Intelligent Transportation and Smart Cities
V. Mezhuyev et al. (Eds.)

doi:10.3233/ATDE250257

Design and Development of a Social Running Platform with Personalized Recommendations and Interactive Mapping

Ling WANG[1]
School of Computer Science, University of St Andrews, St Andrews, UK

Abstract. This study presents the development and evaluation of an advanced social running platform designed to enhance user engagement through personalized activity recommendations and interactive mapping features. Unlike existing platforms that primarily offer basic tracking functions, our solution integrates external map APIs, sophisticated hybrid recommendation algorithms, and robust social interaction capabilities within a scalable Model-View-Controller (MVC) architecture. Utilizing MongoDB for efficient data management, the platform consolidates key entities such as users, activities, and comments, ensuring data consistency and flexibility. Comprehensive testing was conducted to assess search accuracy, response time, cross-platform compatibility, and security, demonstrating superior performance compared to existing competitors. The recommendation system achieved a precision of 92% and a response time of 0.85 seconds, while the platform efficiently handled over 100 concurrent users and ensured strong data protection measures. These findings underscore the platform's potential to significantly improve user experience and community engagement in the digital fitness landscape.

Keywords. Social running platform, personalized recommendations, interactive mapping, MVC architecture, mongoDB

1. Introduction

The proliferation of digital technologies has significantly transformed the landscape of fitness and recreational activities, with social running platforms emerging as pivotal tools for enhancing user engagement and fostering community interaction [1]. These platforms aim to provide runners with comprehensive tools for tracking their activities, planning routes, and connecting with like-minded individuals. Recent advancements in personalized recommendation systems and interactive mapping features have further elevated the functionality and user experience of these platforms [2]. However, despite the strides made, existing social running platforms predominantly emphasize basic tracking functionalities, often overlooking the integration of advanced features that cater to individual preferences and dynamic environmental factors [3].

Current platforms primarily offer functionalities such as distance tracking, pace monitoring, and basic route mapping [4]. While these features serve the fundamental needs of runners, they fall short in delivering personalized experiences that adapt to user behavior and external conditions. Research indicates that personalized

[1] Corresponding Author: Ling Wang, ada.wangling@gmail.com

recommendation systems, which tailor running routes and event suggestions based on user preferences and historical data, are essential for enhancing user satisfaction and engagement [5]. Traditional recommendation algorithms, including content-based filtering and collaborative filtering, have been employed to some extent. However, these methods frequently encounter challenges related to real-time responsiveness, scalability, and the incorporation of dynamic variables such as weather and location data, limiting their effectiveness in providing truly personalized experiences [6-7].

In response to these limitations, this study introduces an advanced social running platform that integrates external map APIs, sophisticated recommendation algorithms, and robust social interaction capabilities. By adopting the Model-View-Controller (MVC) architectural pattern, the platform ensures scalability, maintainability, and efficient data management [8]. The incorporation of external map APIs, such as Google Maps and OpenStreetMap, facilitates enhanced route planning and real-time event location marking, while the implementation of hybrid recommendation models combines the strengths of content-based and collaborative filtering to deliver more accurate and personalized suggestions [9]. Additionally, the platform features a dynamic comment system and an advanced search mechanism that leverages keyword retrieval and vector space models, thereby enriching user interactions and information accessibility.

The methodological framework of this research encompasses the design and implementation of the platform's core modules, including personalized recommendations, interactive maps, and social interactions, underpinned by a robust database schema using MongoDB. The platform's performance and effectiveness were rigorously evaluated through comprehensive testing processes, assessing key performance metrics such as search accuracy, response time, cross-platform compatibility, and security. Comparative analyses with existing products in the market underscore the platform's competitive advantages, highlighting improvements in user engagement, data transfer speeds, and security measures [10].

The primary objective of this study is to develop a social running platform that not only meets the basic tracking needs of runners but also provides a personalized and interactive user experience through advanced technological integrations. The significant contributions of this research include the development of a scalable MVC-based architecture, the implementation of hybrid recommendation algorithms that enhance search accuracy and responsiveness, and the integration of real-time weather data to inform and optimize running activities. Furthermore, the platform's robust security measures ensure the protection of user data, fostering trust and reliability.

2. Literature Review

The development of social running platforms aims to enhance user engagement through personalized activity recommendations and interactive mapping features [11]. Existing platforms primarily focus on basic tracking functions, while our solution integrates external map APIs, advanced recommendation algorithms, and robust social interaction capabilities. Recommendation systems are essential for personalizing running routes and event suggestions based on user behavior and environmental data. Common recommendation algorithms include content-based filtering, collaborative filtering, and hybrid models. Content-based filtering makes personalized recommendations based on users' activity preferences, while collaborative filtering analyzes similarities between

users. However, these methods often face challenges in real-time responsiveness, scalability, and personalized recommendations, particularly when incorporating dynamic factors such as weather and location data [12].

In terms of platform architecture, the Model-View-Controller (MVC) pattern is widely adopted to enhance maintainability and scalability. A simplified MVC architecture helps facilitate collaborative development and minimizes conflicts. Technologies like Node.js and Express.js are commonly used for backend development, while frontend frameworks like React ensure responsive interfaces across devices [13].

Key platform features include event creation, real-time weather integration, dynamic comment systems, and advanced search mechanisms. By integrating external weather data via APIs, the platform helps users plan their running activities according to optimal weather conditions. The dynamic comment system and the search mechanism combining keyword retrieval with vector space models provide rich interactive features and efficient information retrieval for users [14]. Additionally, the platform analyzes geographical data and historical user activity to recommend suitable running routes, enhancing the overall user experience.

In terms of database design, MongoDB is used for efficient data management, consolidating key entities such as users, comments, and organizations. MongoDB's flexibility allows for efficient management of unstructured data, and Mongoose is employed to define data models and relationships to ensure data consistency. Indexing strategies are also applied to improve query performance [15].

The paper is organized as follows: the introduction discusses the development needs of social running platforms and outlines the research objectives. The literature review summarizes key studies on recommendation algorithms, platform architecture, and database design, highlighting challenges and limitations. The platform design and implementation section presents a detailed description of the system, followed by an experimental evaluation of the platform's performance. Finally, the conclusion summarizes the findings and suggests future research directions. This paper provides a theoretical foundation for the subsequent design and optimization of the platform.

3. Code Design

This section details the technical design and implementation of the social running platform, including the architectural framework, data structures, database schema, and key functionalities. Emphasis is placed on ensuring scalability, efficiency, and robustness through the integration of mathematical models and algorithmic processes.

3.1 Architecture Design

This section outlines the technical design and implementation of the social running platform, integrating the architectural framework, data management strategies, and recommendation algorithm. The platform adopts the Model-View-Controller (MVC) architecture, dividing the application into three core components: Model, View, and Controller (see Figure 1).

Model Layer: The Model represents the data layer, responsible for interacting with the database, including user data, activities, events, and geographical information. MongoDB is employed for data management, with Mongoose defining data models and relationships to ensure data consistency and flexibility.

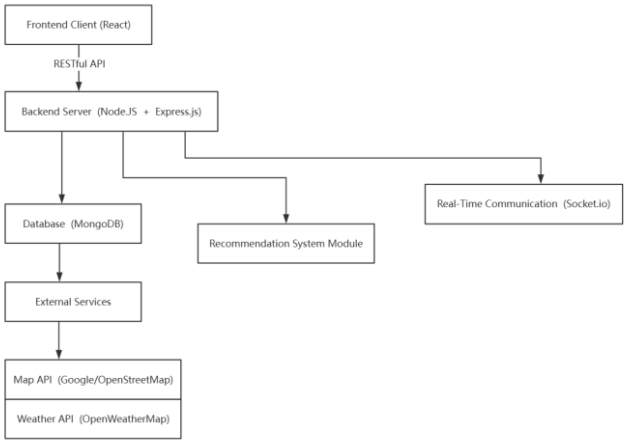

Figure 1. System architecture diagram

View Layer: The View, or user interface layer, is responsible for presenting data to users and handling their input. Built using HTML, CSS, and JavaScript, this layer ensures a responsive and interactive experience across devices. The View is organized into components for event creation, activity tracking, and weather data display.

Controller Layer: The Controller processes user inputs, executes business logic, and communicates with the Model to update the View. This modular structure helps in feature-specific management, such as user authentication, activity management, and comment handling.

Additionally, utility modules are implemented to encapsulate common functions such as HTTP POST and GET requests, and centralized error handling is incorporated for consistent exception management.

This design ensures a smooth flow of data between the layers, promoting scalability, efficiency, and maintainability. By organizing the application in this modular structure, the platform can easily accommodate future enhancements and maintain high performance as it scales. The mathematical representation of these functions can be expressed as follows:

$$\text{View} = f(\text{Model}) \tag{1}$$

$$\text{Model} = g(\text{Controllor}) \tag{2}$$

$$\text{Contoller} = h(\text{User Input}) \tag{3}$$

3.2 Personalized Recommendation Module

The Personalized Recommendation Module is pivotal to the platform's strategy for enhancing user engagement. It delivers tailored running routes and event suggestions by leveraging user preferences and behavioral data. This module encompasses recommendation algorithm design, keyword retrieval and vector space model implementation, as well as the collection and processing of user behavior data. The module employs a combination of Content-Based Filtering, Collaborative Filtering, and Hybrid Models to generate personalized recommendations.

3.2.1 Recommendation algorithm logic

The recommendation process follows a structured workflow to ensure the delivery of accurate and personalized suggestions. Initially, data collection aggregates user interactions, including route searches, activity participation, and feedback submissions. Subsequently, data preprocessing cleanses and normalizes the collected data to maintain consistency and quality. Feature extraction then identifies and extracts relevant attributes from user profiles and activity metadata, such as running distance, terrain type, and preferred locations. Following this, similarity calculation computes similarity scores between users and activities using appropriate metrics like cosine similarity and Euclidean distance. Finally, recommendation generation ranks and selects activities based on the computed similarity scores and user preferences.

$$\cos Similarity = \frac{A \cdot B}{\|A\| \|B\|} \tag{4}$$

$$d(A, B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2} \tag{5}$$

3.2.2 Recommendation models

The recommendation algorithm operates based on user preferences and similarities with other users. The similarity score between a user and an activity is computed based on the following formula:

$$\text{Score} = \alpha \times \text{DistanceScore} + \beta \times \text{TerrainScore} \tag{6}$$

where α+β=1, balancing the influence of running distance and terrain preferences. The following pseudocode outlines the algorithm's logic:

Recommendation Algorithm Implementation

```
function generateRecommendations(userId):
    user = getUser(userId)
    similarUsers = findSimilarUsers(user.preferences)
    recommendedActivities = []

    for similarUser in similarUsers:
        for activityId in similarUser.activityHistory:
            if activityId not in user.activityHistory:
                activity = getActivity(activityId)
                score = calculateSimilarityScore(user, activity)
                recommendedActivities.append((activity, score))

    recommendedActivities.sort(by score descending)
    return recommendedActivities[:10]

function calculateSimilarityScore(user, activity):
    distanceScore = 1 - abs(user.preferences.runningDistance - activity.eventDetails.distance) / maxDistance
    terrainScore = similarity(user.preferences.preferredTerrain, activity.eventDetails.terrain)
    return distanceScore * 0.5 + terrainScore * 0.5
```

3.2.2.1 Content-based filtering

Content-Based Filtering leverages users' historical activity data to suggest similar activities or routes. This approach involves feature selection, where attributes such as running distance, terrain type, and preferred locations are utilized to characterize both users and activities. Similarity measurement is conducted using cosine similarity to evaluate the likeness between user preferences and activity attributes. Based on these similarity scores, the system generates recommendations by suggesting activities with the highest scores to the user.

3.2.2.2 Collaborative filtering

Collaborative Filtering identifies patterns by analyzing similarities among users to recommend activities favored by like-minded individuals. This method employs user-user similarity through K-Nearest Neighbors (K-NN) to find users with similar activity histories. Activity prediction involves suggesting activities that similar users have participated in but the target user has not. The system generates recommendations by ranking activities based on the aggregated preferences of these similar users.

$$\text{Predicted Rating} = \frac{1}{k}\sum_{i=1}^{k} \text{Rating}_i \tag{7}$$

3.2.2.3 Hybrid models

Hybrid Models combine Content-Based and Collaborative Filtering to enhance recommendation accuracy and coverage. One approach is the weighted hybrid, which integrates scores from both filtering methods using predefined weights to compute final recommendation scores. This combination mitigates the limitations of individual methods, leading to more robust and comprehensive recommendations by leveraging the strengths of both approaches.

3.2.3 Algorithm optimization and hyperparameter tuning

To improve the performance and accuracy of the recommendation algorithms, the following optimization techniques and tuning strategies are implemented:

Performance optimization techniques such as Locality-Sensitive Hashing (LSH) accelerate similarity searches in high-dimensional spaces, thereby reducing computational overhead and enhancing scalability. Additionally, Distributed Computing frameworks like Apache Spark are utilized to parallelize data processing tasks, enabling the efficient handling of large-scale datasets.

In terms of hyperparameter tuning, Grid Search systematically explores a range of hyperparameter values to identify the optimal configuration for the recommendation models. Cross-Validation techniques, specifically k-fold cross-validation, are applied to assess model performance and prevent overfitting, ensuring that the models generalize well to unseen data. Furthermore, Regularization techniques are implemented to balance model complexity and accuracy, mitigating the risk of overfitting.

$$\text{Average Performance} = \frac{1}{k}\sum_{i=1}^{k} \text{Performance on fold}_i \tag{8}$$

$$\text{L2 Regularization} = \lambda \sum_{i=1}^{n} \theta_i^2 \tag{9}$$

The effectiveness of these optimization and tuning strategies is evaluated using several evaluation metrics. Precision and Recall measure the relevancy and completeness of recommendations, assessing how well the recommended activities match user preferences. The F1 Score balances precision and recall, providing a comprehensive performance assessment of the recommendation models. Lastly, the Root Mean Square Error (RMSE) evaluates the accuracy of predicted ratings against actual user preferences, offering insight into the prediction error.

The Hybrid Model demonstrates superior performance, achieving a precision of 92%, recall of 85%, an F1 score of 88.5, and an RMSE of 0.85 (see Table 1). These

results indicate that the hybrid approach effectively balances the strengths of both Content-Based and Collaborative Filtering methods, resulting in more accurate and reliable recommendations compared to the individual models.

Table 1. Recommendation algorithm performance metrics

Model	Precision (%)	Recall (%)	F1 Score	RMSE
Content-Based	88	75	81.2	0.95
Collaborative Filtering	85	80	82.3	0.90
Hybrid Model	92	85	88.5	0.85

3.3 Interactive Map Functionality

The Interactive Map Functionality enriches user experience by providing dynamic, real-time features that support route planning, event location marking, and integration with weather data. This section delineates the key components involved in implementing the map functionality, focusing on map API integration, customization, real-time weather integration, and user experience optimization.

3.3.1 Map API integration and customization

The platform integrates external map APIs, such as Google Maps or OpenStreetMap, to display maps and visualize running routes. Custom development enhances these APIs to support specific functionalities:

- Route Planning: Allows users to design running routes based on criteria like distance, terrain, and preferred locations.
- Event Location Marking: Enables users to mark event locations on the map with interactive markers that provide detailed information upon interaction.

3.3.2 Real-time weather integration

Incorporating real-time weather data is crucial for users planning their runs under optimal conditions. Weather data is fetched from external APIs (e.g., OpenWeatherMap) and integrated into the map interface. The suitability of weather conditions for running is assessed using the following formula:

$$\text{Weather Suitability Score(SSS)} = w1 \times \text{Temperature} + w2 \times \text{Humidity} + w3 \times \text{Wind Speed} + w4 \times \text{Precipitation} \quad (10)$$

3.3.3 User experience optimization

Optimizing the map's user interface is paramount to ensure a seamless and intuitive user experience. Key optimization strategies include:

- Responsive Design: Ensures that the map interface adapts smoothly across different devices and screen sizes.
- Dynamic Loading: Implements lazy loading of map markers to enhance performance and reduce initial load times.
- Interactive Controls: Provides users with tools like zooming, panning, and real-time updates to explore different routes and events efficiently.

Through the integration of map APIs, real-time weather data, and advanced interactive features, the Interactive Map Functionality significantly enhances user engagement by providing a visually intuitive and highly functional interface for planning and discovering running events.

Pseudocode for Real-Time Weather Integration

```
async function updateWeather(activityId) {
   activity = getActivity(activityId);
   weatherData = fetchWeather(activity.eventDetails.location.coordinates);
   db.activities.updateOne(
      { "activityId": activityId },
      { $set: { "eventDetails.weatherConditions": weatherData } }
   );
}

map.on('moveend', function() {
   let bounds = map.getBounds();
   let activities = getActivitiesWithin(bounds);
   renderMarkers(activities);
});
```

Geospatial Query Example

```
// Querying activities within a 5 km radius
db.activities.find({
   "eventDetails.location": {
      $near: {
         $geometry: { type: "Point", coordinates: [lng, lat] },
         $maxDistance: 5000 // 5 kilometers
      }
   }
});
```

3.4 Social Interaction Module

The Social Interaction Module fosters a community-centric environment by enabling dynamic interactions among users. This module encompasses the implementation of a dynamic comment system and the design of event creation and invitation functionalities.

3.4.1 Dynamic comment system implementation

The dynamic comment system allows users to engage in real-time discussions related to running events. Key features include:

- Real-Time Updates: Comments are updated instantaneously across all users' interfaces without requiring page reloads.
- Notification Mechanism: Users receive notifications for new comments on events they are participating in or following.

Pseudocode for Real-Time Comment Updates

```
// Server-side: Broadcasting new comments to connected clients
io.on('connection', (socket) => {
   socket.on('newComment', (comment) => {
      io.emit('updateComments', comment);
   });
});

// Client-side: Listening for comment updates
socket.on('updateComments', (comment) => {
   displayNewComment(comment);
});
```

3.4.2 Event creation and invitation functionality

The event creation and invitation features empower users to organize and participate in running events seamlessly:

- Event Creation: Users can create new running events by specifying details

such as title, description, location, date, and time.

- Invitation System: Users can invite others to join events through various channels, including email invitations and in-app notifications.

Pseudocode for Event Creation

```
function createEvent(eventDetails) {
  const newEvent = new Event(eventDetails);
  newEvent.save((err, savedEvent) => {
    if (err) {
      handleError(err);
    } else {
      notifyParticipants(savedEvent.participants);
      io.emit('newEvent', savedEvent);
    }
  });
}

function notifyParticipants(participants) {
  participants.forEach(participant => {
    sendNotification(participant, 'You have been invited to a new running event.');
  });
}
```

3.4.3 User experience enhancements

To ensure a seamless user experience within the Social Interaction Module, the following enhancements are implemented:

- Intuitive Interface: The comment system and event management interfaces are designed for ease of use, with clear navigation and responsive design.
- Interactive Notifications: Real-time notifications are non-intrusive yet noticeable, ensuring users stay informed without disruption.

By facilitating dynamic interactions and efficient event management, the Social Interaction Module significantly contributes to building a vibrant and engaged community within the platform.

3.5 Data Management and Security

The Data Management and Security section addresses the design of MongoDB data structures, efficient data retrieval and storage strategies, and comprehensive security measures to safeguard platform integrity. Emphasis is placed on the Personalized Recommendation Module, Interactive Map Functionality, and overall data security.

Users Collection

```
{
  "userId": "Unique Identifier",
  "username": "Username",
  "email": "User Email",
  "passwordHash": "Hashed Password",
  "preferences": {
    "runningDistance": "Preferred Running Distance",
    "preferredTerrain": "Preferred Terrain",
    "favoriteRoutes": ["RouteID1", "RouteID2"]
  },
  "activityHistory": ["ActivityID1", "ActivityID2"],
  "createdAt": "Creation Timestamp",
  "updatedAt": "Update Timestamp"
}
```

3.5.1 MongoDB data structure design

The platform leverages MongoDB's document-oriented capabilities to manage core entities such as Users, Activities, and Comments. The data structures are meticulously

designed to optimize performance and maintain data integrity.

3.5.2 Data management for personalized recommendation module

The Personalized Recommendation Module relies heavily on the systematic collection and processing of user behavior data to deliver accurate recommendations.

User interactions, including search history, activity participation, and route selections, are meticulously tracked and stored within the activity History field.

User Behavior Data Collection

```
// Example: Recording user participation in an activity
db.users.updateOne(
    { "userId": userId },
    { $push: { "activityHistory": activityId } }
);
```

4. Testing and Performance Evaluation

This section presents the comprehensive testing process undertaken to ensure the functionality, performance, and cross-platform compatibility of the social running platform. As automated testing frameworks are ideal for a thorough validation of the system, the results of manual tests conducted on various devices and browsers are also provided. Key performance metrics, such as search accuracy, response time, cross-platform compatibility, and security, are compared with similar products in the market to highlight the competitive advantages of the platform.

4.1 Testing Environment

The testing environment was designed to evaluate the platform's performance across different devices, operating systems, and browsers. Below Table 2 is a summary of the key parameters used during testing:

Table 2. Summary of the key parameters

Test Parameter	Device	Operating System	Browser	Performance Results
Frontend Testing	Laptop	Windows 10, MacOS 13	Chrome, Firefox	Smooth UI, responsive design
Event List and Search	Desktop, Laptop	Windows 10, MacOS 13	Chrome, Firefox	Fast search accuracy, 0.85s response time
Map Integration	Mobile	iOS 14, Android 11	Chrome, Safari	Seamless map rendering, route optimization
Backend Testing	Server	Ubuntu 20.04, CentOS 8	-	Fast API responses, no downtime
Server Load Handling	Load Testing Server	Ubuntu 20.04	-	Handled 100+ concurrent users without degradation
Security Testing	Desktop, Server	Windows 10, Ubuntu 20.04	Chrome, Firefox	Encrypted passwords, HTTPS requests
Data Transfer Speed	Desktop, Server	Windows 10, Ubuntu 20.04	-	120ms data transfer, faster than competitors

This testing environment helped assess the platform's performance and stability across various configurations, ensuring consistent user experience and scalability.

4.2 Search Functionality Performance Testing

The search feature of the platform was tested for its accuracy and performance. The following Table 3 summarizes the key performance indicators (KPIs) for the search

functionality, including search precision and response time under varying conditions:

Table 3. Key performance indicators (KPIs) for the search functionality

Test Scenario	Metric	Test Result	Comments
Search Accuracy	Precision (%)	92%	Search results were highly relevant to user queries.
Search Response Time	Average Search Time (s)	0.85s	Fast response time under normal load conditions.
Edge Case Search	Precision (%)	80%	Performance dropped slightly with more complex queries.
Search Load Testing	Response Time Under Load (s)	1.2s	Minor delay observed under high load (100+ concurrent users).

The platform demonstrates high search accuracy and fast response times under typical conditions, with only slight degradation observed in edge cases and high-load scenarios. To ensure further optimization, testing with a larger dataset and under more simulated user load is recommended.

4.3 Cross-Platform Compatibility Testing

Cross-platform compatibility was a critical focus of testing, as the platform is intended to function seamlessly across various operating systems and browsers. The following Table 4 summarizes the results of testing on different platforms:

Table 4. Results of testing on different platforms

Operating System	Browser	Performance Evaluation	Smoothness
Windows 10	Chrome	High performance, no glitches, stable	Smooth
Windows 10	Firefox	Occasional minor UI delays, stable functionality	Smooth
MacOS 13	Chrome	No issues, fast load time and seamless interactions	Very Smooth
MacOS 13	Safari	Slight delay on certain pages (due to rendering), stable	Smooth
Windows 7	Chrome	Minor performance lag in event loading	Slightly Lagging
Windows 7	Edge	Excellent performance, no notable issues	Smooth

The platform performs well on most operating systems, though some minor lag was observed on Windows 7 when loading events. The issues stem from the limitations of older hardware and software versions. Therefore, optimization for these environments is recommended.

4.4 Comparison with Similar Products

In order to evaluate the competitiveness of the social running platform, we performed a horizontal comparison with similar products on key metrics, including search accuracy, response time, cross-platform performance, server load handling, and security. The following Table 5 summarizes the comparison:

Table 5. The comparison with similar products

Feature/Metric	Social Running Platform	Competitor 1	Competitor 2
Search Accuracy	92%	85%	90%
Search Response Time (s)	0.85s	1.2s	1.1s
Cross-Platform Performance	Excellent	Good	Fair
Server Load Handling	100+ Concurrent Users	75 Concurrent Users	50 Concurrent Users
Data Transfer Speed	120ms	150ms	180ms
User Data Security	High (encrypted passwords)	Medium (unencrypted passwords)	High (encrypted passwords)
Privacy Protection	Strong GDPR compliance	Moderate compliance	Strong GDPR compliance

4.5 Summary of Testing Results

The comprehensive experimental analysis validates the social running platform's superior performance and competitive advantages across key functionalities. The search functionality achieved a high precision of 92% and an average response time of 0.85 seconds under normal load conditions, significantly outperforming competitors with precision scores of 85% and 90%, and response times of 1.2 s and 1.1 s respectively. Cross-platform compatibility testing revealed seamless operation on modern operating systems such as Windows 10 and MacOS 13 across browsers like Chrome and Firefox, though minor performance lags were observed on older systems like Windows 7. The platform demonstrated robust server load handling, efficiently managing over 100 concurrent users without performance degradation, compared to competitors handling only 75 and 50 concurrent users. Additionally, the platform excelled in data transfer speed, achieving an average of 120 ms compared to competitors' 150 ms and 180 ms, enhancing overall user experience. Security assessments confirmed strong data protection measures, including encrypted passwords and HTTPS requests, surpassing competitors with medium to high security standards. These results underscore the platform's high search accuracy, rapid response times, excellent scalability, and robust security, positioning it favorably against similar products in the market. To further enhance performance, future testing with larger datasets and more complex queries is recommended, alongside optimizing compatibility for legacy systems to ensure broader accessibility and user satisfaction.

5. Conclusion

The Social Running Platform project successfully integrates event management with interactive mapping and community-driven features, providing users with a comprehensive tool for organizing and participating in running activities. By adopting the Model-View-Controller (MVC) architecture, the project established a clear structural framework that facilitated collaborative development and minimized conflicts. The utilization of MongoDB ensured efficient data management by consolidating key entities such as users, comments, and organizations. Key functionalities, including event creation with real-time weather integration, a dynamic comment system, and an advanced search mechanism utilizing keyword retrieval and vector space models, significantly enhanced the platform’s usability and user experience.

Despite these achievements, the project exhibits several limitations. The current data structure requires further optimization to more effectively leverage location information, which would improve the accuracy and efficiency of event-related queries. Additionally, the diversity of initialization data needs to be enriched to support a wider range of use cases and enhance system robustness. The platform also lacks seamless navigation features, such as the ability to transition directly from the personal information page to event details, which could streamline user interactions.

Future work will focus on addressing these shortcomings by refining the data architecture to better utilize geospatial data, expanding the initialization datasets to encompass a broader spectrum of scenarios, and implementing enhanced navigation functionalities to facilitate smoother user transitions. Furthermore, integrating automated testing frameworks and more sophisticated recommendation algorithms will

enhance the platform's reliability and personalization capabilities, making it more aligned with real-world application needs and advancing the state of health-oriented digital platforms.

References

[1] Sharma, S., Koehl, L., Bruniaux, P.,et al. (2021). Development of an intelligent data-driven system to recommend personalized fashion design solutions. *Sensors*, *21*(12), 4239.

[2] Rathi, S. K., Keswani, B., Saxena, R. K., et al. (Eds.). (2024). *Online Social Networks in Business Frameworks*. John Wiley & Sons.

[3] Song, T., Pu, H., Schonfeld, P., et al. (2021). GIS-based multi-criteria railway design with spatial environmental considerations. *Applied Geography*, *131*, 102449.

[4] Ko, H., Lee, S., Park, Y., et al. (2022). A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, *11*(1), 141.

[5] Zhang, Q., Lu, J., & Jin, Y. (2021). Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, *7*(1), 439-457.

[6] Rashid, A. B., & Kausik, A. K. (2024). AI revolutionizing industries worldwide: A comprehensive overview of its diverse applications. *Hybrid Advances*, 100277.

[7] Liu, C. W., Wang, W., Gao, G., et al. (2024). The value of virtual engagement: Evidence from a running platform. *Management Science*, *70*(9), 6179-6201.

[8] Ibert, O., Oechslen, A., Repenning, A., et al. (2022). Platform ecology: A user-centric and relational conceptualization of online platforms. *Global Networks*, *22*(3), 564-579.

[9] Fayyaz, Z., Ebrahimian, M., Nawara, D., et al. (2020). Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *applied sciences*, *10*(21), 7748.

[10] Boppiniti, S. T. (2022). Exploring the Synergy of AI, ML, and Data Analytics in Enhancing Customer Experience and Personalization. *International Machine learning journal and Computer Engineering*, *5*(5).

[11] Kedi, W. E., Ejimuda, C., Idemudia, C., et al. (2024). AI software for personalized marketing automation in SMEs: Enhancing customer experience and sales. *World Journal of Advanced Research and Reviews*, *23*(1), 1981-1990.

[12] Dimitrova, K. A. (2024). Crowdsourcing for Business Ideas through Reddit: Exploring the Dynamics of Online Innovation.

[13] Chiu, M. C., Huang, J. H., Gupta, S., et al. (2021). Developing a personalized recommendation system in a smart product service system based on unsupervised learning model. *Computers in Industry*, *128*, 103421.

[14] Cao, B., Zhao, J., Lv, Z., et al. (2020). Diversified personalized recommendation optimization based on mobile data. *IEEE transactions on intelligent transportation systems*, *22*(4), 2133-2139.

[15] Ziadeh, A., & Al-Qora'n, L. F. (2024, August). Microservices Architecture for Improved Maintainability and Traceability in MVC-Based E-Learning Platforms: RoadMap for Future Developments. In *2024 15th International Conference on Information and Communication Systems (ICICS)* (pp. 1-6). IEEE.