

Mobile Services
Project Report

Where2Eat

Taslim Arif
Francesca Guzzi

Free University of Bolzano
June 9, 2009

Table of Contents

1. Project Description and motivation.....	3
2. System functionalities.....	4
2.1 Rendezvous management.....	4
2.2 Restaurant search.....	4
2.3 Negotiation.....	4
2.4 Negotiation history.....	4
2.5 Friend Management.....	4
3. Human Computer Interaction.....	5
3.1 Initiate a rendezvous.....	5
3.2 Searching Restaurants.....	5
3.3 Sending the first proposal.....	6
3.4 Accepting or sending counter proposal.....	7
3.5 Load old rendezvous.....	7
4. Code Structure.....	8
5. Technical problems and solutions.....	9

1. Project Description and motivation

Where2Eat is a mobile application designed for people who want to search together a restaurant where they can go to eat. The application provides the users the possibility to search for a restaurant by features, send proposals to each other and negotiate until a common decision is taken.

The application has been designed taking into account the possible scenarios of two people choosing a restaurant. To come to an agreement two users need:

- access to the list of available restaurants
- a filtering system like a search engine
- the possibility to speak/interact with the other person for negotiating.

In a daily scenario, such tasks would require for instance to be on the phone with the other person, while one of the user browses in the internet.

The main goal of the Where2eat project is to provide an application that handles restaurant searching and negotiation at the same time, also giving support for converging to a decision.

2. System functionalities

2.1 Rendezvous management

The user can start a new rendezvous by choosing a friend and the purpose of the meeting. The friend receives the rendezvous request and can either accept or reject the invitation. In case of acceptance the negotiation can start.

2.2 Restaurant search

The user can search for a restaurant based on the following features:

- Price
- Location
- Category

The user can also browse detailed information about the restaurants, such as image and description.

2.3 Negotiation

The user can send and receive restaurant proposals. When a restaurant is received the user can decide to accept the proposal or search for a new restaurant to make a counter offer. Before sending the counter offer, the user can see how much the categories of the new restaurant differ from the last restaurant received.

Any time the user has the overview of the status of the negotiation, because the application always shows the list of received and proposed restaurants.

2.4 Negotiation history

All rendezvous are stored on the mobile phone persistently. The user can decide to load a rendezvous any time he wants. The user can also load past rendezvous that have already come to an agreement and rejected rendezvous

2.5 Friend Management

The list of contacts is stored persistently on the mobile device (name, surname and number). The user can see the list of contacts, edit and add a new contact.

When the user creates a new rendezvous, the contact list is provided to choose the friend to invite.

3. Human Computer Interaction

A typical interaction between two users consists of the following steps:

3.1 Initiate a rendezvous

When starting the application the user has three options:

- 1) New Rendezvous: start a new negotiation
- 2) Old Rendezvous: load an old negotiation
- 3) Contacts: view and edit the list of contacts

By selecting “New RendezVous” the user can create a new negotiation and is asked to specify the friend to invite and the purpose of the meeting. There is also the possibility to change the contact number if it does not correspond the number in the contact list.

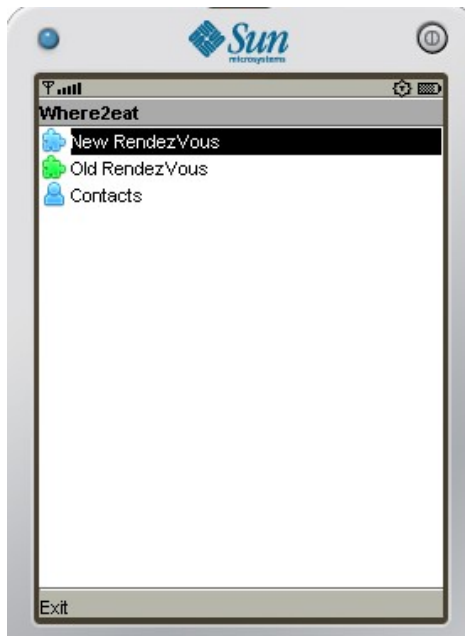


Figure 1: Starting screen

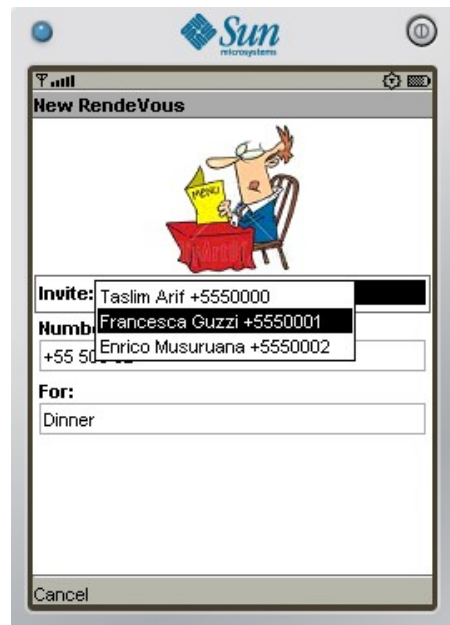


Figure 2: New Rendezvous

3.2 Searching Restaurants

When a rendezvous is created, the user can search for the first restaurant to propose by categories. The system will find the restaurants that satisfy the user's query and show them in a list. From the results list it is possible to browse the details of each restaurant, that contain the categories, image and description.

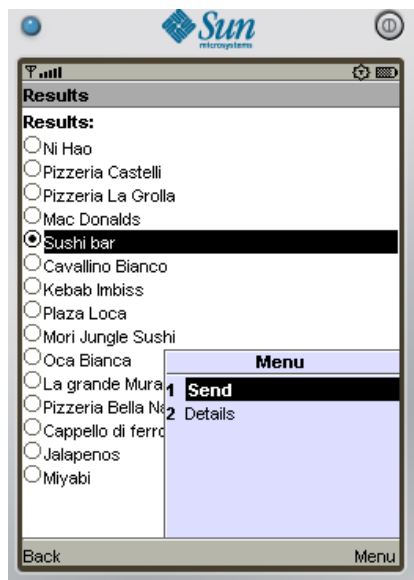


Figure 3: Categories selection Figure 4: Results of the search Figure 5: Restaurant details

3.3 Sending the first proposal

After choosing a restaurant the user can send the negotiation request with his first proposal. The receiver is asked whether to start or not the negotiation. In case of rejection, the negotiation stops. In case of acceptance the negotiation can start and both users can now send proposals to each other.



Figure 6: Sending proposal

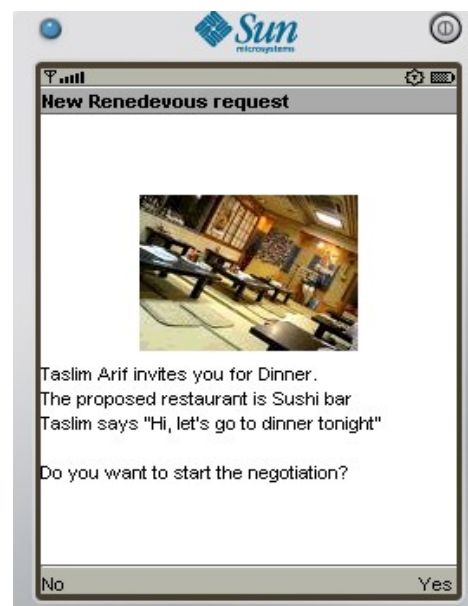


Figure 7: Negotiation request

3.4 Accepting or sending counter proposal

While a negotiation is going on the user has the following options:

- accept a received restaurant
- insist on an already made proposal by sending the same restaurant again
- search for a new restaurant to propose

The restaurants received and proposed appear in temporal descending order and the user can look up the details of those restaurants any time.

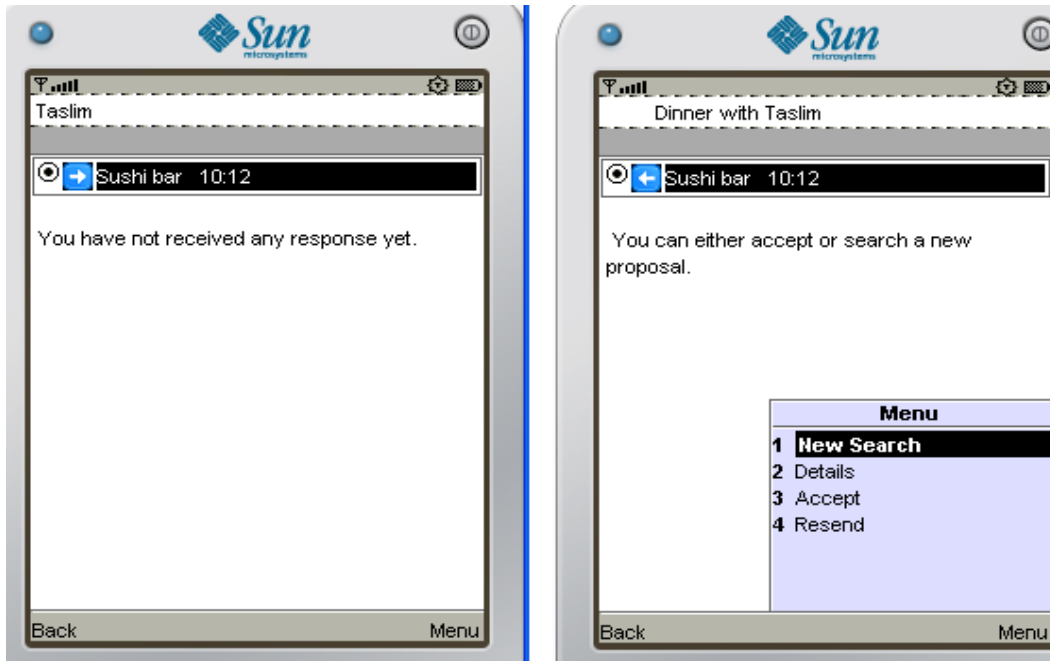


Figure 8: Ongoing negotiation

3.5 Load old rendezvous

The user can load past rendezvous and restart incomplete negotiations.

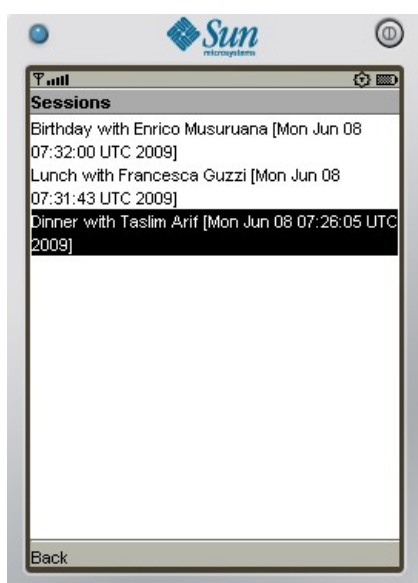


Figure 9: Load rendezvous



Figure 10: Proposed and received restaurants

4. Code Structure

The application is composed of three main parts:

Midlet

The MIDlet is the core of the application, it contains all the GUI objects and it is responsible for mapping the user inputs to the desired actions. It also handles the communication between the peers, by creating the messages to send and processing the incoming messages.

Data classes:

The application has three main entities:

- Friends: the contacts that the user can invite to a rendezvous.
- Restaurants: the restaurants among which the user can choose with the detailed information that the user can look up.
- Sessions: a session contains a rendezvous, the list of received and proposed restaurants, the users involved, time and the current status of the negotiation.

For each entity a class to hold all the information has been defined and an additional Manager class to manage the list of objects and the persistent storage.

Utility classes:

The utility classes provide methods to send and parse the sms.

The following table gives an overview of all the classes:

Name	Description
Friend	Holds information about contact
FriendManager	Stores and retrieves friends list and provides related utility functions
Restaurant	Holds information about restaurant
RestaurantItem	Holds information about each proposal
RestaurantManager	Provides searching functions for restaurants and other utilities to manage restaurants list
Session	Holds information about session
SessionManager	Manages list of sessions including loading and storing into persistence storage
DetailsForm	Restaurant details form
SMS	Holds SMS message components
Sender	Responsible for sending SMS
SmsParser	Parses SMS message to form SMS object
Where2eatMIDlet	MIDlet class responsible for coordination of all forms and their interactions, command actions.

5. Technical problems and solutions

Intuitive user interface

The main challenge in developing the application was the design of a simple and intuitive user interface, that could show to the user all the possible (and permitted) actions and the current status of the negotiation: sent and received proposals, pending requests, accepted and rejected requests.

This issue was solved with the use of dynamically changing colours and icons, that enable the user to understand intuitively the status of the considered restaurants. Arrows have been used to represent incoming and outgoing proposals. Blue, red and green have been used to represent respectively pending, rejected and accepted proposals.

Dynamic messages are provided to understand the negotiation status correctly and the possible next steps.

Converging mechanism

Another important issue was to find a way to actually help the users to come to a decision of a restaurant that could possibly satisfy both.

The definition of categories allowed to characterise and compare the restaurants. The comparison of the new offer with the last offer received provided before sending any counter proposal, helps the user to understand how much the two restaurants differ.

Status management

A complex issue was to handle a lot of scenarios with many attributes and reacting properly in each possible situation.

To solve this problem all the possible interactions between the users were analysed, and a finite state automaton was developed. In this way the set of actions to be taken for each state transition could be defined, and therefore every scenario and user interaction was properly handled.

Communication protocol

The last problem regarded the efficient exchange of messages between the users, that act both as sender and receiver.

A protocol was established to manage the communication, which happens via SMS.

The following tables shows the five possible message types with the content of each message.

	Message content	Message type
0	own session id, restaurant id, purpose	Negotiation request
1	other 's session id, own session id	Accepted negotiation
2	other's session id	Rejected negotiation
3	other's session id, restaurant id	New proposal
4	other's session id, restaurant id	Accepted proposal