HCI Project 3: HEURISTIC EVALUATION

By Brian Nguyen, Woo-Jae Lee, and Cody Fulford

University of Calgary

CPSC 481 - HCI

**~~To do:~~**

~~1. Explain how the prototype was built~~

~~2. Explain the major components of the system~~

**~~Task 1 and 2 takes up two pages.~~**

3. Illustrate interface components by using the tasks you designed in P1 as your guide.

4. Include small screenshots of the interface where appropriate, where there is variance from these tasks point them out and justify design changes

**Task 3 and 4 takes up four to six pages.**

5. Perform a heuristic analysis on your application.

a. You should identify a small set of major problems that must be mixed and summarize this.

b. Describe the issues that were identified along with the heuristics that were violated.

c. Describe how you would address them in another iteration.

d. Sort these in terms of severity, in terms of functional/conceptual area (in the way that makes sense in respect to your system.

d. Justify your choice for presentation and make it clear that sort is being done in the written presentation.

**Task 5 takes up a maximum of six pages.**

6. Include an appendix. Include as an appendix the raw notes from your heuristic evaluation in particular, the notes from your individual heuristic evaluations

**Task 6 is easy**

**Prototype building**

Our group utilized the agile software development method where our prototype will be a product of continuous and iterative improvement. Since our prototype was developed under a small time frame, we had to choose one that allowed for a sufficiently short development process. Agile was chosen as it fits our needs as a small group with a short time frame within to work. It also allows for iterative requirement changes depending on our classmate’s feedback, clients and supervising ta.

For our vertical prototype deliverable, one of the essential features that we decided to include was a function that displays certain information about a ski resort. In our research phase, in P1, we conducted a survey which enumerated the common methods that people use to learn about ski results which we found to be word of mouth (~73%) and through the resort website (~65%). Henceforth, another major component of the system was the ability to allow users to give ratings for resorts and leave comments. Finally, the most important question that we asked heavily influenced the rest of our major components. That question was: What sort of information do you look for most often when searching for ski resorts?

Our survey revealed that people are mostly interested in seeing what routes are available at a ski resort, and the conditions at that particular ski resort. The main finding was that users are particularly interested in seeing the difficultly of the slopes. Hence the display information component of the application allows the user to 1) See ratings 2) See snow conditions. Finally, we created a function that allowed users to interactively plan their routes, leave reviews and track the time they needed to complete their run. With this information in mind, we created affinity diagram to compare ideas from different perspectives and agreed on that notion that we needed to focus on creating a minimalist design; a low clutter interface with big buttons. This is because a typical user on the ski hill would be in a fast paced environment therefore a clean design where every button is easy to press is ideal for the user in question. For users off the hill, they would also appreciate this.

The major components that we came up with and the design principles that we wanted to emphasize was based on our survey and affinity diagramming. We recognized that our users would most likely be casual skiers and snowboarders, so rather than building an application that could specialize in one thing, (hotel/lodge booking, restaurant reservation near ski resorts, transportation were some ideas that we thought of) we choose to build components that would aid the majority of our target pool to serve as a basis for a prototype.

Our first mockup that was presented to the class involved four ski resorts that we were interested in. These ski resorts were local to Alberta and we wanted to create these prototype functions (display resort data, route planning, reviews) for all of these four resorts. We realized our application would have modular components meaning that the data that exists in functions for one ski resort will not affect the functions that exist for another ski resort. The children classes of a particular resort will not communicate or exchange data with other children classes of different resorts. Knowing this, we decided to shrink our scope and just focus on building modular components for one ski resort namely Lake Louise. These modular components were built vertically up and could be used as a basis for components of other ski resorts.

**Major Components**

1. Information about a ski resort

To start off simple, we created a mock information page for Lake Louise. We thought that the information on this page should have low clutter, be eye catching and adhere to our idea of a minimalist design. The three ‘must have’ information pieces are: the current weather at the resort (temperature, wind and precipitation), the snow conditions (snow quality, has the snow been groomed or not, is the snow powdery) and the amount of open lifts there are on the resort.

In a fully functional application, we would have made this component more interactive. We wanted to give the user the ability to pick a subset of data that they would like displayed instead of conforming to what we set up as default information. We thought that the information displayed was important to us, but might not be beneficial to our end users hence the idea of creating mini “widgets” that the user can move around on the page.

1. Make a review about the resort

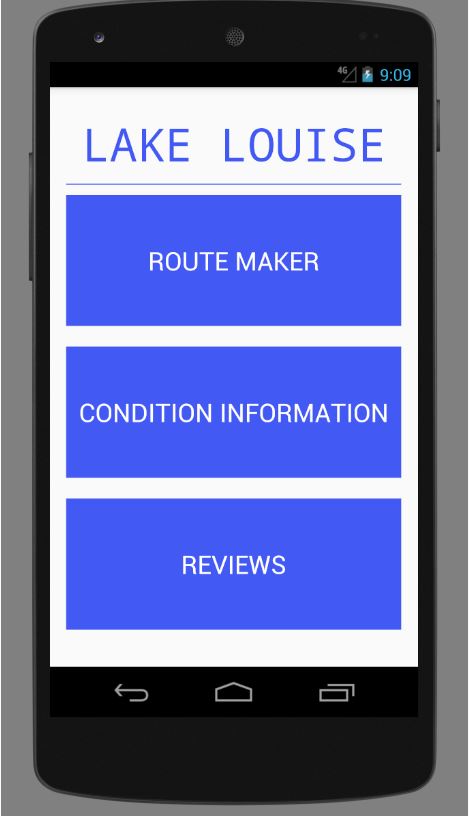
Simply put, the review page allows users to see what others have said about the resort and also gives users the ability to create their own review.

A fully realized review function would give users the ability to see the nearby restaurants and hotels among other things. They would be able to scroll through a selection of items to figure out where is a good place to go.

1. Route making

The route making component gives users the ability to see an overview of the resort, as well as planning routes that they would like to ski or snowboard on. The route making page allows users to see a list of possible slopes that they could go down. Once a slope is selected, various routes can be mix and matched. A submit button leads to a page where the user can start a timer to measure how long it takes them to complete the route. Finally, they would be able to share the routes that they selected on social media as well as making a review of the selected slope.

We wanted to go with the idea of a minimalistic design so that the user is able to navigate through the application with ease. In our affinity diagram session, all of us pointed out the need of having a low clutter interface. In the environment that we expect some of our users to be in, big buttons will make it easier for our users to navigate through the interface. Users might be wearing gloves so fat fingering a button is easy when buttons are small. This is a design choice we made in attempt to circumvent this problem.

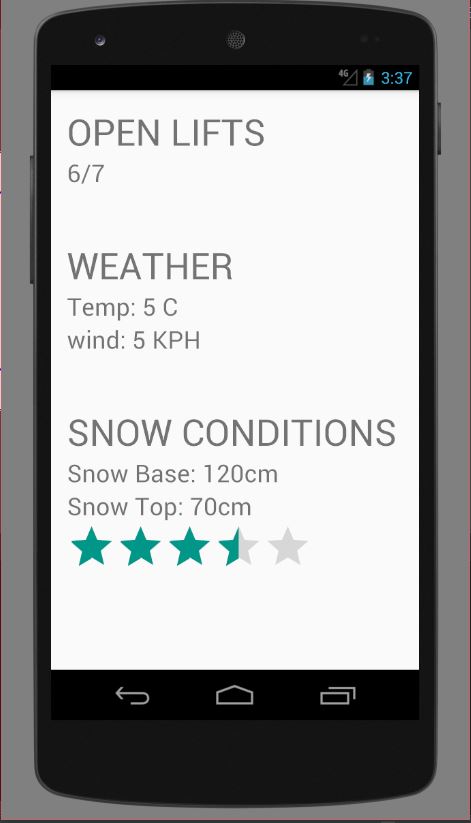


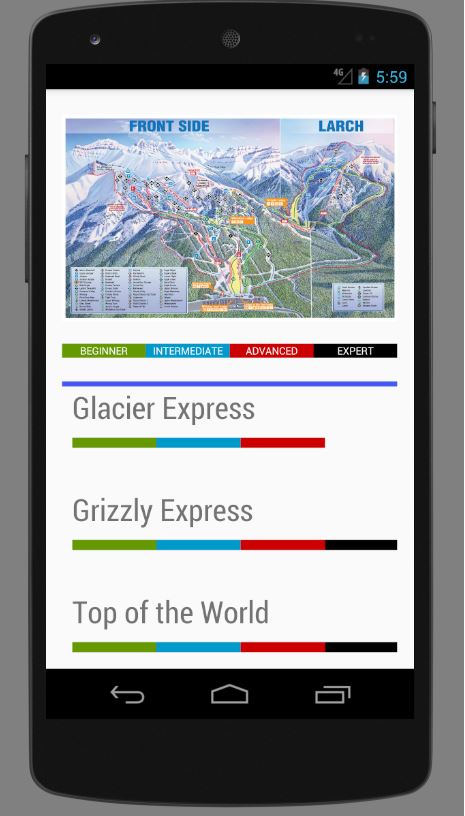
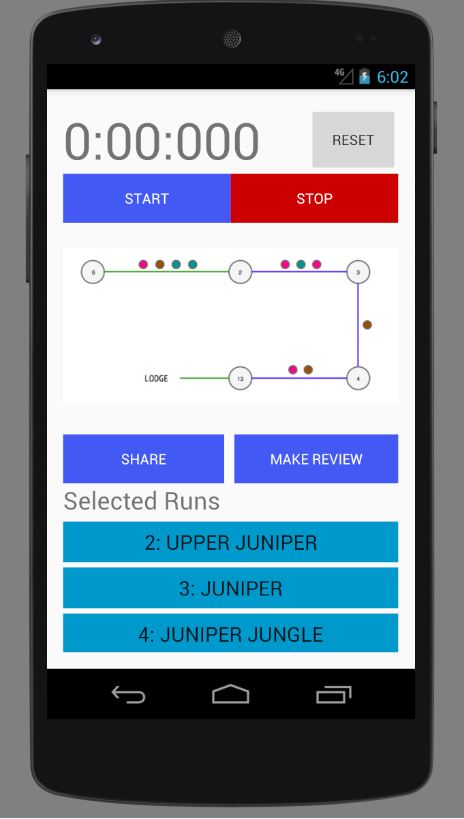
Terry (Task 7) who is a ski patrol member is an example of a person uses the application on the go at the ski resort. He is expected to be outside, constantly in motion and an interface that is minimalistic will allow him to navigate through the application with ease.

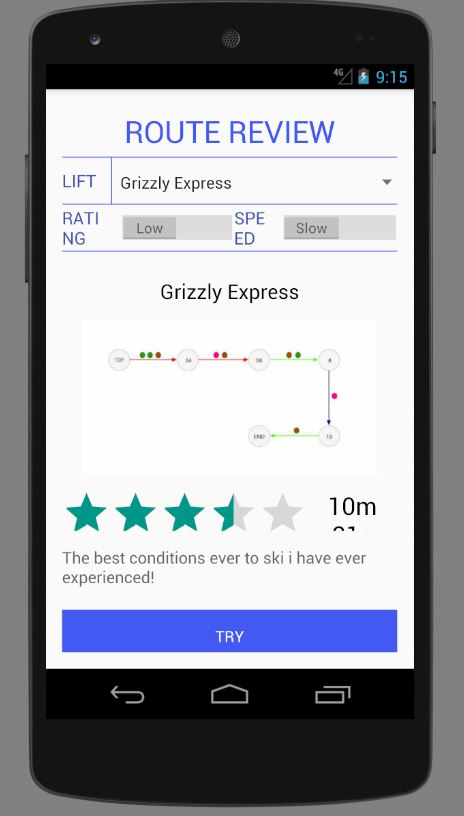
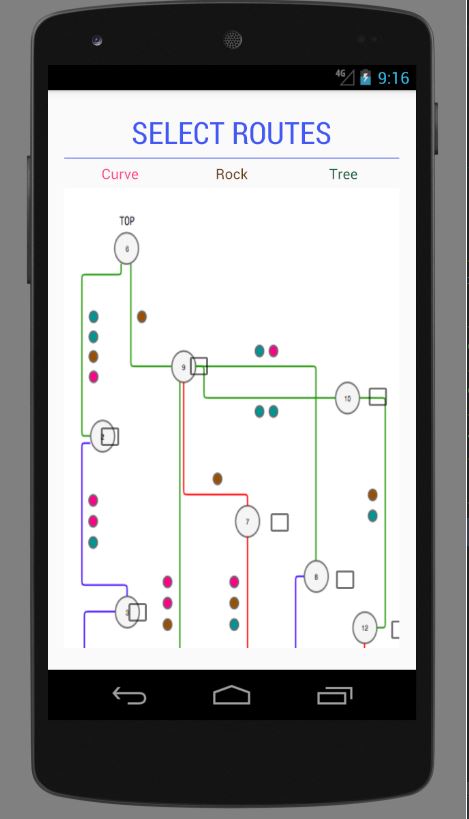
The three big buttons allude to what functions our application will have. We rely on the user’s intuition to make the right selection. Using usability testing with potential users, we can determine if our button text is too ambiguous. Hopefully the decisions that we made for our button text is good. We wanted to avoid having similar descriptions (for example snow rating and review) which may lead users into selecting the wrong function that they actually need. Nothing else in our overview besides these three huge buttons. We set up a visual constraint as to how the user can use the application. Simply put, we didn’t want a new user to get lost the moment they opened up the application.

At the beginning of our design phase, we pointed out a problem in some of our initial sketches where we had all this information being presented about a ski resort upon opening the application. Not only was it difficult for a newbie to learn how to navigate through the application, there was a lot of clutter with small buttons that would’ve made the application tremendously difficult to navigate through while wearing gloves outside a resort.

Tom(Task 1) who does not have experience with applications but wants to figure out what sort of skiing conditions make for a good day of skiing will take the visual cue of tapping on the INFO button to gain more insight. Unfortunately for Tom and others, we decided to work on only one resort hence we no longer have a list of resorts that come up when opening the application. The favorite resort page also did not make it to the prototype. In our description of Tom, he wants to find information about several ski resorts in order to compare and contrast each resort.

To be more specific, he wants to determine the recent snowfall and weather forecast of each location. Supposing that Lake Louise was on his next list of things to examine, he should be able to select the correct function.

In the Make Route function, there is an overview of the resort as well as a list of ski hills which have their difficulty listed on top of them. This allows users to be able to quickly navigate through the list of slopes and quickly determine if a slope is too easy or too hard for them. For a beginner skier like Raza(Task 6), this function will allow him and beginners alike to determine what slopes that he



Heuristic evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Suggested Fix | Severity Rating |
|  |  |  |  |

**Task 3 and 4 takes up four to six pages.**

5. Perform a heuristic analysis on your application.

a. You should identify a small set of major problems that must be mixed and summarize this.

b. Describe the issues that were identified along with the heuristics that were violated.

c. Describe how you would address them in another iteration.

d. Sort these in terms of severity, in terms of functional/conceptual area (in the way that makes sense in respect to your system.

d. Justify your choice for presentation and make it clear that sort is being done in the written presentation.

For reference only delete after

**Visibility of system status**

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

**Match between system and the real world**

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

**User control and freedom**

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

**Consistency and standards**

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow [platform conventions](https://www.nngroup.com/articles/do-interface-standards-stifle-design-creativity/).

**Error prevention**

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.  
(Read full [article on preventing user errors](https://www.nngroup.com/articles/slips/).)

**Recognition rather than recall**

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.  
(Read full article on [recognition vs. recall in UX](http://www.nngroup.com/articles/recognition-and-recall/).)

**Flexibility and efficiency of use**

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

**Aesthetic and minimalist design**

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

**Help users recognize, diagnose, and recover from errors**

[Error messages](https://www.nngroup.com/articles/error-message-guidelines/) should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

**Help and documentation**

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Failed Heuristics | Description | Solution | severity |
| Cody | Match between system and the real world | On the timer page, after the user has started the timer the stop button continues to read stop. It would be better if the button read pause as a stop button implies that when the timer is restarted, the time will reset. Since this is what the reset button is for, the meaning is unclear | Change the text to read Pause | 1 |
| Cody | Aesthetic and minimalist design | On the select route page, the minimalist design is that there are only small selectors for each route. This is a problem because it does not relay the information to the user about how the runs are spaced apart on the actual hill. If a user is wanting to go down one of these routes an, it is possible that this design choice increases the possibility that the user misses their turn. | enhance the detail I the design to better reflect the spacing of runs | 2 |
| Cody | flexibility and efficiency of use | on the select routes page, the size of the buttons that the users have to select the runs is very small. This makes it difficult for the user to select the run with hands that are cold or wearing large gloves. | increase the size of each of the buttons | 3 |
| Cody | error prevention | It is currently possible for the user to select some set of runs where at least one of the selected runs is not connected to any other of the selected runs. This means that it is possible for a run to be generated where a portion of that run is literally impossible to reach from another run | set some of the run buttons to have a radio button like functionality so that when one member of a set of buttons is pressed, any button not related to that set can no longer be pressed | 3 |
| Cody | recognition rather than recall | the timer page has a list of buttons, each stating the name and number of a different run. While this is nice if the user can remember these numbers and names, it might be better if the user could to see a map of the hill where the selected runs are highlighted. This would allow the user to judge distances adn guide other skiers to these runs | instead of having a list of runs, have a map with highlighted runs | 2 |
| Cody | help and documentation | the rating and speed toggles on the route review page do not make sense to me. What do each of the toggles at the top of the page do? Do they sort? Do they filter? The user does not know | make what these buttons do more clear by having some text description or by having the label of each button be more descriptive | 2 |