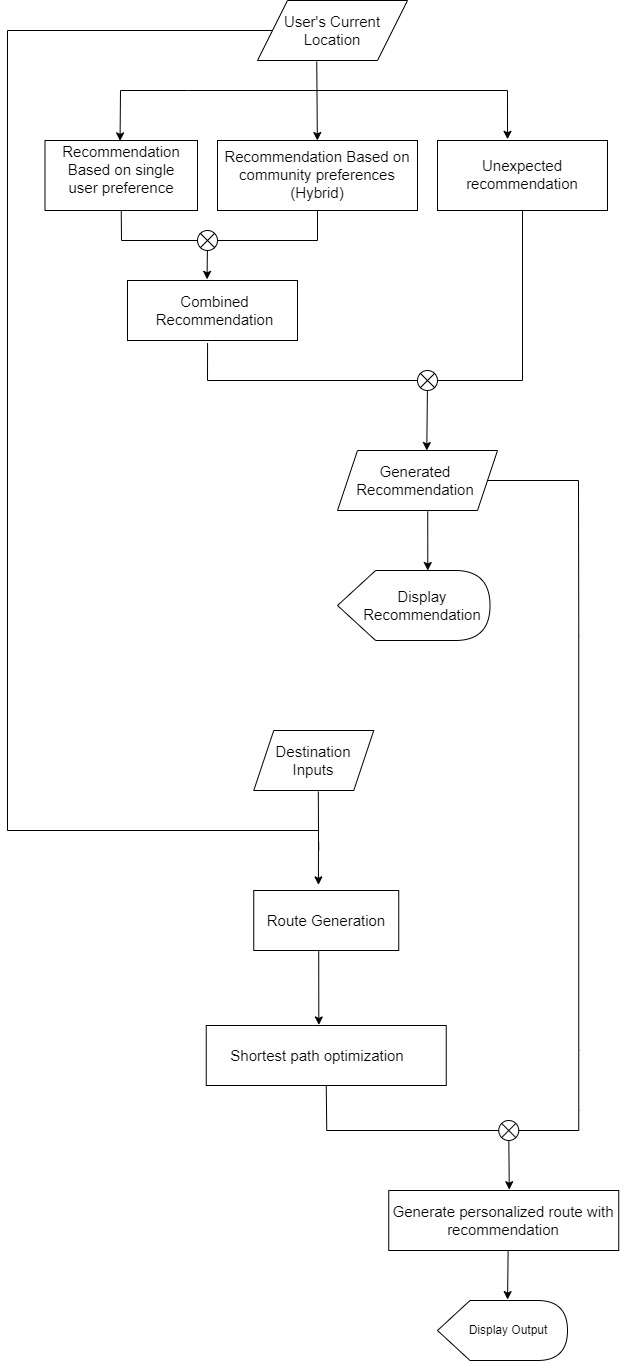
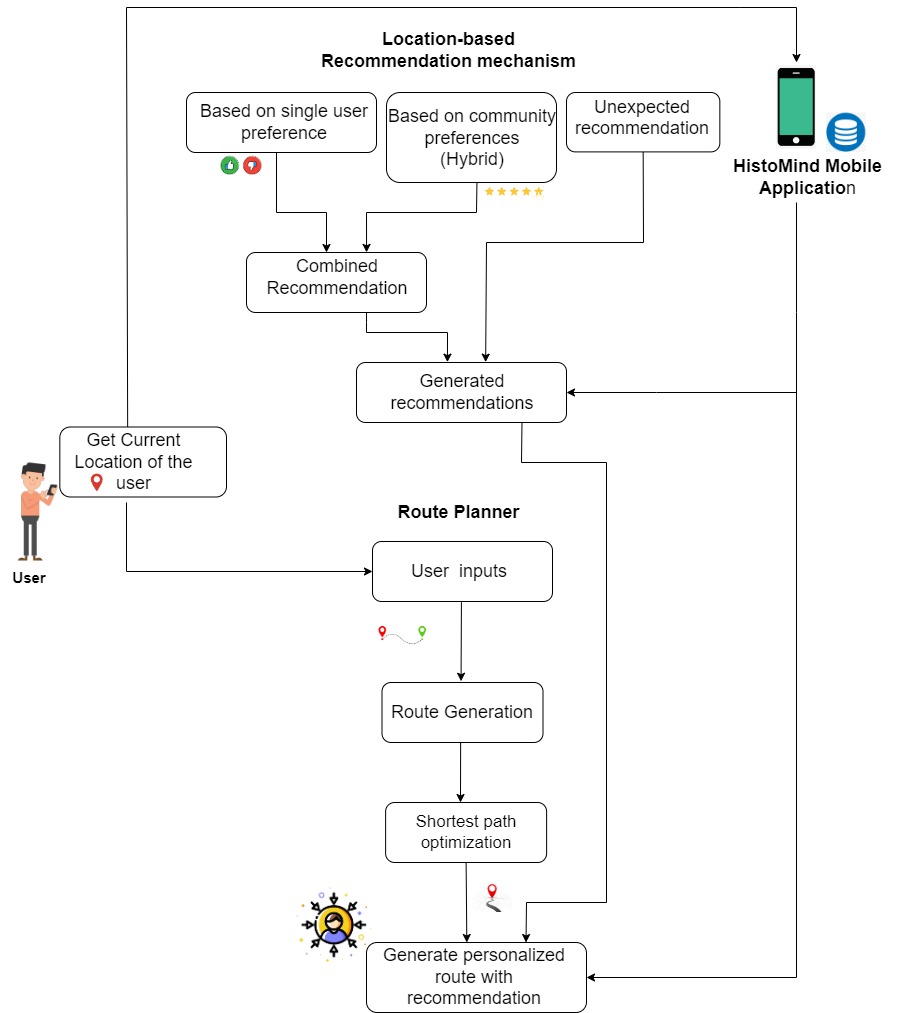
**Smart location-based recommendation mechanism with route planner.**

**Novelty**

*The novelty in the combined recommendation approach lies in the integration of multiple recommendation techniques and data sources, including individual user preferences, community preferences, and unexpected or lesser-known locations. By combining these diverse elements, the system introduces novelty by suggesting recommendations that align with the user's preferences, have social validation from the community, and include unique and unexpected places. The combination ensures a more comprehensive and intelligent recommendation experience, offering personalized suggestions while introducing new and exciting options that users may not have discovered on their own.*

Features

* Location -based recommendation based on single user preference.
* Location -based hybrid recommendation based on community preference. (Ratings/Reviews)
* Combined recommendation.
* Location -based unexpected recommendation.



# **Location -based recommendation based on single user preference.**

* **Collect user preference:** Gather information about the user's preferences related to historical places and tourist attractions. This can include their historical interests, preferred types of attractions, or any specific requirements they have when visiting such places.
* **Collect item data:** Gather data about historical places and tourist attractions in the recommendation system. This data should include attributes such as historical significance, category, visitor ratings, and geographic coordinates.
* **Build user profile:** Create a user profile based on the user's preferences. This profile should capture their preferred types of historical places, preferred locations for visits, and any specific requirements or constraints.
* **Develop a content-based recommendation algorithm:** Develop a recommendation algorithm that relies solely on the content or attributes of historical places and tourist attractions. This algorithm should consider the user's preferences and match them with the attributes of the items.
* **Model training:** In the model training step, train the content-based recommendation algorithm using the collected data.
* Preprocess the item data: Clean and preprocess the item data, including handling missing values, normalizing numeric attributes, and encoding categorical attributes.
* Feature extraction: Extract relevant features from the item data that are important for recommendation, such as historical significance, category, and geographic coordinates.
* Create user-item matrix: Represent the user's preference and the item features in a suitable format, such as a user-item matrix, where each row represents a user, and each column represents an item feature.
* Train the model: Apply a suitable machine learning algorithm, such as a similarity-based approach or a supervised learning model, to train the recommendation model using the user-item matrix. The model should learn to predict the user's preference or rank the items based on their relevance to the user's preferences.
* Evaluate the model: Assess the performance of the trained model using evaluation metrics such as precision, recall, or mean average precision. Can be used techniques like cross-validation to ensure the model's generalizability.
* Hyperparameter tuning: Fine-tune the model's hyperparameters, such as the similarity measure or regularization parameters, to optimize its performance.
* Save the trained model: Once the model training is complete, save the trained model for later use in the recommendation process.
* **Recommendation generation:** Using the trained model, generate recommendations for the user based on their preferences and the item attributes. The algorithm should rank the items based on their relevance to the user's preferences and present the top recommendations.
* **Evaluate and refine the system:** Measure the performance of the recommendation system using evaluation metrics such as precision, recall, or user satisfaction surveys. Continuously refine the algorithm based on user feedback and data analysis.
* **Integrate and deploy the system:** Integrate the recommendation system into the application or platform, ensuring it can handle real-time updates of user preferences and provide recommendations based on the user's input.
* **Monitor and iterate:** Continuously monitor the system's performance, collect user feedback, and iterate on the algorithm and data collection processes to improve the recommendations over time.

# **Location -based hybrid recommendation based on community preference. (Ratings/Reviews)**

**Collect ratings/reviews:** Gather ratings and reviews given by the community for historical places and tourist attractions. This data reflects the preferences and opinions of the community regarding these places.

**Collect item data**: Gather data about historical places and tourist attractions in the recommendation system. This data should include attributes such as historical significance, category, visitor ratings, geographic coordinates, and the ratings/reviews collected from the community.

**Build user profiles:** Create user profiles based on the ratings/reviews provided by individual users. These profiles can capture their preferences, patterns, and sentiment towards historical places and tourist attractions.

**Develop a hybrid recommendation algorithm:** Adapt the recommendation algorithm to consider the specific characteristics of historical places and tourist attractions, as well as the community preference based on ratings/reviews. Combine collaborative filtering (based on community ratings/reviews) with content-based filtering (based on location and historical attributes) to generate recommendations.

**Model training:** In the model training step, will train the hybrid recommendation algorithm using the collected data, including ratings/reviews, item attributes, and location information. Here's an expanded explanation of the model training process for the hybrid approach:

* Preprocess the item data: Clean and preprocess the item data, handle missing values, normalize numeric attributes, and encode categorical attributes.
* Feature extraction: Extract relevant features from the item data, such as historical significance, category, and geographic coordinates.
* Create user-item matrix: Represent the user's preference and the item features, including community ratings/reviews, in a suitable format, such as a user-item matrix.
* Train the collaborative filtering model: Apply collaborative filtering techniques to train a model based on the user-item matrix and community ratings/reviews. This can involve techniques like matrix factorization, neighborhood-based methods, or deep learning approaches. The model learns to predict user preferences or generate recommendations based on collaborative patterns.
* Train the content-based filtering model: Apply content-based filtering techniques to train a model that leverages location and historical attributes. This can involve machine learning algorithms such as decision trees, random forests, or neural networks. The model learns to predict item relevance or generate recommendations based on content similarity.
* Hybridization of collaborative and content-based models: Combine the outputs of the trained collaborative filtering model and the content-based filtering model using hybridization techniques such as weighted averaging, stacking, or ensemble methods. Experiment with different approaches to determine the best hybridization strategy.
* Evaluate the model: Assess the performance of the trained hybrid model using evaluation metrics such as precision, recall, or mean average precision. Use techniques like cross-validation to ensure the model's generalizability.
* Hyperparameter tuning: Fine-tuning the hyperparameters of the hybrid model, such as the weighting factors for the collaborative and content-based components, to optimize its performance. This can be done through techniques like grid search or Bayesian optimization.
* Save the trained model: Once the model training and tuning are complete, save the trained hybrid model for later use in the recommendation process.

**Recommendation generation:** Using the trained hybrid model, generate recommendations for users based on their preferences, location, and community ratings/reviews. The algorithm should rank the items based on their relevance to the user's preferences and the community's feedback.

**Evaluate and refine the system:** Measure the performance of the recommendation system using appropriate evaluation metrics and refine the algorithm based on user feedback and data analysis.

**Integrate and deploy the system:** Integrate the recommendation system into the application or platform, ensuring it can handle real-time updates of user preferences, ratings/reviews, and location data.

**Monitor and iterate:** Continuously monitor the system's performance, collect user feedback, and iterate on the algorithm and data collection processes to improve the recommendations over time.

# **Combined recommendation.**

*NOTE: USE either hybrid model or rule-based approach*

**Hybrid model approach**

* **Obtain recommendations from each approach:** Run both the location-based recommendation based on single user preference and the location-based hybrid recommendation based on community preference approaches separately to generate recommendations for the user.
* **Assign weights to the recommendations:** Assign weights to the recommendations from each approach based on their importance or reliability. These weights can be determined through experimentation, user feedback, or based on the performance of each approach.
* **Normalize the recommendation scores:** Normalize the recommendation scores from each approach to bring them to a common scale. This step ensures that the recommendations from different approaches are comparable and can be combined effectively.
* **Combine the recommendations using a hybrid model:** Train a hybrid model that combines the recommendations from different approaches. The hybrid model can be a machine learning model, such as an ensemble model, a neural network, or a combination of different algorithms.
* **Feature engineering:** Extract relevant features from the recommendations, such as the recommendation scores, weights, or any other relevant attributes. Additionally, can incorporate features related to novelty, diversity, or user preferences to enhance the hybrid model's performance.
* **Train the hybrid model:** Train the hybrid model using the extracted features and the normalized recommendation scores from each approach. The training process involves optimizing the model's parameters to accurately predict the final recommendation scores.
* **Evaluate the hybrid model:** Assess the performance of the trained hybrid model using appropriate evaluation metrics such as precision, recall, or mean average precision. Use techniques like cross-validation to ensure the model's generalizability.
* **Combine the recommendations:** Use the trained hybrid model to combine the normalized recommendation scores from each approach. The model will generate a final recommendation score that balances the contributions from the different approaches based on their weights.
* **Handle conflicts or contradictions:** Address any conflicts or contradictions that arise during the combination process. This can incorporate additional rules or logic to resolve conflicts, prioritize recommendations with higher novelty or diversity scores, or consider user feedback to make a final decision.
* **Rank and present the final recommendations:** Rank the combined recommendations based on the final recommendation scores and present them to the user. Ensure that the recommendations are fair, diverse, and aligned with the user's preferences, community feedback, and the desired level of novelty.
* **Evaluate and refine:** Continuously monitor the performance of the combined recommendation approach and collect user feedback. Refine the hybrid model, feature engineering, and weighting scheme based on user satisfaction and recommendation effectiveness.

Or

**Rule based approach.**

**Obtain recommendations from each approach**: Run both the location-based recommendation based on single user preference and the location-based hybrid recommendation based on community preference approaches separately to generate recommendations for the user.

**Assign weights to the recommendations:** Assign weights to the recommendations from each approach based on their importance or reliability. These weights can be determined through experimentation, user feedback, or based on the performance of each approach.

**Normalize the recommendation scores**: Normalize the recommendation scores from each approach to bring them to a common scale. This step ensures that the recommendations from different approaches are comparable and can be combined effectively.

**Define rules for combining recommendations:** Establish rules or logic that determine how to combine the recommendations. Consider factors such as the user's historical preferences, the frequency of recommendations, or the degree of deviation from the user's usual choices. These rules can prioritize novel recommendations, ensure a certain percentage of recommendations are novel, or handle conflicts between recommendations from different approaches.

**Combine the recommendations based on the defined rules:** Apply the defined rules to combine the normalized recommendation scores from each approach. The rules determine how the recommendations are weighted, selected, or adjusted to create the final combined recommendations.

**Rank and present the final recommendations:** Rank the combined recommendations based on the applied rules and present them to the user. Ensure that the recommendations are fair, diverse, and aligned with the user's preferences, community feedback, and the desired level of novelty.

**Evaluate and refine:** Continuously monitor the performance of the rule-based approach and collect user feedback. Refine the rules and weighting scheme based on user satisfaction and recommendation effectiveness. Iterate on the rules as necessary to enhance the combination of recommendations.

# **Location -based unexpected recommendation.**

**Collect location data:** Obtain the user's current location data through GPS, IP addresses, or manual input. This information will be used as the basis for finding nearby places.

**Acquire place data**: Gather a comprehensive dataset of places, attractions, or points of interest in the target area. This dataset should include information such as location coordinates, names, descriptions, and other relevant attributes.

**Filter popular or well-known places:** Identify and filter out popular or well-known places from the dataset. These are typically places that are widely recognized or frequently visited by tourists or locals.

**Determine proximity threshold:** Set a proximity threshold to define the radius around the user's location within which the system will search for nearby places. This threshold can be adjusted based on the desired level of exploration and the density of lesser-known places in the area.

**Calculate distance:** Calculate the distance between the user's location and each place in the filtered dataset using geographical coordinates or an appropriate distance metric.

**Rank places by proximity**: Rank the filtered places based on their proximity to the user's location. Sort the places in ascending order, starting from the closest ones.

**Introduce variety and uniqueness**: Apply algorithms or rules to prioritize and recommend places that are considered lesser-known, unique, or unconventional. This can be based on factors such as low visitor ratings, limited online presence, or uncommon attributes.

**Apply relevance filters**: Incorporate relevance filters to ensure that the recommended places align with the user's context and interests. For example, consider the user's demographic information, historical preferences, or past interactions with similar places.

**Present recommendations:** Select a subset of the ranked and filtered places as recommendations to the user. The number of recommendations can be determined based on system constraints or user preferences.

**Provide additional information:** Enhance the recommendations by providing additional information about each suggested place, such as descriptions, images, user reviews, or directions for reaching the location.

**Continuously update the dataset:** Regularly update the dataset of places to ensure the inclusion of new and emerging lesser-known locations. This can be done through data collection mechanisms, crowdsourcing, or integration with external data sources.

**2) Route Planner**

**Once the user selects locations, on the map , the shortest path should be optimized and should be integrated with the location based recommendation system.**