Smart Travel Planner: A mashup of travel-related web services

Rabia Jafri, Amal Saad Alkhunji, Ghada Khaled Alhader, Hanan Rabeiah Alrabeiah, Noura Abdullah Alhammad, Sara Khader Alzahrani

Department of Information Technology
King Saud University
Riyadh, Saudi Arabia

***Abstract*—The Smart Travel Planner is a web-based intelligent system for facilitating the travel planning process, which is implemented as a mashup of several travel-related services' APIs. Unlike other similar applications which essentially work as calendaring programs, our system is unique in that it integrates the information and functions needed for travel planning in their entirety within a single piece of software providing such features as checking hotel availability and calculating optimum routes between user-specified locations at various travel destinations, and offering suggestions for sightseeing, all the while making sure to take into consideration the user-specified needs for particular meeting days, times and locations and personal preferences for dining, relaxation and sleep. Furthermore, these services are not limited to a particular city but are offered for any city around the world for which such information exists within the databases of the services included in the mashup.**

Keywords—mashup; travel; web services; web applications; intelligent systems

I. INTRODUCTION

The emerging trend towards globalization in recent years has been accompanied by an unprecedented increase in both local and international travel characterized by a propensity to visit multiple places in the same trip. Consequently, creating a travel plan has become an ever more intimidating, complex and time-consuming process involving multiple steps – to name just a few: choosing among several transportation options, places to visit and hotels to stay in, searching for tourist attractions at the various destinations, creating daily schedules factoring in breaks for meals and rest, sifting through dining choices and –last, but not least - calculating routes with the least travel durations in order to make the most efficient use of the limited time available.

Myriad travel-related websites and web services have proliferated the internet in recent years to assist travelers in planning for their journeys. However, since these are so numerous, are generally disjoint from each other, usually deal with only one or two aspects of travel planning and offer limited customization options for querying and generating results, they have served only to exacerbate the problem: the traveler suffers from information overload, feels compelled to visit several sites for each travel-related decision and then has to manually consolidate the retrieved information and integrate it into an overall plan.

We have therefore, developed the Smart Travel Planner, a web-based intelligent system for facilitating the travel planning process, which is implemented as a mashup of several travel-related APIs. Unlike other similar applications which essentially work as calendaring programs, our system is unique in that it integrates the information and functions needed for travel planning in their entirety within a single piece of software and provides additional functionality by taking into account the users' preferences. It also permits them to block certain times and provides assistance in resolving scheduling conflicts. Moreover, it offers a route calculating function which is singular in its ability to provide users with not just the shortest route but customized suggestions based on their time specifications. Also, this system is not limited to a specific geographical domain but includes all cities around the world supported by Google Maps and other services included in the mashup.

The rest of this paper is organized as follows: Section 2 provides an overview of currently available applications for assisting travelers in planning their trips. Section 3 describes the architecture, implementation and functionalities of the Smart Travel Planner system. Section 4 concludes the paper and identifies several directions for future work.

II. RELATED WORK

The recent proliferation of specialized, publicly available data sources (e.g., geographic mapping data, weather information, etc.) accessible via light-weight APIs has stimulated the development of several applications that combine information from many of these sources in the form of mashups [1]. Consequently, in the domain of travel planning, too, several websites and applications have emerged to meet the growing needs for integrated information access, storage and organization in this area. One well–known example is Expedia [2], which offers localized travel websites for 29 countries and allows customers to book airline tickets, hotel reservations, car rentals, cruises, vacation packages and various other services via the World Wide Web and travel agents. Yahoo! Travel [3] is another prominent website which offers services similar to Expedia. TripIt [4] is an iOS application and a website which enables users to organize trip details into one master online itinerary — even if arrangements are booked at multiple travel sites – the user only need forward all confirmation emails to this service. It also allows users to customize their itineraries (by adding maps and directions, travel notes, photos, recommendations, etc.), to share itineraries with family and friends, and to get mobile alerts or emails about flight delays, cancellations, and gate changes. TravelTracker [5] is an iOS application that allows users to create and email itineraries and offers other options such as entering flight information and getting directions to a destination. It also has several add-ons, such as recording details of travel expenses and integration with TripIt [4].

In addition to the above commercial applications, several researchers are also actively exploring this area and devising innovative solutions. Examples include the following: Dang et al. [6] have developed STAAR (Semantic Tourist InformAtion, Access and Recommendation), an ontology based system that matches semantic and geographic enriched travel data with tourist preferences to search and recommend information. It also incorporates an algorithm for recommending travel routes based on two criteria: itinerary length and user interests. Wenyun et al. [7] have implemented a system that gathers information from various departments related to tourism (i.e., railway, marine, aerial, booking, vehicles, hotels and maps), then reintegrates it and presents it to the customer via a unified browser interface. Navabpour et al.'s [8] application allows users to find the cheapest tickets on the transportation system they prefer (the options include plane, train and bus). It also enables them to make hotel reservations at their destinations and provides an intelligent service which generates all possible combinations of various transportation methods and returns the cheapest mix as a result. Govardhan and Feuerlicht's [1] "Itinerary Planner Mashup" allows users to create an itinerary of the destinations that they plan to visit and displays these on a map. Users can also display additional information (such as weather and forecast data) about each destination.

Despite their usefulness, all these applications lack at least some of the following features: the ability to create user profiles (e.g., Expedia and Yahoo! Travel), view places of interest near the user's destination or current location (e.g., Expedia, Yahoo! Travel, TripIt and TravelTracker), calculate routes (e.g., Expedia and Yahoo! Travel), or track the user's travel history (e.g., TripIt and TravelTracker). Also, none of these offer an option for blocking personal time slots in the travel schedule. Our system provides all these features as well as other functionalities, which we will now proceed to describe in detail.

III. Smart Travel Planner: Functionalities and Implementation

In order to facilitate the process of organizing one's journey, we have developed the Smart Travel Planner, a web-based intelligent system that provides the following main functionalities:

- Allows the user to register, create a profile, update it and set his preferences (e.g., meal types, activities, etc.).
- Allows the user to create an itinerary and add different city plans to it. For each city plan, the user specifies the name of the city and the arrival and departure dates and times. Various locations can then be added to the city plan where the date and time for visiting each location is explicated; the user can edit the locations, city plans and itineraries.
- Allows the user to block certain times in which he does not want anything to be scheduled. These times can be blocked on a daily or weekly basis or for a specific date.
- Allows the user to view information about the cities in the itinerary by automatically providing links to relevant information in Wikipedia.
- Detects and resolves scheduling conflicts.
- Calculates the best routes for visiting all the places specified by the user in each city. There are two options available: The first option allows the user to simply specify the places that he wants to visit and the amount of time that he may want to spend at each place; the system then calculates the shortest route for visiting those places. The second option allows the user not only to specify the places that he wants to visit but also the exact time at which he wants to arrive at each place. The system then calculates the best route among those places and informs the user at what times he needs to leave each place in order to arrive at the next place at the specified time. If any conflicts occur due to overlapping time requests by the user or the user not allowing ample travel time between two locations, in either case, the user is warned about the conflicts and allowed to resolve them either manually or automatically.
- Displays suggestions for hotels, restaurants, attractions and meeting locations based on user preferences and allows these to be added to a city plan.

The main functionalities of our system are shown in the use case diagram in Figure 1. Screenshots of the interfaces for adding a user-requested location to the city plan, adding a location suggested by the system to a city plan and calculating the route among several user-specified places are shown in Figures 2, 3 and 4, respectively.

Our system is a mashup of the following travel-related APIs: Google Maps API [9], Google Places API [10], Expedia API [2] and Wikipedia API [11]. It was implemented using Microsoft Visual Studio 2010 [12] employing the ASP.NET framework. C# and JavaScript were the programming languages utilized. Microsoft SQL server 2008 [13] tools were used to set up the local server and manage the database which holds all the users' information for future retrieval and manipulation. SOAP web services were utilized to retrieve data from the Expedia API while Javascript was used for this purpose for the other APIs. Route calculation was based on Dijkstra's algorithm [14]. A more detailed explanation of how some the individual functions were implemented is provided below.

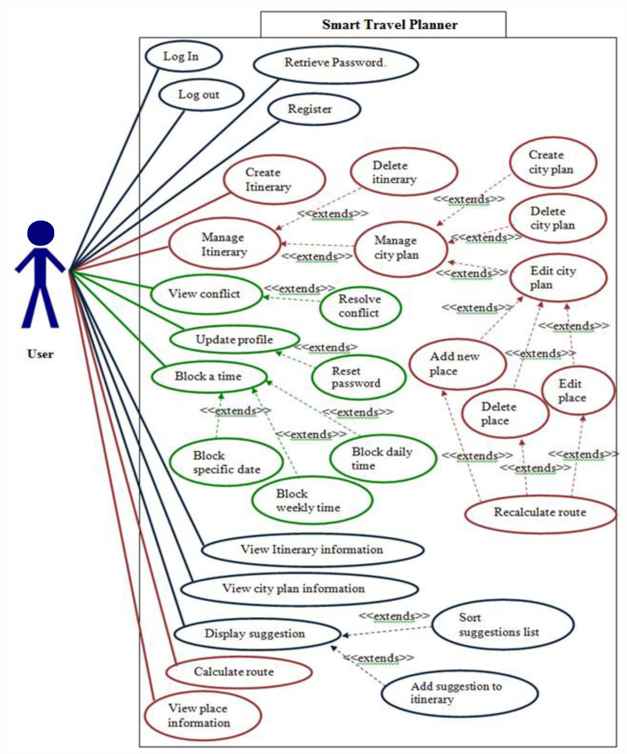
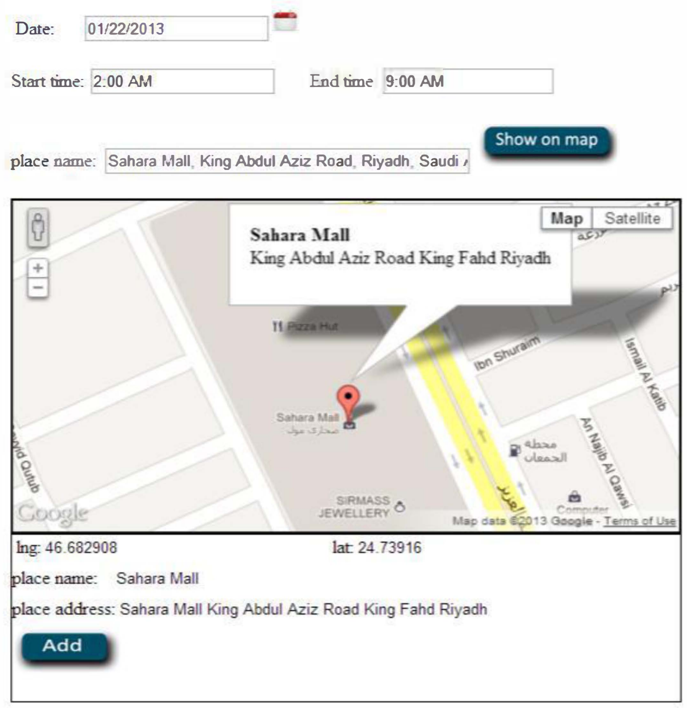


Fig. 1. Use cases that represent the functionalities of the Smart Travel Planner system.

The “Add Place” function, which allows the user to add a new place to the city plan, was implemented using JavaScript to invoke the Google Places API. The user has to enter a name, date, and start and end time for each place (Fig. 2). The system uses Google Places API to provide auto-complete functionality in order to avoid spelling mistakes while entering the place names; it also returns place information (address, latitude and longitude, etc.). This information is then used to display the place on the map and store it in the database. Prior to adding the place to the database, the dates and times are checked for possible conflicts with other already scheduled times/places.

The “Hotel Suggestions” function was implemented using C# with the Expedia API which is called via SOAP web service using the WSDL file. The “Suggested List of Food Places” was implemented using JavaScript to invoke the Google Places API with a request containing keywords (user food preferences, in this case) and radius information. Google Places API then returns all the restaurants/food places that match the keywords in the specified radius and city (Fig. 3). In addition, there is a “Suggested List of Tourist Attractions” for which the request takes the type of attraction (mall, aquarium, museum, etc.) and the radius to provide a list of attractions that satisfy the specified criteria.

One of the most important functions of the system is the “Calculate Route” function which calculates the optimum routes for visiting all the places specified by the user in each city based on two options as described above. For both options, the information about the places is retrieved from the database. The system then uses the Google Maps API and Google


Date: 01/22/2013

Start time: 2:00 AM End time 9:00 AM

place name: Sahara Mall, King Abdul Aziz Road, Riyadh, Saudi A

Sahara Mall
King Abdul Aziz Road King Fahd Riyadh

lng: 46.682908 lat: 24.73916

place name: Sahara Mall

place address: Sahara Mall King Abdul Aziz Road King Fahd Riyadh

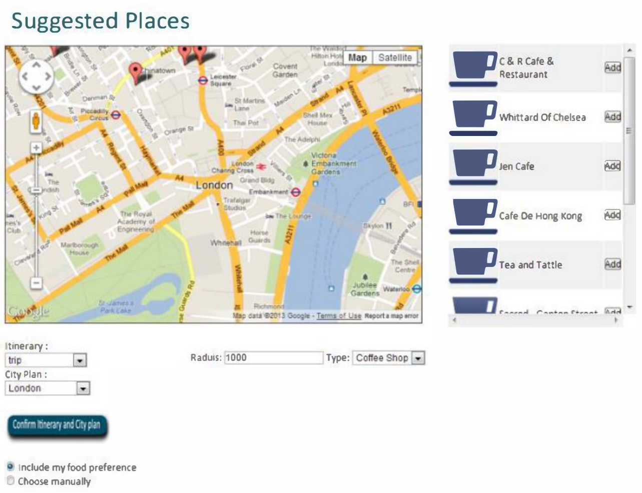
Fig. 2. Interface for adding a user-requested location to a city plan.

Distance Matrix API services to compute the distance and travel duration between the respective places. The best path is then calculated using a modified version of Dijkstra's algorithm [14] (described below) which is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edges and is frequently used in routing problems. Finally, the calculated path is displayed on the map using the Google Maps API Directions Service (Fig. 4).

The modified Dijkstra's algorithm works as follows:

(*Note: Since we utilize the Google Distance Matrix API service which provides the Least Distance path between every pair of nodes, it can be guaranteed that there is only one edge from each node to all other nodes.*)

1. Calculate the distance between the places (from a Google API Distance Matrix).
2. Mark all nodes as unvisited. Set the initial node (start place) as the current node. Create a set of the unvisited nodes called *the unvisited set* consisting of all the nodes except the initial node (start place) and the destination node (destination place).
3. For the current node, consider all of its unvisited neighbors and search for the nearest one.
4. Mark the current node as visited; add it to the result tree and remove it from the unvisited set. Then set the current node to the nearest node.
5. If the unvisited set is not empty, go back to step 3; otherwise, go to the next step (step 6).


Suggested Places

C & R Cafe & Restaurant Add

Whittard Of Chelsea Add

Jen Cafe Add

Cafe De Hong Kong Add

Tea and Tattle Add

Itinerary : trip

City Plan : London

Radius: 1000 Type: Coffee Shop

Confirm Itinerary and City plan

[✓ Include my food preference]

Choose manually

Fig. 3. Interface for adding a coffee shop suggested by the system to a city plan.

6. For the destination node, add all of its neighbors, except the initial node (start place), to a set called *the unfit set*.

7. Search the unfit set for the farthest node from the destination node.

8. Change the position of the farthest node in the result tree by making it as a child to its nearest neighbors and as a parent for the old child and remove it from the unfit set.

9. If the unfit set is not empty, go back to step 7; otherwise, go to the next step (step 10).

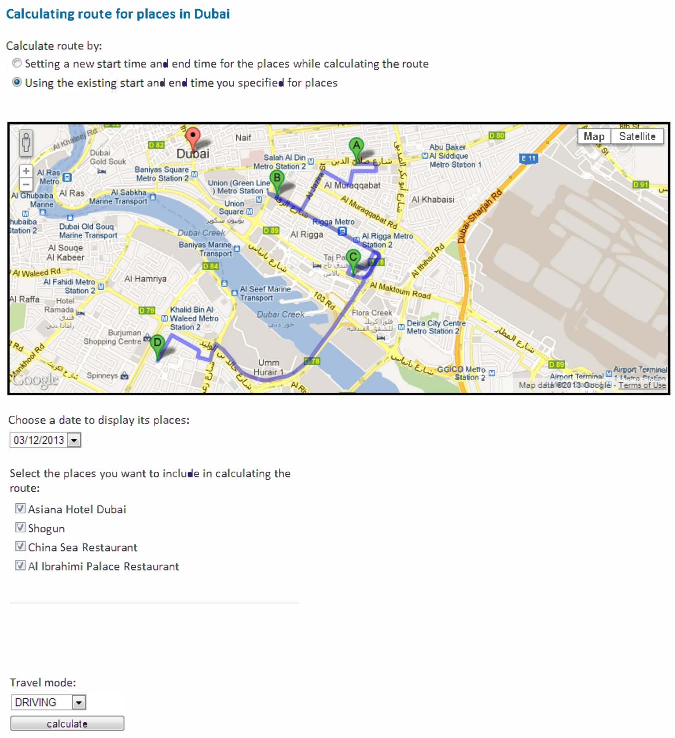
10. The result tree shows the shortest path between the start place and the destination place.

Some implementation difficulties:

A brief overview of some of the problems encountered during the implementation of this system and their resolution is provided here.

Since the Google Maps API service is invoked through the JavaScript code, in our initial implementation, the place data was being passed directly from the Javascript to the database, which we realized later was a security risk. We, therefore, modified the code so that the data is passed from the Javascript to a C# routine, which eventually writes it to the database.

We were unable to find an API available for free that would provide us with information about countries and the states/provinces and cities within them. We did find a database maintained by Google containing an accurate list of cities, states/provinces and countries. However, since this database could not be exported, we had to manually run SQL script to obtain data from this database to construct our own database of city, state/province and country names. A subsequent issue was preventing the page from refreshing when the user chose a country to load its associated states and again, to load the cities associated with those states. Our solution was to create a web service which utilized AJAX to asynchronously (without refreshing the page) update the list of states and cities retrieved


Calculating route for places in Dubai

Calculate route by:

Setting a new start time and end time for the places while calculating the route

[✓ Using the existing start and end time you specified for places]

Choose a date to display its places: 03/12/2013

Select the places you want to include in calculating the route:

[✓] Asiana Hotel Dubai

[✓] Shogun

[✓] China Sea Restaurant

[✓] Al Ibrahimi Palace Restaurant

Travel mode: DRIVING

calculate

Fig. 4. Interface for calculating the route among several user-specified places.

from the database. The same web service was also exploited to asynchronously update the city plans based on the itinerary.

When using the Expedia API for retrieving hotel information, we found that the number of allowed requests was drained rapidly if all the hotel images were retrieved. To resolve this issue, we generated pages where each page could display only five hotel images instead of retrieving all the images and consuming the requests allowed. We also submitted our website to be reviewed by Expedia requesting them to increase the number of requests allowed.

IV. Conclusion and Future Work

This paper describes the Smart Travel Planner, a web-based intelligent system which assists travelers in planning for their trips by providing them with a single application with a unified interface for accessing the overwhelming amount of travel-related information scattered throughout the internet and also enables them to create itineraries, calculate routes among several places, create profiles, set preferences, and block personal time slots. This system has been implemented as a mashup that retrieves and integrates information from several travel-related APIs, such as Google Maps API [9], Google Places API [10], Expedia API [2] and Wikipedia API [11].

We plan to extend this system in several ways in the future: We intend to add additional functionality to the system by providing a hotel room reservation service, car rental options, transportation ticket purchase service (for train, bus, or plane), and email/mobile phone alerts about flight times and meetings, etc. We also plan to improve the "Calculate Route" function by taking current traffic and weather conditions into account when determining the route. Similar to [6], we are contemplating utilizing semantic web technologies to return results better customized to the user preferences. We also aim to port this application to a mobile platform. In this case, GPS information from the mobile device may be used to determine the user's position and the user may be provided with travel-related information relevant to his current context. Furthermore, we aspire to launch an Arabic version of the website and include a more specific database about Saudi Arabia. Since the performance of any application can be improved by storing static data in a local database avoiding the need for constantly refreshing this information, we will thoroughly examine our system to determine which data can be locally stored and also explore what, if any, other optimizations can be applied to enhance the system's performance.

Acknowledgment

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for partially funding the work through the research group project number RGP-VPP-157.

References

[1] S. Govardhan and G. Feuerlicht, "Itinerary Planner: A Mashup Case Study," in *Service-Oriented Computing - ICSOC 2007 Workshops*. vol. 4907, E. Nitto and M. Ripeanu, Eds., ed: Springer Berlin Heidelberg, 2009, pp. 3-14.

[2] "Expedia [online]. Available: http://www.expedia.com."

[3] "Yahoo! Travel [Online]. Available: http://travel.yahoo.com/."

[4] "TripIt [online]. Available: https://www.tripit.com."

[5] "TravelTracker [online]. Available: https://itunes.apple.com/us/app/traveltracker-personal-travel/id284918921?mt=8."

[6] C. Tuan-Dung, P. Thanh-Hien, and N. Anh-Duc, "An Ontology Based Approach to Data Representation and Information Search in Smart Tourist Guide System," in *Knowledge and Systems Engineering (KSE), 2011 Third International Conference on*, 2011, pp. 171-175.

[7] W. Liu and L. Bao, "Application and Exploration of Travel-Service and Information System Based on Web Service," in *Computational Intelligence and Design, 2009. ISCID '09. Second International Symposium on*, 2009, pp. 438-441.

[8] S. Navabpour, L. S. Ghoraie, A. A. Malayeri, C. Jingxi, and L. Jianguo, "An Intelligent Traveling Service Based on SOA," in *Services - Part I, 2008. IEEE Congress on*, 2008, pp. 191-198.

[9] "Google Maps API [online]. Available: https://developers.google.com/maps/documentation/imageapis/?hl=ar-SA."

[10] "Google Places API [online]. Available: https://developers.google.com/places/documentation/?hl=ar-SA."

[11] "Wikipedia [online]. Available: http://www.wikipedia.org/."

[12] "Microsoft Visual Studio 2010 [online].Available: http://www.microsoft.com/visualstudio/eng/products/visual-studio-2010-express."

[13] "Microsoft SQL Server 2008 [online]. Available: http://www.microsoft.com/sqlserver/en/us/default.aspx."

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik,* vol. 1, pp. 269–271, 1959.