1. PROBLEM

Our goal for this project is to provide an electronic tool for budget travelers, aimed at improving their travel experience. Our contextual inquiry at the start of our design process gave us an insight into existing practices of budget travelers, the tools they use, and how they use those tools. We interviewed several travelers at a youth hostel in San Francisco, and from these interviews we began to design an interface for a tool we called the ‘Electronic Travel Assistant’ (ETA). From our contextual inquiry we learned that the primary tool used by budget travelers is the tour book, such as a Lonely Planet guidebook, or a Let’s Go guidebook. These books are typically several hundred pages long, are updated once a year, and contain lists and short descriptions of points of interests such as landmarks, restaurants, and entertainment venues. They also contained some limited maps of the cities they cover, as well as cultural and historical summaries of covered regions. Travelers told us that the most important components of the guidebooks they used were the maps, however limited they were.

In addition, because of the importance of finding a place to stay, the accommodations listings were also a very valuable resource. Furthermore, travelers also enjoyed the cultural segments in guidebooks. There were a lot of complaints, however, that guidebooks were difficult to search, that the listings and reviews in guidebooks seemed to reflect the interests of the authors, and did not usually seem to match travelers interests, and that much of the information was outdated. Guidebooks, therefore, are the most common tool for our target group, and the maps and accommodations listings provided an invaluable resource for someone in a new city. However, they were inadequate in many regards. The goal of ETA, therefore, is to provide an interface that will solve these problems, that will provide efficient access to large amounts of information, personalized information and ideas that reflect the interests and goals of the user, a means to collect information about users interests, cultural information, in-depth map coverage, as well as other useful resources.

2. SOLUTION OVERVIEW

ETA bridges the gap left by conventional resources by managing travel information for the user. And, it does so in two ways. First, ETA gives users quick access to up-to-date travel information, such as accommodations, dining, and entertainment, through synchronization with the Internet. For example, once the user selects the cities that he is interested in, he can download all the germane information concerning the locations onto ETA and can have it at hand when he visits the destinations. Secondly, it automatically creates travel itineraries based on the user’s preferences and interests, and ETA dynamically adjusts them to bring the user within budget. For example, when the user profile is created, ETA will use the information provided to generate interesting and financially compatible list of places to stay and where to eat.

3. TASKS

We wanted our tasks to be representative of the program features and also be able to string together a good mental model of project for the users. Our contextual inquiry also informed us that the users emphasize that a feature such as mapping should be easy to use and accessible, as a result, we made
Task 1 (Hard): The first task is to create the user’s profile in which they entered in their preferences and interests. This task is considered “hard” because it requires the most steps to complete and leaves room for many errors.

Task 2 (medium): The second task is to find an accommodation in a specific city. This task is labeled “medium” since it requires less steps than the first but is more of a cognitive process than the next.

Task 3 (easy): The last task is to locate a particular destination on a map. This one is considered “easy” since it requires the fewest steps and hardly involves any thinking effort by the user.

4. DESIGN EVOLUTION

On deciding to develop a travel assistant application for a handheld device, we wanted to determine who our target group of users would be and what kind of features they would want on the program. To be more specific, we decided to develop the software for budget travelers such as college students. In order to find out what these budget travelers need, we visited youth hostel in San Francisco (Hostelling International youth hostel) to inquire them about how their visits to the new city could be improved. We interviewed three different travelers from various countries in order to get wider range of feedback. Each of the three interviewees gave us valuable feedbacks on what would make their budget travel experiences better. For example, one of the interviewees told us that emergency contact information would be a great help in case of accident. In addition, the interviewees thought that it would be wonderful to get more personalized and fresh information and content, which tour books often fail to provide.

Considering many feedback from the travelers, we put down our initial design using the PowerPoint (unfortunately) as shown in Figure 1.

Before long, we realized that HP Jornada had a very limited screen size compared to conventional computer screens; we wouldn’t be able to fit all the details on a single screen as shown in Figure 1. Thus, before doing our Low Fidelity testing, we trimmed down a lot of details like the icons and the menu layouts so that we will be able to fit our content in a single screen without scrolling. We decided that we need to get rid of the ‘system menu’ on the top that took up a lot of valuable vertical space.

In order to test our prototype without using the interactive software, we prepared low-fi sketches on
approximately 20 5" x 8.5" pieces of ordinary white paper and drew and cut out the parts we designed on our prototype such as buttons, pop-up, as well as all screens (one of the screens shown in Figure 2). Instead of getting rid of the ‘system menus’ from top, we put two lines of menu bars. The upper part of the menu bar was meant to be a global menu; it was always there as the users browse through the pages in ETA. The lower part of the menu bar changed for each screen to menus that were relevant to that particular screen.

We observed three testers as they interacted with our paper prototype shown in Figure 2. Through this test, we were able to identify a lot of different problems that we needed to improve on. For example, all users were confused by our two layered menu system as described previously. As an example, the testers weren’t sure what kind of menus were under Tools, Options, and ToDo’s because wordings weren’t very clear according to one of the testers. Specifically, in order to find an accommodation, users need to click on ToDo menu and find the menu, “Find Accommodation”. However, none of the testers were able to recognize that. Some users felt that it was very restrictive how the program starts. The program starts by asking users to Create Profile; before creating a profile, users can’t perform any functions on the software. One of the testers was confused why they had to create a profile, and the others just didn’t want to create a profile in order to search for accommodation and perform other tasks. Moreover, when creating their profiles, users weren’t sure what the questions were asking and why. For instance, it asks to “Enter (destination) Address” and a lot of users entered their home addresses. Another question asked users to pick their favorite music types and we wanted users to be able to make multiple selections. However, all of the users only picked one of the music types, failing to recognize that they could have made multiple selections.

We got a lot of improvement suggestions in this phase and we all thought that this was the most important user test and most significant interface change. We were able to perform some major interface changes such as menu system, and program’s better starting point because we only had the paper prototype. Our low-fi design was converted into a hi-fi prototype shown in Figure 3. Most importantly, the new interface improved on the starting point of the program. Instead of forcing every user to create a profile, we gave them the options of performing other functions by clicking on one of the intuitive icons.

We thought that by using the icons, it would make it easy even for novice users to recognize what the buttons do. We also simplified the two layered menu system again to a single line menu bar that contains just three menus. Everything that was previously under Tools was moved under Tools. We also added the convenient home button because we noticed that users often asked how to go back home when they got confused. So, we thought that it would help the users to put a home icon on every page. We added a lot of other minor changes as well in addition to these major changes. For example, instead of having users select their favorite music types or dietary restrictions from the menu with drop down look, we changed them into checkboxes to let users know that they could make more than one selection.
Our interactive Hi-Fi prototype shown above was tested again with other group of testers, and it seemed like they were generally able to recognize icons and menus to find their ways around the program. However, there were still testers clicking on wrong buttons. For example, a tester who was not familiar with HP Jornada or Windows CE accidentally exited the ETA program, which could have been prevented if the tester had been warned before completely exiting the program. In addition, we heard major complaints about system keyboard covering the text fields when the keyboard pops up; once the system keyboard pops up, it covers half of the screen including the text fields. In addition, we have changed the favorite music types menu to checkboxes as mentioned above to improve the intuitiveness of multiple selectable checkboxes. However, users still failed to recognize that they could make more than one selection. Moreover, the users wanted more help system about each particular function such as what he could do if he clicked on the Profile icon.

In order to fix the problems we found through the interactive prototype and various heuristic evaluations, we listed the tasks and sorted them by severity rating. Then, we implemented most of the important fixes such as system keyboard problems and better help system. The keyboard problem was easily fixed by putting the textboxes on the upper part of the screen so that system keyboard won’t hide them when popped up. Also, in order to improve the help system for each function for novice users while still giving the expert users flexibility, we decided to introduce the splash screens as shown in Figure 4. With the splash screens, first time users can read what each function is and what they could do with the function. And, they can turn the splash screen off once they understand the button.
We also provided more confirmations before performing critical tasks such as exiting the program, and successfully creating user profile. Moreover, we provided better default values for some of the check boxes. For example, in the question, "Select your favorite music," we wanted the default choices to be all the music instead of none. This could achieve two things for the users. First, it would not restrict users even if they didn’t pay attention to the particular question and proceeded. Second, it would make it clear to other users they could make more than one selection.

Through the iterative tests followed by design improvements, we were able to refine our wild idea into an intuitive software system that lets even novice Windows CE/Palm users to set up their profiles in a breeze, easily find accommodation, and suggest personalized information on dining and entertainment as shown in the task description section.

5. SCENARIOS

Storyboard for Task 1: Creating a Profile
6. FINAL INTERFACE

Description of Final UI Design

The final UI design is centered on a homepage that contains links to all other features of ETA (Figure 1). This is the starting point for navigating throughout the UI. The menus in the menu bar at the top of the screen are used to provide supplemental navigation features that provide the users with an alternative means at moving around the interface. The ‘tools’ menu has links to the ‘lodging’, ‘entertainment’, ‘dining’, and ‘sights’ sections. The ‘tools’ menu also contains a submenu ‘choose city’ that enables the user to toggle between cities that he or she wants to work with (Figure 2). The ‘options’ menu provides links to the ‘search’ and ‘profile’ features, and the ‘help’ menu provides links to help features. Because of the importance of the homepage as the central point of the interface, all other pages in the interface contain links back to the homepage. ‘Forward’ and ‘back’ arrow buttons are also located next to the ‘home’ links on every page to allow users to retrace their history of navigating throughout the interface. The following paragraphs describe functionality of the features we have implemented.
The ‘profile’ section is designed to be a wizard that takes the user through several questions aimed at gathering information about user interests and goals for their trip. Because of the confusing nature of the profile section, a splash screen containing information on what the profile is used for and how to revise any settings later is the first screen of the profile (Figure 3). The user has the option to disable the screen from appearing again in the future. The user moves throughout the ‘profile’ section through ‘back’ and ‘next’ buttons located at the bottom each screen. Finally, and the end of the ‘profile’ section, the user is prompted to save and return to the homepage. Although the ‘profile’ section is an invaluable means of collecting information from the user, it is not required that the user complete it in order to use any of the other features. Input from the profile is intended to affect the ‘suggested’ listings in the ‘dining’, ‘lodging’, ‘sights’, and ‘entertainment’ sections. If the user does not change his or her profile, the suggestions will be identical to all listings, because the profile contains default values that are chosen to not filter out any points of interest. As the user adjusts the profile, listings that do not meet the interests of the user would not be suggested.

Figures 1, 2, & 3, respectively.

The ‘dining’, ‘lodging’, ‘sights’, and ‘entertainment’ sections follow the same template, and are therefore used in the same way (Figure 4). They start with general information for each of the listings, and links for each listing to a page with more details and summaries. The page of general listings also has an option at the top of the page to toggle between recommended listings and all listings for the particular section. When a user clicks on the ‘details’ button for a listing, he or she is presented with more information about the particular point of interest. The ‘details’ page also has links to a map of the listing and to directions to the listing.

The ‘maps’ section contains the mapping resources of the interface. The main page of the ‘maps’ section is divided into two listings of links (Figure 5). The first two links are to the mapping features ‘find location’ and ‘directions’. The user has an option here to get a map or directions to a location that the user must enter. The ‘quick links’ provide a links to maps of important neighborhoods or areas of a city. These links would change with the city. If the user simply wants to browse through the neighborhoods of a city, he or she can simply click on any of the ‘quick links’. If the user needs to find a map of an address, he or she can click on the ‘find location’ link. The next screen will prompt the user for an address, and is designed in the same way that the Yahoo.com maps section is designed. Once the user enters an address, a map for that address is shown on the ‘results’ screen. The ‘results’ screen is displays a map of a given address, with buttons to zoom, or move around that map (Figure 6). The ‘results’ page is used by several sections of the interface, for the sake of consistency of design. It is used when users click on ‘show map’ from the ‘details’ page of a listing, from the ‘quick links’, on the ‘maps’ main page, and as the results page when a users decides to use ‘find location’.
Intentionally Unimplemented Features

Some design elements of ETA were left unimplemented because we felt that the functional sections provided enough feel for users to test out the important features of the system. Elements like dining, entertainment, sights, and search (choose city) were unimplemented. The backends for lodging and maps have been implemented to good degrees, and it would be beyond the scope of this course to completely do so. The backward and forward buttons on the upper right hand corner of the screen were meant for future versions of the program in which they would help users navigate through large number of screens. And, we felt that the user interface was intuitive enough to do without a multi-layered help system. Although, we left that option open for future versions.

Benefits from Using the Development Tools

Electronic Travel Assistant for Hi-Fi prototype was written in Java to be run on Chai, the Java Virtual Machine on HP Jornada. We worked with WebGain’s Visual Café 4.5 Standard Edition to generate Java Graphical User Interface code by a simple drag-and-drop. It allowed us for a faster development by minimizing the need for handwritten code. For example, using WYSWYG feature of Visual Café, we were able to generate many lines of Java code by drag-and-dropping various AWT features such as menu bar, radio buttons and check boxes onto the Frame. In addition, Visual Café provided us with a project manager, which for example listed all Java files that we were editing to allow us a better navigation capability among the source files in case we needed to refer to other files. In order to check for the incompatibility of Chai with Windows Virtual Machine, we used a software tool provided by HP that notified us for possible incompatibilities. Then, we compiled the Java code in Visual Café IDE to transfer the binary class files onto Jornada using ActiveSync. ActiveSync interface also gave us a complete control over our memory usage in Jornada.

Limitations of Using the Tools

Although the Visual Café IDE allowed us to develop the AWT user interface relatively quick, it did not have the "Chai compatible" AWT mode. In other words, we did not know whether the generated code would be compatible with Chai Virtual Machine until we ran the compatibility checking software mentioned in previous paragraph. Once we spotted out any incompatibilities using the tool, we needed to go back and change or remove the code. On a same note, Visual Café did not provide us with the actual view of how it would look running on Chai because it was not tightly integrated with the IDE. In addition, it was sometimes impossible for us to tweak minor details of user interface using the WYSWYG capability of Visual Café. Thus, we needed to change the code manually in order to control the interface in details. It was a difficult task due to the sheer size of code it generated. Moreover, it did not generate any comments or sensible variable names.