- Perceptual optimization range
- Linear interpolation method
- Top-K destination selection

```
Compression Algorithm: r = d_target / d_real
Coordinate Transformation: φ_new = φ₁ + (φ₂ - φ₁) × r
λ_new = λ₁ + (λ₂ - λ₁) × r
Target Distances: d_AR = 10 + (i × 5) meters where i ∈
[0, 1, 2]
Bearing Preservation: |θ_new - θ_original| < 0.1°
Compression Ratios: r ∈ [1.01×10⁻⁴, 1.04×10⁻³]
```

Technical Specifications:

- **Precision:** ±1.5 meters
- **Range:** 10-20 meters optimal
- **Objects:** Top 3 destinations

LAYER 3: DEVICE

Camera-Relative 3D Positioning

- 3D Cartesian transformation
- Device orientation adaptation
- Camera-facing alignment
- Depth perception scaling
- Real-time rendering optimization

```
3D Position Calculation: x = d_AR × sin(θ_bearing) z =
d_AR × cos(θ_bearing) y = VerticalOffset(index)
Vertical Positioning: y = {-4.0m, 0.0m, +4.5m} for
index {0, 1, 2}
Rotation Components: rotation_x = {-20°, 0°, 0°}
rotation_y = θ_bearing rotation_z = 0°
Scale Factor: S(i) = 1.0 × (1 - i × 0.15)
```

Technical Specifications:

- **Orientation:** 0-360° coverage
- **Response Time:** <200ms
- **Scale Range:** 0.7 - 1.0
- **Depth Levels:** 3 vertical positions

Figure 8. HCTA for AR tourism applications

2.3.1. Layer 1: geographic coordinate system (GCS)

a. Coordinate representation

In the context of markerless AR tourism applications, precise coordinate specification becomes critical for establishing the spatial relationship between user location and tourist destinations. The world geodetic system 1984 (WGS84) serves as the global reference frame, providing a standardized ellipsoidal model that enables consistent geographic positioning across different platforms and applications, can be seen in (1) [45].

$$P_{geo} = (\varphi, \lambda, \mathrm{h})) \tag{1}$$

where,
- φ = latitude in decimal degrees [-90°, +90°]
- λ = longitude in decimal degrees [-180°, +180°]
- h = height above WGS84 ellipsoid (optional)

b. Degree to radian conversion

The conversion from degrees to radians represents a critical preprocessing step in spherical calculations. This essential conversion is systematically applied to geographic coordinates using the following equations: For the first latitude, lat_1 (in degrees), its radian equivalent $\varphi 1$ is calculated using (2). Similarly, the second latitude, lat_2 (in degrees), is converted to $\varphi 2$ in radians according to (3). Furthermore, the difference in longitudes, $(lon_2 - lon_1)$ (in degrees), is transformed into $\Delta\lambda$ in radians as described by (4). These conversions ensure that all angular inputs are in the appropriate radian format.

$$\varphi_1 = lat_1 \times (\frac{\pi}{180}) \tag{2}$$

$$\varphi_2 = lat_2 \times (\frac{\pi}{180}) \tag{3}$$

$$\Delta\lambda = (lon_2 - lon_1) \times (\frac{\pi}{180}) \tag{4}$$

where,
- φ (phi): latitude in radians
- λ (lambda): longitude in radians
- π: Pi constant (3.14159)

c. Haversine component

The haversine component represents the heart of the distance calculation, combining both latitudinal and longitudinal differences into a single scalar value that captures the three-dimensional relationship between two points on Earth's surface. The value a resulting from (5) will then be used in the subsequent steps of the Haversine formula to compute the final distance. This is the fundamental part that "weights" the contributions of both latitude and longitude changes simultaneously in the Earth's three-dimensional space [46].

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \times \cos(\varphi_2) \times \sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{5}$$

where,
- $\Delta\varphi$: $\varphi_2 - \varphi_1$ (latitude difference)
- $\Delta\lambda$: $\lambda_2 - \lambda_1$ (longitude difference)

The haversine formulation is specifically designed to avoid computational problems that arise in alternative approaches. The law of cosines formula does indeed suffer from round-off errors for small distances due to the computation of $cos(c) = 1 - \frac{c^2}{2}$ for small c, leading to loss of precision when subtracting nearly equal numbers.

d. Great-circle distance

The transformation from the haversine component to the actual angular distance employs the inverse haversine function, implemented through the two-parameter arctangent function [47]. This angular distance, denoted as 'c', is precisely calculated using (6) and distance in meter calculated using (7).

$$c = 2 \times atan2 \times (\sqrt{a}, \sqrt{1-a}) \tag{6}$$

$$d = R \times c \tag{7}$$

where,
- d: distance in meters
- R: earth's radius (6,371,000 m)
- $atan2$: two-parameter arctangent function

Subsequently, a circle bearing calculation is applied to determine the angular direction between the user and the destination [48]. This enables the system to rotate the AR object along the Y-axis, ensuring that it consistently faces the camera within a 3D environment. The Figure 9 illustrates this transformation and orientation process. By calculating the object's bearing relative to the user's position, the front of the AR object can be consistently oriented toward the camera's center point. As a result, the object's front face remains visible regardless of camera rotation.

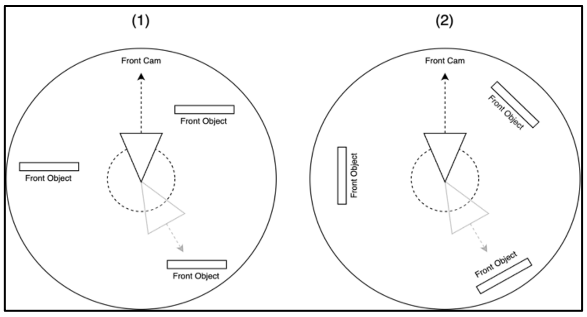


Figure 9. Circle bearing between user position and tourist attraction

e. Vector components

The great-circle bearing calculation requires computing two vector components that represent the direction from the starting point to the destination on a spherical Earth model. Component X, which captures the eastward-westward directional influence, is calculated using (8). Simultaneously, Component Y, representing the northward-southward directional influence, is derived using (9).

$$X = \sin(\Delta\lambda) \times \cos(\varphi_2) \tag{8}$$

$$Y = \cos(\varphi_1) \times \sin(\varphi_2) - \mathit{sin}(\varphi_1) \times \cos(\varphi_2)\cos(\Delta\lambda) \tag{9}$$

where,
- φ_1: latitude of starting point (in radians)
- φ_2: latitude of destination point (in radians)
- $\Delta\lambda$: difference in longitude ($\lambda_2 - \lambda_1$) in radians

f. Bearing angle

The final step in determining the great-circle bearing is to calculate the bearing angle itself, which utilizes these vector components. The initial bearing angle, θ, in radians, is computed directly from Components X and Y using the two-argument arctangent function, as defined by (10).

$$\theta = atan2(X, Y) \tag{10}$$

To make this bearing intuitive for navigation, it is then converted from radians into standard compass degrees. This conversion involves scaling by $\frac{\pi}{180}$ and ensuring the result is within the 0° to 360° range using the modulo 360 operation, as shown in (11)

$$bearing = \left(\theta \times \frac{180}{\pi} + 360\right) mod\ 360 \tag{11}$$

where,
- θ: bearing angle in radians
- $atan2(X, Y)$: two-argument arctangent function that preserves quadrant information

The result gives 0° = North, 90° = East, 180° = South, and 270° = West.

2.3.2. Layer 2: perceptual coordinate system (PCS)

a. Adjustment ratio

Since AR objects at actual geographic distances would often be too far to see clearly (potentially kilometres away), we need to bring them closer to an optimal viewing distance while maintaining their directional accuracy. This adjustment ratio, denoted by r, is calculated by dividing the optimal AR viewing distance d_{AR} by the true geographic distance d_{actual} obtained from the Haversine formula, as shown in (12).

$$ratio\ (r) = \frac{d_{AR}}{d_{actual}} \tag{12}$$

where,
- r: adjustment ratio (0 < r < 1)
- d_{AR}: optimal AR viewing distance (typically 10-20 meters)
- d_{actual}: true geographic distance calculated using Haversine formula

This ratio represents how much we need to “shrink” the actual distance to achieve the desired AR viewing distance.

b. Adjusted coordinates

The adjusted coordinates are calculated by linear interpolation along the great-circle path. Specifically, the adjusted latitude, φnew, is determined by adding the product of the latitude difference $(\varphi_{dest} - \varphi_{user})$ and the adjustment ratio (r) to the starting latitude (φ_1), as depicted in (13).

$$\varphi_{new} = \varphi_{user} + (\varphi_{dest} - \varphi_{user}) \times r \tag{13}$$

Similarly, the adjusted longitude, λ_{new}, is calculated by adding the product of the longitude difference $(\lambda_{dest} - \lambda_{user})$ and the adjustment ratio (r) to the starting longitude (λ_1), as shown in (14).

$$\lambda_{new} = \lambda_{user} + (\lambda_{dest} - \lambda_{user}) \times r \tag{14}$$

where,

- (φ_1, λ_1): user's current position coordinates
- (φ_2, λ_2): actual point of interest (POI) location coordinates
- $(\varphi_{new}, \lambda_{new})$: adjusted coordinates for AR display
- r: adjustment ratio from (6)

2.3.3. Layer 3: device coordinate system (DCS)

a. AR distance calculation

The (15) calculates how far away an AR object will appear from the user. The distance is calculated based on an index. The base distance is 10 meters. For each increment of the index i, the distance increases by 5 meters.

$$d_{ar} = d_{base} + (i \times d_{increment}) \tag{15}$$

where,

- d_{ar}: AR distance (in meters)
- d_{base}: base distance (constant: 10 meters)
- i: object index (integer: 0, 1, 2, ...)
- $d_{increment}$: distance increment per index (constant: 5 meters)

b. 3D cartesian position calculation

The 3D position coordinates are calculated using polar-to-Cartesian transformation as illustrated in (16) and (17), incorporating the bearing angle and optimized AR distance.

$$x = d_{ar} \times \sin(\theta\ bearing) \tag{16}$$

$$z = d_{ar} \times \cos(\theta\ bearing) \tag{17}$$

where:

- x, z: horizontal plane coordinates (meters)
- $\theta_{bearing}$: bearing angle in radians
- d_{AR}: AR distance from (13)

c. Vertical position formula

Vertical position formula as seen in (18) determines the object's height (yposition) relative to the user's eye level.

$$\begin{aligned}&y_{position} = \{\\ &-4.0\ if\ i = 0\ (below\ eye\ level)\\ &0.0\ if\ i = 1\ (at\ eye\ level)\\ &4.5\ if\ i \geq 2\ (above\ eye\ level)\\ &\}\end{aligned} \tag{18}$$

d. Rotation formula

Rotation formula defines sets the orientation or direction the object is facing in 3D.

1) X-rotation ($rotation_x$), this is the object's upward, or downward tilt as seen in (19).

$$\begin{aligned}&rotation_x = \{\\ &-20°\ if\ i = 0\ (tilted\ down)\\ &0°\ if\ i\ \geq 1\ (horizontal)\\ &\}\end{aligned} \tag{19}$$

$if\ i = 0$, the object is tilted down by 20 degrees.
$if\ i \geq 1$, the object is horizontal (no tilt).

2) Y-rotation ($rotation_y$), this is the object's left or right direction (bearing), which is determined by the value of the variable θ bearing as seen in (20).

$$rotation_y = \theta_{\text{bearing}}(for\ all\ objects) \tag{20}$$

where,
θ_{bearing} : Bearing angle (in degrees, typically 0-360°)
3) Z-Rotation ($rotation_z$), the value is always 0 degrees, meaning the object is not rolled sideways as seen in (21).

$$rotation_z = 0° \ (for\ all\ objects) \tag{21}$$

e. Scale factor calculation
The scale factor formula as illustrated in (22) provides depth perception by gradually reducing object size with increasing distance priority.

$$S(i) = S_{base} \times \ (1 - i \ \times \ decay_{factor}) \tag{22}$$

where,
- $S(i)$: scale factor for object index i
- S_{base}: base scale factor (1.0)
- $decay_{factor}$: scale reduction per index (0.15)
- i: object index (0, 1, 2, ...)

f. Complete 3D transformation vector
This is the final output that combines all previous calculations into a format the AR system can use.
4) Position Vector (P_{ar}), defines the object's location (coordinates), as illustrated in (23).

$$P_{ar} = (x, y, z) = (0, y_{position}, 0) \tag{23}$$

where,
- x: $d_{ar} \times sin(\theta\ bearing)$
- *y: (from (9))*
- z: $d_{ar} \times cos(\theta\ bearing)$

x, z is Calculated from distance and bearing for a real-world AR system, though simplified to 0 in the provided visualization's vector representation for conceptual clarity.
5) Rotation vector (R_{ar}), defines the object's orientation, as illustrated in (24).

$$R_{ar} = (\alpha, \beta, \gamma) = (rotation_x, \theta_{\text{bearing}}, 0) \tag{24}$$

where,
- $\alpha: rotation_x$ (19)
- $\beta: rotation_y$ (20)
- $\gamma: rotation_z$ (21)

2.3.4. Vertical positioning framework and spatial distribution analysis

Figure 10 illustrates the vertical positioning of AR objects within a user-centric coordinate system, where the user/camera serves as the reference origin at eye level ($y = 0$). Three objects are placed at different distances: $i = 0$ at 10 m with a vertical offset of –4.0 m and a –20° tilt, $i = 1$ at 15 m aligned with eye level ($y = 0$), and $i = 2$ at 20 m elevated by +4.5 m. The diagram highlights the spatial relationships through trajectory lines, vertical offsets, and rotation indicators. The downward tilt applied to $i = 0$ ensures perspective-correct rendering for ground-level content, while $i = 1$ and $i = 2$ remain horizontally oriented. This framework demonstrates how distance ($d = 10 + 5i$) and discrete vertical offsets establish a consistent, spatially coherent AR visualization.

2.3.5. Polar coordinate distribution and angular positioning framework

Figure 11 presents the top-view distribution of AR objects using a polar coordinate system, with the user position at the center and concentric circles denoting distance intervals. Objects are placed according to a linear distance formula $d = 10 + (i \times 5)$ meters and a bearing angle θ (0°–360°). Specifically, $i = 0$ is located 10 m away at a 45° bearing, $i = 1$ at 15 m with a 90° bearing, and $i = 2$ at 20 m with a 315° bearing. The angular indicators and radial lines illustrate how distance and bearing jointly define spatial positioning, enabling a full 360° distribution of AR objects. This framework establishes a flexible and mathematically consistent method for organizing AR content in user-centric environments.

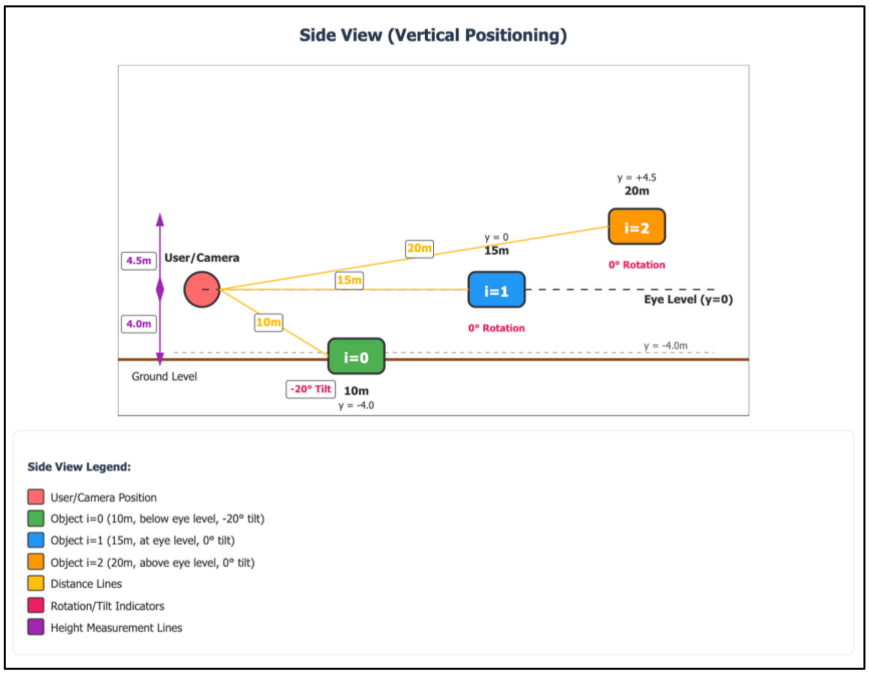


Figure 10. The illustration of side view (vertical positioning)

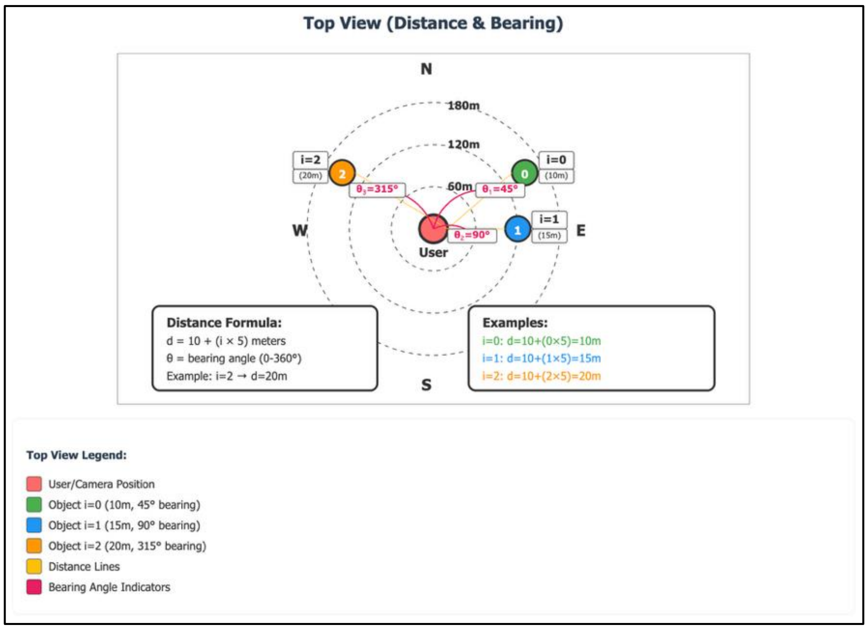


Figure 11. The illustration of top view (distance and bearing)

2.4. Phase 4: system logic of cyber–physical–social system integration

Figure 12 illustrates the integration of CPSS in user location processing. At the physical layer, mobile devices collect geospatial data via GPS, compass, gyroscope, and camera sensors, which are locally preprocessed and secured with authentication tokens. The cyber layer processes this data using Google Maps APIs, the Haversine formula, and the HCTA, while managing POIs, media, and reviews in a structured database. The social layer enables users to contribute reviews, photos, and new destinations, which are moderated and reintegrated into the system. This cyclical flow ensures continuous refinement of AR content and reflects the core principles of CPSS in smart tourism.

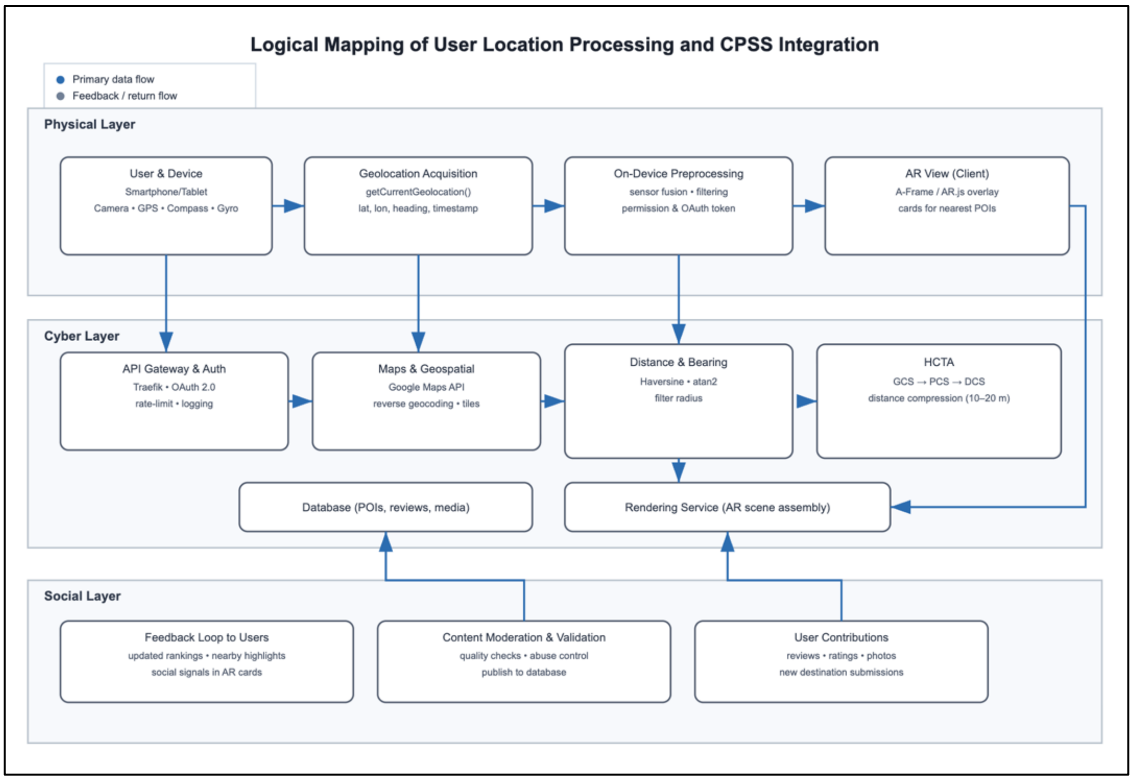


Figure 12. Integration and logical mapping CPSS and user location processing

2.5. Phase 5: evaluation

The evaluation phase was designed to verify the system's feasibility across different categories of Android devices. All experiments were conducted using a Wi-Fi network plan of 75 Mbps, ensuring a stable environment for real-time data communication and AR rendering. The effective throughput during testing was consistently within the range of 68–75 Mbps for download bandwidth and 20–30 Mbps for upload bandwidth, which provided sufficient capacity to handle API requests, geolocation services, and AR visualization. To ensure representative testing, multiple devices with varying hardware configurations and Android operating system versions were included. These devices ranged from entry-level to flagship smartphones, thereby covering diverse user scenarios in practical deployments. The detailed specifications of the devices used in the evaluation are presented in Table 1.

Table 1. Specifications of Android devices used in evaluation

Device brand and model	Android OS version	CPU / RAM
Realme C2	Android 9	2C / 2GB
Vivo 1902	Android 10	4C / 3GB
Oppo A53	Android 11	8C / 4GB
Huawei P30	Android 11	8C / 6GB
Vivo V21	Android 12	8C / 8GB
Samsung A51	Android 12	8C / 6GB
Xiaomi Redmi K40	Android 13	8C / 8GB
Google Pixel 6	Android 14	8C / 8GB
Vivo Y27s	Android 14	8C / 8GB

3. RESULTS AND DISCUSSION

3.1. Backend API architecture

Figure 13 shows the backend architecture deployed on a virtual private server (VPS) using Docker. The system employs NestJS (representational state transfer (REST) API) running on Node.js, connected to a MySQL database managed through Prisma object–relational mapping (ORM) and phpMyAdmin. A Traefik API gateway handles request routing and domain mapping, while Docker ensures deployment consistency and portability. This architecture provides scalable, structured, and reliable API services for mobile applications.

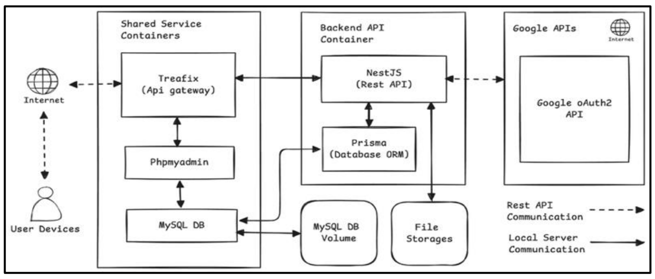


Figure 13. Backend API architecture

3.2. Mobile android architecture

Figure 14 shows the client-side architecture packaged as an Android app rendered via WebView. Developed mainly with VueJS and supporting libraries, the compiled web content runs through an internal HTTP server engine. The stack includes Node.js for development, Pinia for state management, and Axios for API communication. This integration ensures efficient data handling, modularity, and cross-platform compatibility in a hybrid Android environment.

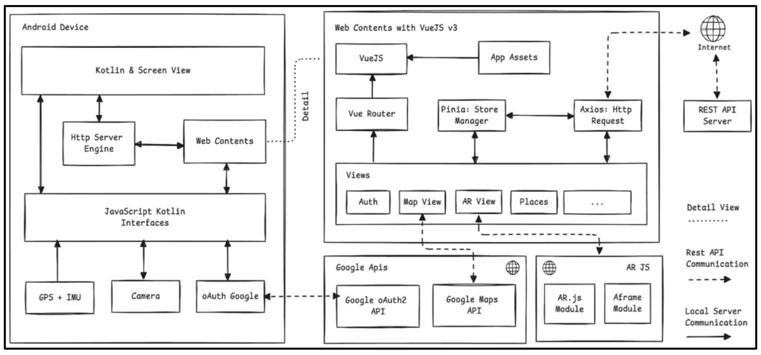


Figure 14. Mobile android architecture

3.3. Application interfaces and function

3.3.1. Home page

The homepage of the application as illustrated in Figure 15 displays a highlight area alongside a categorized and complete list of tourist attractions. In the highlight area, tourist destinations are shown based on their popularity, which is determined by the number of likes, reviews, and views. Algorithm 1 presents the pseudocode for retrieving and displaying highlighted tourist attraction data.

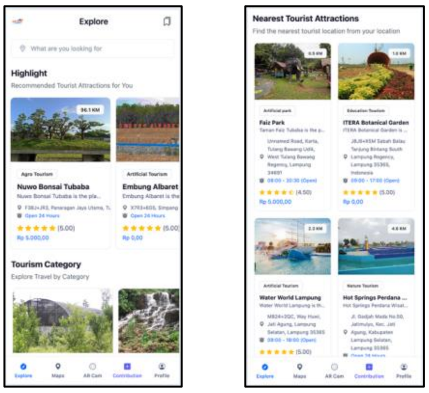


Figure 15. Highlight section on the home interface

Algorithm 1. GetHighlightPlaces (function to retrieve highlighted tourism places)

```
Function getHighlightPlaces() -> List of PlaceEntity
  places = Fetch places from database
    - Filter:
      - status = 'PUBLISHED'
      - dislike count < 5
    - Sort by:
      - rating (descending)
      - like count (descending)
      - saved count (descending)
    - Include related data:
      - category
      - map image cover
    - Limit result to 25 places
  For each place in places:
    If place has a category:
      Set map marker URL using category's map marker
    If place has a map image cover:
      Set map image cover URL using base URL and image filename
  Return places
```

The system begins by querying the places table to obtain tourist attraction data, filtering out entries with more than five dislikes and joining category and file tables to generate complete metadata. Using user coordinates as input, an initial query retrieves identifiers (IDs) and locations of all destinations. Distances are then calculated, sorted in ascending order, and the 50 nearest entries selected. A second query fetches detailed information for these places, followed by a final sorting step. The overall procedure for identifying nearby attractions is outlined in Algorithm 2.

Algorithm 2. GetNearestPlaces (function to get the nearest places based on user coordinates)

```
Function getNearestPlaces(dto: NearestSearchPlaceDto) -> List of PlaceEntity
  places = Fetch all places from the database
    - Filter:
      - status = 'PUBLISHED'
      - dislike count < 5
    - Select:
      - id, latitude, longitude
  myCoord = { latitude: dto.latitude, longitude: dto.longitude }
  placeDistances = Empty list to store place id and distance
  For each place in places:
    placeCoord = { latitude: place.latitude, longitude: place.longitude }
    distance = Calculate distance between myCoord and placeCoord
    Add { placeId, distance } to placeDistances
  Sort placeDistances by distance (ascending)
  nearestPlaceIds = Get top 50 placeIds from sorted placeDistances
  nearestPlaces = Fetch places from the database
    - Filter:
      - id in nearestPlaceIds
    - Include related data:
      - category
      - map image cover
  placeObject = Empty dictionary to store places by id
  For each place in nearestPlaces:
    If place has a map image cover:
      Set map image cover URL using base URL and image filename
    Store place in placeObject using place.id as key
  nearestPlacesSorted = Empty list
  For each placeId in nearestPlaceIds:
    Add place from placeObject to nearestPlacesSorted
  Return nearestPlacesSorted
```

3.3.2. Maps page

The server processes user-submitted keywords to query categories or tourist attractions, returning up to 100 entries prioritized by ratings and likes. Results are filtered to retain only locations within 100 km of the user, then transmitted to the client. The client renders these locations on the Google Maps canvas by clearing old markers, adding new ones, and adjusting the zoom level to ensure all markers are visible. The complete procedure is outlined in Algorithm 3, with the interface shown in Figure 16.

Algorithm 3. RenderPlaceSearchList (function to render place search list on the map)

```
Function renderPlaceSearchList()
  If googleMap is null, return
  For each marker in placeMarkers:
    Remove marker from map
  Clear placeMarkers array
  Import Marker from Google Maps API
  For each place in placeSearchList:
    Create marker using place's latitude, longitude, and category map marker
    Add marker to the map and to placeMarkers array
    Shorten place description if longer than 70 characters
    Get HTML content for marker popup and replace placeholders with place data
    Create info window with the generated content
    If showInfoWindow is true, open info window
    On marker click:
      Set marker animation to bounce
      Open info window
      Stop animation after 1.5 seconds
  Collect all coordinates (user and places)
  Calculate the center coordinates
  If centerCoord is false, return
  Pan the map to the center coordinates
  Calculate the distance of each place from the center
  Set zoom level according to the maximum distance:
    - Zoom level 12 for distances < 10 km
    - Zoom level 11 for 10-20 km
    - Zoom level 8 for 20-150 km
    - Zoom level 7 for 150-300 km
    - Zoom level 5 for 300-1000 km
    - Zoom level 4 for distances > 1000 km
  Apply the zoom level to the map
```

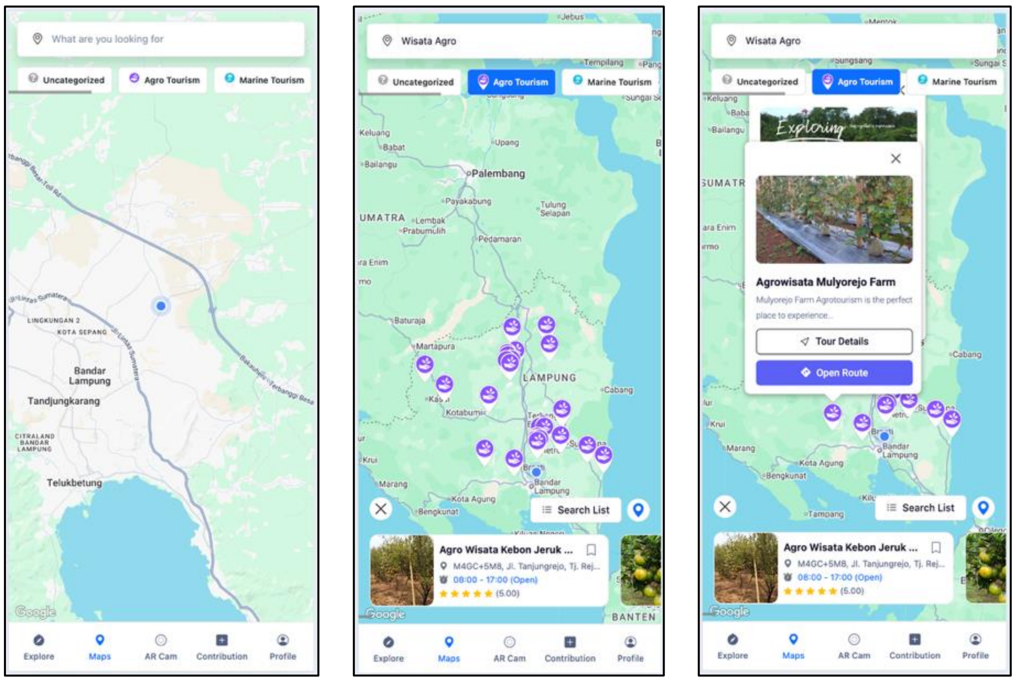


Figure 16. Maps page

The system retrieves categories matching the keyword and queries related tourist attractions. Distances are calculated, and locations beyond 300 km are filtered out. For valid entries, image URLs are generated and sent to the client, which renders them on the Google Maps canvas. Markers are placed based on coordinates, the center point is calculated, and the zoom level is dynamically adjusted according to the maximum distance between locations (e.g., zoom 12 for <10 km, zoom 11 for 10–20 km), ensuring all markers remain visible.

3.3.3. AR camera page

The “renderArCameraPlaces” function orchestrates the AR tourism workflow, from detecting the user’s GPS location to rendering objects in AR. After obtaining coordinates, it retrieves the three nearest attractions, calculates distances, and applies coordinate transformations to reposition distant locations into a visible range of 10–20 meters. Orientation is determined using circle bearing, while depth perception is simulated by placing objects at different vertical levels (below, at, and above eye level) with corresponding rotations. Text truncation ensures readability before data is passed to the rendering pipeline. The full procedure is outlined in Algorithm 4.

Algorithm 4. RenderArPlaces (function to get current location and render AR places)

```
Function renderArCameraPlaces()
  // Phase 1: Geolocation Setup
  Call getCurrentGeolocation()
  Set myCoordinates ← current user location {latitude, longitude}
  Set coordLoaded ← true
  // Phase 2: Fetch Nearby Places
  Call getPlaceArMapSearch(myCoordinates)
  nearbyPlaces ← placeArSearchList (top 3 nearest places)
  // Phase 3: AR Coordinate Transformation
  arPlaces ← Empty array
  For each place in nearbyPlaces with index i do:
  // Calculate actual distance
  actualDistance ← getDistance(myCoordinates, place.coordinates) // in meters
  // Transform coordinates for AR visibility
  targetDistance ← 10 + (i × 5) // 10m, 15m, 20m
  adjustedCoordinates ← moveCloser(myCoordinates, place.coordinates, targetDistance)
  // Calculate circle bearing for proper orientation
  circleBearing ← 360 - getGreatCircleBearing(myCoordinates, adjustedCoordinates)
  // Set initial position and rotation
  position ← "0 -4 0" // below eye level
  rotation ← "-20 ${circleBearing} 0" // tilted down
  // Adjust position based on index for depth perception
  If i = 1 then:
  position ← "0 0 0" // at eye level
  rotation ← "0 ${circleBearing} 0" // horizontal
  If i ≥ 2 then:
  position ← "0 4.5 0" // above eye level
  rotation ← "0 ${circleBearing} 0" // horizontal
  // Create VR Place object
  vrPlace ← {
    ...place,
    latitude: adjustedCoordinates.latitude,
    longitude: adjustedCoordinates.longitude,
    distance: actualDistance / 1000, // convert to KM
    circleBearing: circleBearing,
    position: position,
    rotation: rotation,
    name: displayName,
    description: displayDescription
  }
  Add vrPlace to arPlaces array
  // Phase 4: AR Scene Rendering
  Call renderARScene(arPlaces)
End Function
```

The “moveCloser” function addresses the challenge of rendering distant tourist attractions in AR, where objects beyond 20–30 m become imperceptible. It employs the Haversine formula to calculate the great-circle distance, then applies linear interpolation to reposition objects within 10–20 m while preserving directional bearing. As outlined in Algorithm 5, the function converts coordinates, computes an adjustment ratio, and returns transformed latitude and longitude values that maintain spatial accuracy and relative positioning in the AR view.

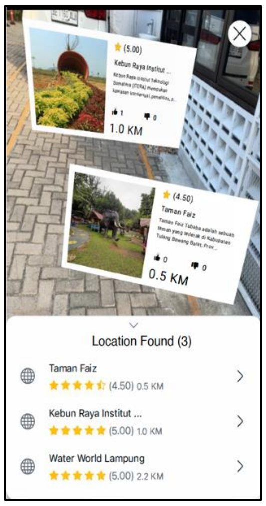
The renderARScene function (Algorithm 6) manages AR visualization by initializing the A-frame environment with GPS tracking and camera integration. It generates interactive AR entities for each transformed location, positioned using adjusted coordinates and rotation values. Each entity is displayed as an information card containing images, ratings, text, interaction icons, and distance data, optimized for mobile viewing. The function also supports touch-based navigation and dynamically adjusts resolution and scene parameters to ensure smooth performance across devices. The resulting AR interface is shown in Figure 17.

Algorithm 5. MoveCloser (function to get current location and render AR places)

```
1:   Function moveCloser(start, target, newDistance)
2:     // Convert degrees to radians
3:     startLat ← toRadians(start.latitude)
4:     startLon ← toRadians(start.longitude)
5:     targetLat ← toRadians(target.latitude)
6:     targetLon ← toRadians(target.longitude)
8:     // Calculate coordinate differences
9:     dLat ← targetLat - startLat
10:    dLon ← targetLon - startLon
11:    // Haversine formula for distance calculation
12:    a ← $\sin^2(dLat/2) + \cos(startLat) \times \cos(targetLat) \times \sin^2(dLon/2)$
13:    c ← $2 \times \mathrm{atan2}(\sqrt{a}, \sqrt{(1-a)})$
14:    actualDistance ← earthRadius × c
15:    // Calculate adjustment ratio
16:    ratio ← newDistance / actualDistance
17:    // Apply linear interpolation
18:    newLat ← startLat + (dLat × ratio)
19:    newLon ← startLon + (dLon × ratio)
20:    Return {
21:    latitude: toDegrees(newLat),
22:    longitude: toDegrees(newLon)
23:  End Function
```

Algorithm 6. RenderARScene (function to Get current location and render AR places)

```
1:   Initialize render context and setup viewport dimensions
2:   Load scene graph and camera parameters
3:   Set up lighting environment (ambient, directional, point lights)
4:   Configure material properties and shaders
5:   For each object in scene:
6:   Apply model transformations (position, rotation, scale)
7:   Calculate model-view-projection matrix
8:   Check if object is in view frustum
9:   If visible, submit to render queue
10:  Sort render queue by material/shader for performance
11:  For each render pass:
12:  Bind appropriate framebuffer (shadow maps, G-buffer, etc.)
13:  Clear buffers (color, depth, stencil)
14:  Set render state (depth test, blending, etc.)
15:  For each object in current pass:
16:  Bind shader program
17:  Set uniform values (matrices, textures, etc.)
18:  Bind vertex/index buffers
19:  Issue draw call
20:  Apply post-processing effects (bloom, DOF, tone mapping)
21:  Render UI and overlay elements
22:  Present final frame to display
23:  Update performance metrics and debug information
```


(5.00)
Kebun Raya Institut ...
1.0 KM
(4.50)
Taman Faiz
0.5 KM

Location Found (3)

Taman Faiz
(4.50) 0.5 KM

Kebun Raya Institut ...
(5.00) 1.0 KM

Water World Lampung
(5.00) 2.2 KM

Figure 17. Markerless AR interface using camera

3.4. Testing and evaluation

3.4.1. Evaluation of location accuracy in map-based services

The accuracy test is conducted to evaluate the precision of distance calculations between the user's location and various destination points. In this scenario, the user is positioned at Sumatera Institute of Technology, with coordinates (-5.358804, 105.314882). The measured distances in the application are compared with reference values from Google Maps. The accuracy test results are presented in Table 2.

Table 2. Comparison of location accuracy test results between google maps and the developed application

Location name	Coordinate (lat, long)	Distance in app (Km)	Distance in maps (Km)	Error rate (%) = | (distance in app - distance in maps) / distance in app | × 100
Taman Nasional Way Kambas	(-4.7138, 105.9331)	98.93	97.2	1.75
Pantai Mahitam Lampung	(-5.5822, 105.3615)	25.24	24.99	0.99
Pantai Mutun Lampung	(-5.526, 105.2785)	18.92	18.38	2.9
Pantai Arang Lampung	(-5.776, 105.4126)	47.39	45.9	3.14
Jumbo Seafood Lampung	(-5.4398, 105.2644)	10.56	10.53	0.3
Pantai Embe Lampung	(-5.6676, 105.1822)	37.18	35.73	3.91
Pantai Sari Ringgung	(-5.5233, 105.2599)	19.18	19.18	0.02

3.4.2. Evaluation of resource utilization on the AR camera interface

Figure 18 summarizes CPU and RAM usage during AR Camera operation across different Android versions. Newer devices (Android 13–14) showed lower resource consumption (CPU 8–10%, RAM 140–160 MB), delivering smooth and responsive performance (e.g., Google Pixel 6, Vivo Y27s). In contrast, older devices (Android 9–10) required significantly more resources, with the Realme C2 recording the highest usage (CPU 21%, RAM 280 MB), resulting in choppy performance. These results confirm an inverse correlation between resource utilization and AR performance, emphasizing the role of optimized hardware and modern OS in ensuring efficient and seamless AR experiences.

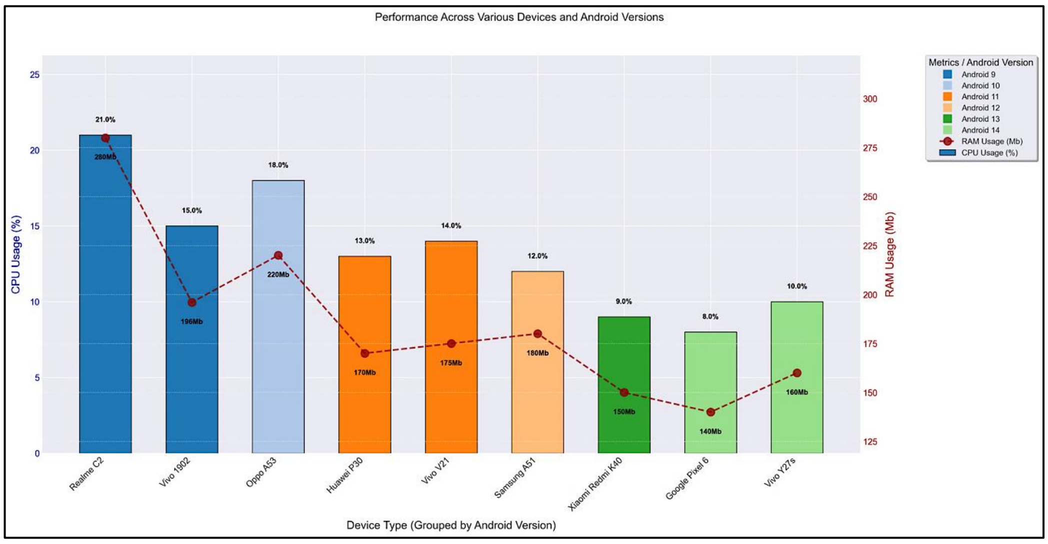
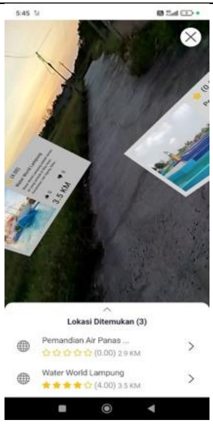
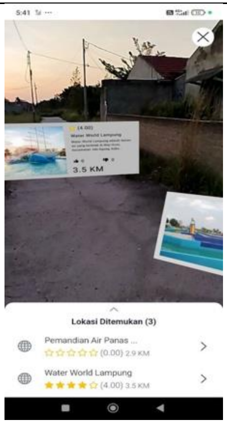
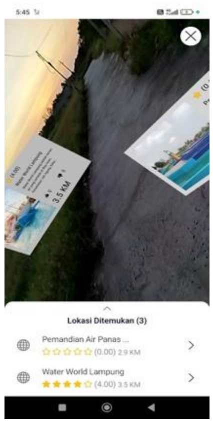
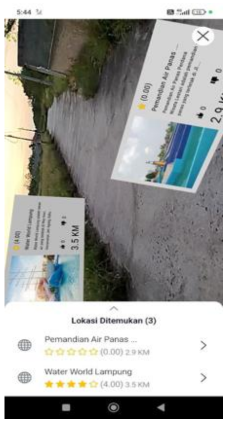
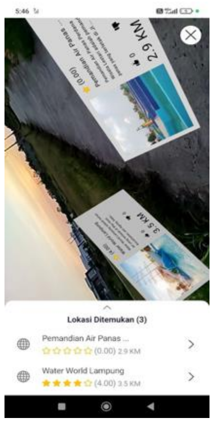
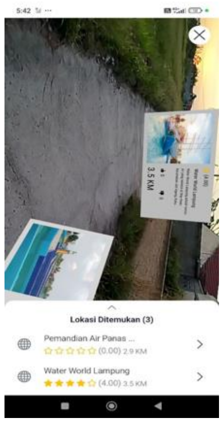


Figure 18. Evaluation of RAM and CPU usage across multiple devices

3.4.3. Evaluation of AR object rendering at different device orientations

This test evaluates the visibility and orientation of AR objects relative to the user's position when viewed from different angles. The user is positioned at Institut Teknologi Sumatera (ITERA) with coordinates (-5.358804, 105.314882), and the displayed objects represent three tourist locations: Lembah Hijau, Langka Langit 2, and Pantai Mahitam. The objective is to verify whether the AR objects align accurately with their real-world directions. The reference angle starts at 0 degrees facing north and rotates clockwise. The results of the AR object orientation test based on camera angles are presented in Table 3.

Table 3. Object recognition accuracy at device rotation angles: 45°, 90°, 135°, 180°, 225°, and 270°

Angle (degree)	Real object	Readable object	Visualization	Angle (degree)	Real object	Readable object	Visualization
45	2 objects	2 objects		90	2 object	2 object	
The test at a 45-degree rotation angle showed that two real objects were successfully detected and correctly rendered within the AR interface. This indicates that a minor rotation from the initial north-facing orientation does not affect the system's object recognition or spatial accuracy.				At a 90-degree rotation, the AR system consistently identified and displayed two real objects, both of which were accurately recognized and readable. The results suggest stable horizontal alignment of AR elements when the device is rotated perpendicularly to the initial view.			
135	2 objects	2 objects		180	2 object	2 object	
The test conducted at 135 degrees confirmed that the system maintained accurate detection of two real objects. Both were clearly rendered and readable, reinforcing that the AR rendering pipeline remains stable even at diagonal viewing angles.				When the device was rotated to 180 degrees (facing south), the AR interface continued to recognize and display the same two objects correctly. This finding indicates that the system maintains orientation consistency and does not misinterpret reversed compass directions.			
225	2 objects	2 objects		270	2 object	2 object	
At 225 degrees, the system detected three real objects in the camera frame, but only two were correctly recognized and readable. This slight deviation is likely caused by overlapping object markers or partial occlusion, suggesting minor visual interference under certain rotation conditions.				The evaluation at 270 degrees demonstrated that two real objects were consistently detected and accurately recognized. The results confirm that even at near-complete lateral rotation, AR object visibility and alignment remain reliable.			

3.4.4. API computation time testing

Figure 19 illustrates the system's API performance measured in requests per second (RPS) under different levels of concurrent user activity. The horizontal axis represents the number of concurrent users (10, 25, 50, and 100), while the vertical axis indicates the throughput, measured as the number of successful API requests processed per second. As shown, the "Search by Keyword" API achieved the highest throughput (56 RPS at 25 concurrent users), followed by the "Recommended Tourist Spots" API, which maintained stable performance around 54–55 RPS across all loads. Meanwhile, proximity-based APIs ("Nearest Spots" and "3 Nearest Locations in AR Camera") produced slightly lower RPS values (46–49) due to additional spatial computation overhead. Despite these variations, no request failures were recorded, confirming system stability and resilience under simultaneous user access. The results demonstrate that the VPS configuration (2 CPU cores, 2 GB RAM, and 20 GB storage) remains adequate for moderate concurrent traffic, though future scaling or caching strategies may further enhance performance.

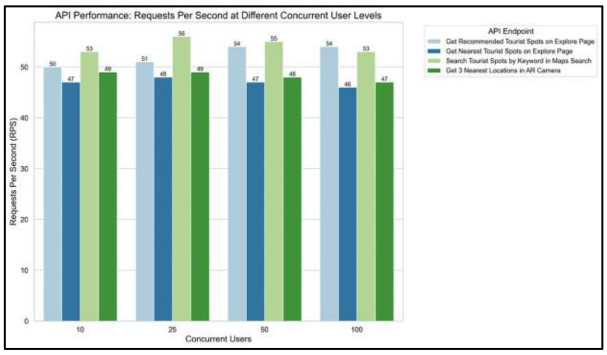


Figure 19. Evaluation of API response time under varying request rates and concurrent loads

3.4.5. User evaluation using user experience questionnaire short

In addition to the technical performance assessment, a user study was carried out with 20 participants to evaluate the experiential aspects of the proposed system. The UEQ-S was adopted, focusing on three dimensions: pragmatic quality, hedonic quality, and overall experience. As shown in Figure 20, the results were highly positive, with pragmatic quality scoring 2.31, hedonic quality scoring 2.24, and an overall score of 2.275, all of which fall within the "Excellent" benchmark category. These findings confirm that the application is not only functionally efficient and reliable, but also stimulating and enjoyable for end users.

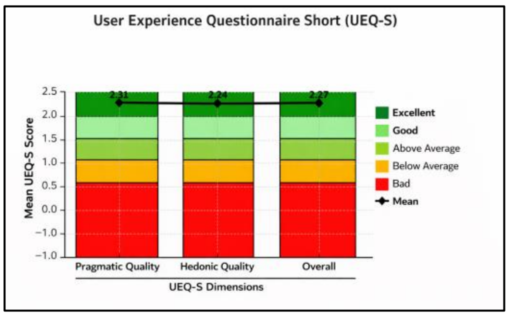


Figure 20. UEQ-S result

4. CONCLUSION

This study demonstrates the potential of integrating markerless AR with CPSS to enhance tourism promotion in Lampung province. The system achieved high spatial accuracy, with an average location error of only 1.5% across seven destinations, stable resource utilization on modern Android devices (CPU 8–10% and RAM 140–160 MB), and robust scalability under load (zero request failures at 100 concurrent users). These results validate the technical feasibility of the proposed solution. Beyond technical performance, the application offers practical benefits by enabling tourists to enjoy immersive and interactive experiences, while providing tourism boards, local governments, and guides with a more effective digital tool for destination promotion and competitiveness.

Future research should prioritize extending accessibility for mid-range devices, improving personalization through AI, and integrating IoT-based data such as weather or visitor density. The addition of offline capabilities will further enhance usability in areas with limited connectivity. Collaboration among technology developers, policymakers, and tourism stakeholders will be essential to ensure sustainable adoption and the broader impact of smart tourism applications in Lampung and beyond.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support provided by the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia under the Fundamental Research scheme (Grant No. 1570aj/IT9.2.1/PT.01.03/2024).

FUNDING INFORMATION

This research was funded by the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia under the Fundamental Research scheme, Grant No. 1570aj/IT9.2.1/PT.01.03/2024.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ilham Firman Ashari	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	
Fanesa Hadi Pramana		✓	✓		✓			✓		✓				
Muhammad Zainal Arifin		✓				✓				✓		✓		
Purwono Prasetyawan		✓				✓		✓		✓				

C	:	**C**onceptualization	I	:	**I**nvestigation	Vi	:	**Vi**sualization
M	:	**M**ethodology	R	:	**R**esources	Su	:	**Su**pervision
So	:	**So**ftware	D	:	**D**ata Curation	P	:	**P**roject administration
Va	:	**Va**lidation	O	:	Writing - **O**riginal Draft	Fu	:	**Fu**nding acquisition
Fo	:	**Fo**rmal analysis	E	:	Writing - Review & **E**diting			

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available on request from the corresponding author, [Ilham, IFA].

REFERENCES

[1] D. Fauzi and P. A. Sumirat, “Evaluation of Indonesian tourism marketing towards quality tourism,” *International Journal of Magistravitae Management*, vol. 1, no. 2, pp. 117–129, Dec. 2023, doi: 10.33019/ijomm.v1i2.22.

[2] D. Nusraningrum, “The sustainability of competitive strategy in the tourism services industry,” *European Journal of Business and Management Research*, vol. 7, no. 4, pp. 60–65, Jul. 2022, doi: 10.24018/ejbmr.2022.7.4.1475.

[3] T. Suryanto, M. Hayati, and Y. Susanti, “The potential of halal tourism system on growth for the province Lampung’s tourism industry,” *Journal of Environmental Management and Tourism*, vol. 13, no. 6, pp. 1616–1628, 2022.

[4] S. Dias and V. A. Afonso, “Impact of mobile applications in changing the tourist experience,” *European Journal of Tourism, Hospitality and Recreation*, vol. 11, no. 1, pp. 113–120, Dec. 2021, doi: 10.2478/ejthr-2021-0011.

[5] Wasino, D. E. Herwindiati, and H. Maupa, “The effects of tourism Web development on prospective travelers by considering persuasive and liking principles,” *International Journal of Social Science and Business*, vol. 6, no. 4, pp. 574–584, Nov. 2022, doi: 10.23887/ijssb.v6i4.49498.

[6] M. H. Ronaghi and M. Ronaghi, “A contextualized study of the usage of the augmented reality technology in the tourism industry,” *Decision Analytics Journal*, vol. 5, Dec. 2022, doi: 10.1016/j.dajour.2022.100136.

[7] S. M. C. Loureiro, J. Guerreiro, and F. Ali, “20 years of research on virtual reality and augmented reality in tourism context: A text-mining approach,” *Tourism Management*, vol. 77, Apr. 2020, doi: 10.1016/j.tourman.2019.104028.

[8] A. Bec, B. Moyle, V. Schaffer, and K. Timms, “Virtual reality and mixed reality for second chance tourism,” *Tourism Management*, vol. 83, Apr. 2021, doi: 10.1016/j.tourman.2020.104256.

[9] D. Kagungan, Y. Neta, H. Yanfika, Rudy, R. Perdana, and A. G. Zainal, “Tourism development policy strategy in Lampung province based on community capacity,” in *Proceedings of the 3rd Universitas Lampung International Conference on Social Sciences (ULICoSS 2022)*, 2023, pp. 1129–1134. doi: 10.2991/978-2-38476-046-6_108.

[10] R. Lestari, T. Rachmawati, F. A. Kamandanu, and D. Syahrobi, “Lampung tourism supply during the pandemic and post Covid-19 pandemic,” in *Proceedings of the 2nd International Indonesia Conference on Interdisciplinary Studies (IICIS 2021)*, 2021. doi: 10.2991/assehr.k.211206.012.

[11] F. X. Sumarja, B. Sujadmiko, T. D. Nguyen, and D. E. Rusmawati, “Transfer of land ownership and marginalization as impact of tourism industry,” *Hasanuddin Law Review*, vol. 9, no. 2, pp. 197–210, Sep. 2023, doi: 10.20956/halrev.v9i2.4593.

[12] Ni Made Ayu Windu Kartika and I Gede Mudana, “Implementation of sustainable tourism at Ketapang Beach Lampung,” *International Journal of Travel, Hospitality and Events*, vol. 2, no. 3, pp. 201–210, Oct. 2023, doi: 10.56743/ijothe.v2i3.298.

[13] S. P. Harianto, M. K. Tsani, R. Arioen, T. P. Zuhelmi, and Surnayanti, “Perceptions of marine tourism in Lampung Bay’s Small Islands: A comparative study,” *International Journal of Design & Nature and Ecodynamics*, vol. 18, no. 5, pp. 1261–1271, Oct. 2023, doi: 10.18280/ijdne.180529.

[14] S. P. Harianto, N. Walid masruri, G. D. Winarno, M. K. Tsani, and T. Santoso, “Development strategy for ecotourism management based on feasibility analysis of tourist attraction objects and perception of visitors and local communities,” *Biodiversitas Journal of Biological Diversity*, vol. 21, no. 2, Jan. 2020, doi: 10.13057/biodiv/d210235.

[15] A. S. Chandra and M. E. Rachman, “The analysis of tourism industry strategies on regional original revenue of Lampung province,” *International Journal of Economics, Business and Innovation Research*, vol. 3, no. 5, pp. 632–644, 2024.

[16] R. Sulistiowati, Y. Yulianto, S. Bakri, M. Mukhlis, and D. A. Saputra, “Analysis of factors influencing re-visit intentions and recommending post-pandemic marine tourism destinations in Lampung Province,” *Journal of Environmental Management and Tourism*, vol. 14, no. 6, pp. 2799–2814, Sep. 2023, doi: 10.14505/jemt.v14.6(70).26.

[17] D. Yin, X. Ming, and X. Zhang, “Understanding data-driven cyber-physical-social system (D-CPSS) using a 7C Framework in social manufacturing context,” *Sensors*, vol. 20, no. 18, Sep. 2020, doi: 10.3390/s20185319.

[18] T. Sobb, B. Turnbull, and N. Moustafa, “A holistic review of cyber–physical–social systems: New directions and opportunities,” *Sensors*, vol. 23, no. 17, Aug. 2023, doi: 10.3390/s23177391.

[19] I. F. Ashari, “Implementation of cyber-physical-social system based on service oriented architecture in smart tourism,” *Journal of Applied Informatics and Computing*, vol. 4, no. 1, pp. 66–73, Jun. 2020, doi: 10.30871/jaic.v4i1.2077.

[20] S. Pasandideh, P. Pereira, and L. Gomes, “Cyber-physical-social systems: Taxonomy, challenges, and opportunities,” *IEEE Access*, vol. 10, pp. 42404–42419, 2022, doi: 10.1109/ACCESS.2022.3167441.

[21] M. Zaifri, H. Khalloufi, F. Z. Kaghat, A. Azough, and K. A. Zidani, “From earlier exploration to advanced applications: Bibliometric and systematic review of augmented reality in the tourism industry (2002–2022),” *Multimodal Technologies and Interaction*, vol. 7, no. 7, Jun. 2023, doi: 10.3390/mti7070064.

[22] P. Q. Brito and J. Stoyanova, “Marker versus markerless augmented reality. Which has more impact on users?,” *International Journal of Human–Computer Interaction*, vol. 34, no. 9, pp. 819–833, Sep. 2018, doi: 10.1080/10447318.2017.1393974.

[23] I. F. Ashari, “Analysis and implementation of augmented reality using Markerless and a-star algorithm (Case study: Gedung Kuliah Umum ITERA),” *Computer Engineering and Applications Journal*, vol. 11, no. 3, pp. 177–190, Oct. 2022, doi: 10.18495/comengapp.v11i3.414.

[24] J. A. S. Boediono, M. R. Aulia, and F. I. Maulana, “Markerless augmented reality application for Indonesian traditional house education,” *Procedia Computer Science*, vol. 227, pp. 718–725, 2023, doi: 10.1016/j.procs.2023.10.576.

[25] G. M. Adrian Putradinata, I. F. Ashari, and E. D. Nugroho, “Evaluation of the implementation of augmented reality on vegetable objects using marker-based tracking and SUS,” *InComTech : Jurnal Telekomunikasi dan Komputer*, vol. 15, no. 2, pp. 110–124, Sep. 2025, doi: 10.22441/incomtech.v15i2.24770.

[26] S. Ferbangkara *et al.*, “Usability of Lampung heritage virtual reality tour,” *Journal of Engineering and Scientific Research*, vol. 4, no. 2, Jan. 2023, doi: 10.23960/jesr.v4i2.107.

[27] O. Bimber and R. Raskar, *Spatial Augmented Reality: Merging Real and Virtual Worlds*. Boca Raton, FL, USA: CRC Press, 2005.

[28] G. Zhang *et al.*, “Exploration of visual variable guidance in outdoor augmented reality geovisualization,” *International Journal of Digital Earth*, vol. 16, no. 2, pp. 4095–4112, Dec. 2023, doi: 10.1080/17538947.2023.2259874.

[29] M. A. Putra, M. Madlazim, and E. Hariyono, “Exploring augmented reality-based learning media implementation in solar system materials,” *IJORER : International Journal of Recent Educational Research*, vol. 5, no. 1, pp. 29–41, Jan. 2024, doi: 10.46245/ijorer.v5i1.440.

[30] J. Safari Bazargani, M. Zafari, A. Sadeghi-Niaraki, and S.-M. Choi, “A survey of GIS and AR integration: Applications,” *Sustainability*, vol. 14, no. 16, Aug. 2022, doi: 10.3390/su141610134.

[31] H.-Y. Chang *et al.*, “Ten years of augmented reality in education: A meta-analysis of (quasi-) experimental studies to investigate the impact,” *Computers & Education*, vol. 191, Dec. 2022, doi: 10.1016/j.compedu.2022.104641.

[32] F. Munoz-Montoya, M.-C. Juan, M. Mendez-Lopez, R. Molla, F. Abad, and C. Fidalgo, “SLAM-based augmented reality for the assessment of short-term spatial memory. A comparative study of visual versus tactile stimuli,” *PLOS ONE*, vol. 16, no. 2, Feb. 2021, doi: 10.1371/journal.pone.0245976.

[33] M. A. Nasirudin, M. F. Md Fudzee, N. Senan, C. S. Che Dalim, D. Witarsyah, and A. Erianda, “Systematic literature review on augmented reality with persuasive system design: Application and design in education and learning,” *JOIV : International Journal on Informatics Visualization*, vol. 8, no. 2, pp. 862–873, May 2024, doi: 10.62527/joiv.8.2.2702.

[34] E. E. Cranmer, M. C. tom Dieck, and P. Fountoulaki, “Exploring the value of augmented reality for tourism,” *Tourism Management Perspectives*, vol. 35, Jul. 2020, doi: 10.1016/j.tmp.2020.100672.

[35] A. W. Liang, N. Wahid, and T. Gusman, “Virtual campus tour application through markerless augmented reality approach,” *JOIV : International Journal on Informatics Visualization*, vol. 5, no. 4, pp. 354–359, Dec. 2021, doi: 10.30630/joiv.5.4.743.

[36] Y. Zhauniarovich, A. Philippov, O. Gadyatskaya, B. Crispo, and F. Massacci, “Towards black box testing of Android apps,” in *2015 10th International Conference on Availability, Reliability and Security*, Aug. 2015, pp. 501–510. doi: 10.1109/ARES.2015.70.

[37] M. Schrepp, A. Hinderks, and J. Thomaschewski, “Design and evaluation of a short version of the user experience questionnaire (UEQ-S),” *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 6, 2017, doi: 10.9781/ijimai.2017.09.001.

[38] B. A. Yilma, H. Panetto, and Y. Naudet, “Systemic formalisation of cyber-physical-social system (CPSS): A systematic literature review,” *Computers in Industry*, vol. 129, Aug. 2021, doi: 10.1016/j.compind.2021.103458.

[39] P. A. Werner, “Review of implementation of augmented reality into the georeferenced analogue and digital maps and images,”

Information, vol. 10, no. 1, Dec. 2018, doi: 10.3390/info10010012.
[40] S. Singh, J. Singh, B. Shah, S. S. Sehra, and F. Ali, "Augmented reality and GPS-based resource efficient navigation system for outdoor environments: Integrating device camera, sensors, and storage," *Sustainability*, vol. 14, no. 19, Oct. 2022, doi: 10.3390/su141912720.
[41] Ö. Şeker, G. Dalkılıç, and U. C. Çabuk, "MARAS: Mutual authentication and role-based authorization scheme for lightweight internet of things applications," *Sensors*, vol. 23, no. 12, Jun. 2023, doi: 10.3390/s23125674.
[42] M. Hamzah *et al.*, "Distributed control of cyber physical system on various domains: A critical review," *Systems*, vol. 11, no. 4, Apr. 2023, doi: 10.3390/systems11040208.
[43] H. Yu, H. Qi, and K. Li, "CPSS: A study of cyber physical system as a software-defined service," *Procedia Computer Science*, vol. 147, pp. 528–532, 2019, doi: 10.1016/j.procs.2019.01.233.
[44] Q. Li, Z. Fang, and M. Qu, "Cyber-physical-social system (CPSS) architecture framework and methodology," in *Proceedings of the 3rd International Conference on Innovative Intelligent Industrial Production and Logistics*, 2022, pp. 206–217. doi: 10.5220/0011559600003329.
[45] A. Hussaini, K. Tang Kang Wee, A. Izzati Abdul Hamid, A. G. Abubakar, and C. W. Tan, "Comparative analysis of GDM2000, WGS84, and MRT68 coordinate reference systems based on different converter modules," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15494–15498, Aug. 2024, doi: 10.48084/etasr.7124.
[46] E. Maria, E. Budiman, Haviluddin, and M. Taruk, "Measure distance locating nearest public facilities using haversine and euclidean methods," *Journal of Physics: Conference Series*, vol. 1450, no. 1, Feb. 2020, doi: 10.1088/1742-6596/1450/1/012080.
[47] R. Bansal, "Deriving and testing the great circle theory," *International Journal of Statistics and Applied Mathematics*, vol. 6, no. 5, pp. 16–24, Sep. 2021, doi: 10.22271/maths.2021.v6.i5a.722.
[48] M. Valizadeh, B. Ranjgar, A. Niccolai, H. Hosseini, S. Rezaee, and F. Hakimpour, "Indoor augmented reality (AR) pedestrian navigation for emergency evacuation based on BIM and GIS," *Heliyon*, vol. 10, no. 12, Jun. 2024, doi: 10.1016/j.heliyon.2024.e32852.

BIOGRAPHIES OF AUTHORS

Ilham Firman Ashari received the Bachelor's degree in Informatics from Diponegoro University, in 2015, the Master's degree from Bandung Institute of Technology, in 2018. He is currently an Assistant Professor at Institut Teknologi Sumatera, Indonesia. His current research interests include cybersecurity, artificial intelligence, and software development. He can be contacted at email: firman.ashari@if.itera.ac.id

Fanesa Hadi Pramana is currently pursuing Bachelor's degree in Informatics Engineering at Institut Teknologi Sumatera, Indonesia. His current research interest include software development and cyber security. He can be contacted at email: fanesa.120140189@student.itera.ac.id.

Muhammad Zainal Arifin received the Bachelor's degree in Informatics from Brawijaya University, in 2003, the Master's degree from Institut Teknologi Sepuluh November, in 2008 and Ph.D. degree from Universiti Teknikal Malaysia Melaka, in 2023. He is currently an Assistant Professor at Universitas Negeri Malang, Indonesia. His current research interests include artificial intelligence, data mining, and mobile applications. He can be contacted at email: arifinzainal@um.ac.id.

Purwono Prasetyawan received the Bachelor's degree in Electrical Engineering from STMIK Indonesia Mandiri, in 2010, the Master's degree from Bandung Institute of Technology, in 2013. He is currently an Assistant Professor and a Lecturer at Institut Teknologi Sumatera, Indonesia. His current research interests include artificial intelligence and cyber-physical system. He can be contacted at email: purwono.prasetyawan@el.itera.ac.id.