CPSC 481: Foundations of HCI, Fall 2022
Dr. Ehud Sharlin, Ashratuz Zavin Asha, Christopher Smith

Interaction Design Project Part I
Task Centered Design and Prototyping

Out: Tue. September 6th, 2022
Due: Mon. October 17th, 2022

Intermediate project deadlines:
Mandatory weekly tutorials attendance of your designated tutorial.
Mandatory weekly group meetings with your TA (during your designated tutorials time)
Your proposed preferences for your project (three preferences via D2L dropbox, see options in appendix 3) by Thu. September 15th at 9am.
First group project presentations, TCSD Phase one and two: Sep. 23-29th
Second group project presentations, TCSD Phase three and four: Oct. 3-7

Please submit your final portfolio of the 'Final Project-Part I' by Mon. Oct. 17th, 2022, 8AM, electronic submission (20% of your final course mark)

Included Handouts

- Assignment 1: Task Centered Design and Prototyping: general
- Assignment 1: FAQ
- Appendix 1: Getting to know users and their tasks
- Appendix 2: Developing and evaluating initial prototypes
- Appendix 3: Project options list
- Grading Sheet

Overview

This project is a hands-on exercise on task-centered system design (TCSD) and prototyping, which is the first step in an iterative User-Centered System Design. Fundamentally, this means that you begin your design by getting to know the intended users, their tasks, and the working context of their actions. Only then do you consider what the actual system design should look like, because you would base the design on real people, real tasks, and real needs. User-Centered System Design is not an industrialized process where some cookbook formula can be applied. Nor is it an intuitive process where a programmer can sit in their office and think they know what the user and their tasks are. Rather, it is a hands-on process that requires you to go out and identify actual users, talk to them about what tasks they are trying to do, and understand the entire context of their work. You then base your designs on this information. Because it's only normal for your initial designs to be crude and riddled with usability problems, you will have to identify potential problems by continually evaluating your design and by crafting new designs. This is called iterative design (the design-implementation-evaluation cycle).

<u>What you need to do:</u>

In this assignment, you will begin your iterative design of a particular system (see Appendix 3 for a list of some of the projects assigned to groups) using Task-Centered System Design methods and low fidelity prototyping techniques. Your project advisor (TA) will help you select a project based on the preferences submitted by your group, this would be the same project that you will implement later this term, during the second (and final) part of the project (if you feel strongly about a specific project idea please touch base with your project advisor directly; changing projects between the first and final project parts is not recommended and needs to be approved first by your TA and Ehud as it will require extra work to investigate the background for the new project).

The immediate purpose of part I is to give you experience in:

- Articulating good task descriptions.
- Using the task descriptions to decide upon system requirements.
- Brainstorming low fidelity prototypes based upon the task descriptions.
- Evaluating the various prototypes through a task-centered walk-through.

The outcome of the assignment on Task-Centered System Design is a design portfolio containing:

- A list describing the expected users of the system and their work contexts.
- A list of actual, representative tasks that people are expected to do.
- A prioritized list of system requirements.
- A low fidelity prototype.
- A task-centered walkthrough of the prototype.

Teams

You will be assigned to a team work with four other students, forming a group of 5 (+/- 1 depending on overall tutorial size). The idea of working with others is to get alternate design ideas, multiple ways of looking at things, and more breadth at eliciting and interpreting evaluations, and it is also a necessity due to the large number of students taking the course. Your group will be formed by your TAs who will notify you of your other group members during the first week of the term.
Note that if this was being done "for real", the best team would have people from diverse backgrounds, which will give the team different perspectives on the problem. For example, a real team could comprise a project manager, a marketing person, a programmer, a representative end user, and/or a help desk person who regularly deals with end users.

Team Evaluation

As part of working within your team, each week you'll complete a "most valuable player" (MVP) survey voting for one member of your team as most valuable team member. The goal is twofold: the MVP survey helps your instructor and TA understand who is contributing and it allows you to reflect on your relative contribution to the group. The MVP survey will be done via D2L where you can name a team member (who is not you) and the reason for nominating them. You can also decline to name a team member for a week but you must justify why. This is part of your project assignment and will be mandatory. Incomplete surveys will result in a failing grade for the assignment – you are encouraged to remind your teammates to complete your MVP surveys each week but remember that the surveys themselves are meant to be confidential, please do not disclose your vote to your teammates.

Deliverables (Due when you hand in your portfolio for Part 1)

Your team will deliver a PDF portfolio containing their system design and discussion, written to the imaginary Vice President of Usability of your company.

Report format:
Your report should ideally be no longer than 20 pages in length, in 1.5 line spacing, with Times New Roman fonts, size 12. Note that when counting the pages you **should not count** the title page, table of contents, all the appendices, and Section 2/Phase 3: "Low Fidelity Prototypes".

Your team portfolio will include the following sections:

Section 1: Identification and requirements

Setting the stage for your project ("phase 0", which is required in addition to the four phases of the task-centered process):

- **Background environment**, including the current work situation and why a computer system is coming in
- **What the system will be used for as well as the general expectations that the system should satisfy.**
- **System constraints** that limit the design e.g., budget, equipment, operating system, legacy systems, etc.

1. **Phase 1 of the task-centered process: Identification**
    a. **Expected types of users** of the system, including their experience, expected training, etc.,

   b. **Work contexts** that describe the work setting and typical situations of the users.
   c. **Concrete task examples**. You will list at least 5-7 concrete task examples that have the properties listed in Appendix 1. Try to keep task descriptions short and to the point. Each task should be accompanied by a paragraph that describes the class of the expected user (e.g., a typical customer), the relative importance of the task (e.g., frequently done and important, infrequently done but still important, rare and not important, etc), and whatever other nuances you feel should be included. *Be sure to include a paragraph or two that describes how the tasks were collected and validated. If you took extraordinary measures to acquire and refine this information - it may make a difference to your assignment grade so don't forget to mention anything that may be relevant!*
2. **Phase 2 of the task-centered process: Tentative list of requirements**.
   a. From the task examples, extract the <u>major</u> system requirements and prioritize them into a) absolutely must include; b) should include; c) could include; and d) exclude. *Each category should be accompanied by a discussion as to why items were placed in that category*.

<u>Your team will be asked to present one or two sample tasks and requirements during the tutorial as well as to provide some additional information in order to set the stage for your project</u>, all team members are required to participate in the team presentation during the tutorial.

Section 2: The first prototype and walkthrough *(an annotated design + several pages for each walk-through evaluation)*

3. **Phase 3 of the task-centered process: Prototyping** (storyboard or Pictive)**.** Develop several low-fidelity prototypes of designs that you believe will satisfy the major requirements.
   Your team will include the prototypes in your portfolio. You should not only include your final design but also show the early ideas that you generated but may have discarded. This will help to illustrate to your marker how you went through an iterative design process as you created the prototypes (remember the cyclical nature of the design-implementation-evaluation cycle).
4. **Phase 4 of the task-centered process: team discussions and walkthrough.** Discuss the prototypes ideally with your teammate and with potential users and user representatives (but possibly also with friends, family or colleagues). You should be concerned here with how the general interface representation fits the users' view of their tasks. For the prototype designs that seem promising, convert the tasks from Section 1 to scenarios and <u>perform a walkthrough evaluation</u> of your prototype. In your portfolio, list in point form the problems and successes for each task. In essay form, summarize the major design problems that must be corrected, as well as what seems to work well for the system as a whole.

You will be asked to present your prototype and discuss the walkthrough results to your TA. Again the choice of low-fidelity (lo-fi) prototyping technique that you use is up to you (e.g., Pictives with sticky notes, Pictive with transparencies, a card board storyboard, etc.), just make sure that the prototype is complete and clear/neat enough to make sure your TA and other viewers of your presentation can understand how your interface is supposed to work.

A note on the Portfolio Format: The portfolio is intended to document the progression of your design, which includes your final project. Your project portfolio will be submitted as a PDF file and needs to be well organized. Please list your team members' names, email addresses and the title of your project on the first page. The second page should be a table of contents followed by the grading sheet (see p.14).

A note on the grading. Grading will be based upon the sophistication and maturity of the work, the logic of the written and oral presentations during the tutorials, and the completeness of the work. Although it would be nice to have truly elegant designs at this stage we are focusing more on how well you completely and objectively evaluate your designs. More than likely you will make major redesign changes when you return to your project for the final part (part II) of the project, later this term.

Method

Phase 1: Identification

Steps 1A+B. Generating a list of expected users, and an initial list of tasks.

In this phase, you interview knowledgeable people about their real-world tasks and observe them doing their tasks. Your goal is to generate an initial list of concrete task descriptions (see Appendix 1).

Depending upon your situation, you may or may not be able to access your real users in the short amount of time available before tutorial presentations. Consequently, your team should select the approach below that best fits your constraints.

i. *The ideal: Interviewing the user.* Get in touch with current or potential users. These users may now be using manual paper methods, competing systems, or antiquated systems for doing their tasks. Interview them about why and how they do their work activities, and what they expect out of a system. Ideally, this interview will occur while you are observing them do their work activities. These interviews and observations will give you some contact with actual users and give you a feel for the real situation. This is more important than you think, for it will make you realize that 'the user' is not an abstract notion, but real people with real needs and concerns. It will help you put a face on the faceless, and will help you understand where they are coming from.

ii. *A reasonable alternative: Interviewing the user representative.* When you cannot get in direct contact with end users, you can use users representatives instead, or people that have insight on the task. These will be people who may have the most knowledge of the users' needs. Examples are help desk people, or a worker's manager. However, it is crucial that the user representative has a deep and real (rather than idealized) understanding of what the users actually do. People who work "in the trenches" with the staff are the best bet.

iii. *When all else fails: Making your beliefs of the task space explicit.* If you cannot get in touch with either real end users or representatives (which unfortunately is quite expected given the covid crisis), use your own team's knowledge, and your family, collogues or friends to articulate expected tasks. While this runs the risk of producing tasks that bear no relation to reality, at least you will get a diverse list of tasks out, and will at least provide a concrete description of who your are developing for and what they plan to do. Note: If you don't have enough time to get in touch with actual users in time for your project presentation, try to get in touch with actual users in the coming phases of your project, if possible.

For whatever approach you chose to employ, make sure that you complete the following steps. If you have a user and/or representative, you would complete these steps with them. If you are "making it up", try to imagine as realistic a situation as possible.

a. Have the user/representative recount a few (3-4) stories that typify the actual use of their system and/or process. Where possible, describe the people, the particular problems they wanted to solve, what information they brought into the meeting, the steps taken in the actual process, the constraints they had (e.g., time), what was produced, and whether they were satisfied with the outcome. All details are relevant so make sure that you are thorough in your investigation. Alternatively, the task could be derived from direct observation of users doing their work. Ideally they should be based on a combination of both approaches.
b. On a more general and less detailed level, list as many related tasks and their variations as possible.
c. There will be many task variations in the list. Identify (with the user, if possible) which tasks are frequent, which are infrequent but still important, and which are rarer and not very important.

At this point, you will have a set of concrete, detailed examples of tasks that people now perform, or would want to perform on your system. Each task description should have the attributes described in the Appendix 1 and the lecture.

Step 1C. Validating the tasks.

The next phase is to get a reality check of your task list. Have end-users and/or client representatives review your tasks. They should check to see if the set of expected users you came up with is an actual reflection of potential end-users of your product, if the tasks capture the variations of those done by real people, and if details are realistic (they will be, if they are based on real people!). You should ask for details that were left out of the original task description, get corrections, clarifications, and suggestions, and then re-write the task descriptions.

Reminder: The validation step is especially critical if you used a user representative, alternative (ii) or just your teammate, family members or friends, alternative (iii) instead of a real user. While it may not be possible for you to interview and observe many real users, you can probably get one to comment on a compiled list of prototypical tasks.

Phase 2. Deciding upon key users and a tentative list of requirements.

The task examples will provide clues on specific system requirements that you could use to design your system as well as who your target users will be. Because it is unrealistic to meet all requirements and address all users, it is your job to prioritize them. From the task examples (and possibly by further discussion with end users), decide upon the major system requirements and prioritize them into a) absolutely must include; b) should include; c) could include; and d) exclude. Similarly, decide upon what kind of users you must address, as well as those that you will exclude.

Phase 3. Develop low fidelity prototypes.

From the task examples and requirements, you should sketch out several competing interfaces. Discuss and choose the most promising of these, and develop a low-fidelity prototype (using storyboards or Pictive methodology) that demonstrates how the interface fulfills the requirements. See Appendix 2.

Specifically, use the key users, their tasks, and the prioritized requirements as a type of requirements document to help you brainstorm prototypes that illustrate how your system would appear to the user. You should be creating low fidelity prototypes e.g., paper sketches, storyboards, or Pictive (you can try a different method for each prototype if you are really keen!). You should not be concentrating on prettiness or completeness; rather, you are trying to show the overall interaction style of your system. Each prototype should contain the core screens that illustrate how the system will work as a whole, including a sample interaction based upon some of the key tasks.

Phase 4. Task-centered walkthrough.

Test the prototype for usability bugs (problems, faults, weaknesses) by performing a task-centered walkthrough. For your submission, please include full walkthroughs of **only the final** of the low-fidelity prototypes you developed.

Assignment 1: Frequently Asked Questions

Q1: The grading sheet indicates that we need to do "approximately" 5-7 task examples. So if my team has only 4 or less, I would only need 4 or less walkthroughs, right?
Answer:
+ You would only need to do as many walkthroughs as there are written task examples (hence the required number is approximately 5-7 walkthroughs). However, if your team misses some important task examples, not only will you lose marks in the task section but also in the walkthroughs section.
+ When your TAs mark your assignment, they don't simply count the number of task examples or walkthroughs you have. They read the content and give your team a mark based on the quality of your work rather than quantity. The numbers on the grading sheet are used to give you an idea of the scope of the assignment.

Q2: The table format in the walkthrough section takes up a lot of space, so can my team go over the 20 pages limit?
Answer:
+ In this case, yes. Note that the assignment states "should ideally be no longer than 20 pages". We know that the table format takes up a lot of vertical space, so it is okay to go over the 20 pages limit if you must (but please don't go over 25 pages). Again, more pages do not necessarily mean a higher mark. It is the content of your portfolio that matters.

Q3: What should be placed into our assignment portfolio appendices?
Answer:
+ The appendices are not included in the assignment‘s page count, so your team can use this space flexibly, allowing you to reflect on efforts that are perhaps not fully represented within the required, page limited, core part of the portfolio.
How should you use the appendices exactly?
+ Include your various low-fidelity prototypes, sketches, pictives, storyboards in an appendix.
+ include your early prototype iterations, on top of the ones that you are using for the walkthrough, in an appendix.

Q4: How should early iterations of the prototype be included in an appendix?
Answer:
+ To allow your TAs to evaluate these early iterations, each of the early prototypes should contain the core screens that illustrate how the system will work as a whole, including a sample interaction based upon some of the key tasks.
+ For each of the early prototypes write a brief description: How did this version improved previous iterations? What were its strengths and weaknesses? What further improvements had to be made?

Appendix 1: How to Write Good Task Examples

Don't forget to read the class notes and accompanying papers on Task-Centered System Design.

Good task examples:

- Say what the user wants to do but does not say how the user would do it
    - You are not to make any assumptions about the system interface
    - We will eventually use this to compare different interface design alternatives in a fair way
- Are very specific
    - Says exactly what the user wants to do
    - We will eventually use this to specify the actual information the user would want to input to a system, and what information they will want out of it
- Describe a complete job
    - Should list all aspects of the task, from beginning to end
    - This forces designer to consider how interface features will work together
    - We will eventually use this to contrast how information input and output is carried through the dialog, i.e.,:
        - Where does information come from?
        - Where does it go?
        - What has to happen next?
- Say who the users are
    - Use actual people
    - The tasks reflects real interests of real users
    - The success of a design is strongly influenced by what users know and their real work context; we will eventually use this information to see if people realistically have the desire, knowledge and/or capabilities to accomplish their task with the system.

Appendix 2: Developing and Evaluating Initial Prototypes

Don't forget to read the class notes and accompanying papers on prototyping.

Step 1. Developing low fidelity prototypes.

Use the key users, tasks, and system requirements generated previously as a type of requirements document to help you brainstorm prototypes that illustrate how your system would appear to the user. You should be creating several *low fidelity* prototypes e.g., paper sketches, Storyboards, or Pictive (try a different method for each prototype!). You should also be thinking about what *conceptual model* the system will portray to its customers. You should not be concentrating on prettiness or completeness; rather, you are trying to show the overall representation and interaction style of your system. Each prototype should contain the core screens that illustrate how the system will work as a whole, including (perhaps) a sample interaction based upon some of the key tasks.

If you are really keen you can read ahead (see recommended readings, or approach Ehud) and use the ideas from the "The Psychology of everyday things" and "Information visualization" lectures to help you, as well as your general knowledge of other systems (there is nothing wrong with copying ideas, providing its legal!).

Hints: To get diversity, you and your teammate may want to try and create a few rough sketches separately before gathering as a team and brainstorming your approach.

Step 2. Evaluate the prototypes.

- Discuss each prototype to see whether it is a possibility in principle (e.g., are there obvious problems with the conceptual model? Can it actually be implemented?)
- Perform a task-centered walk through for each of your key tasks, and each of your user types. You can find additional details in your notes from class (see the Cheap Shop example).
- From the designs that are left, elicit reactions and further discussion from customers / counter people / appraisers. You may find that your end-users will tell you about further tasks and task details which were not thought about before, as well as come up with suggestions about your designs!

Step 3. Reconsider priorities, and make a preliminary decision.

Based on the prototype and evaluation exercise, you may wish to reconsider which users that you will address as well as what tasks and requirements you will support. You may find that you were wildly optimistic about what you could actually achieve!

At this point, you should have a reasonable idea of which prototype styles are worth pursuing, or whether you should start again. Make your decision on what direction to follow. If you have more than one direction, you may want to continue developing both a bit further. If you have no worthy candidates, return to step 2.

Hint. Don't feel committed to any prototype. This is not the period of time where you will be making final design decisions, rather this is a period which enables developing prototypes that are quick to generate and cheap to discard. Use this time to explore the design space. While you may want to just get on with the process, a bad design choice now can have disastrous and expensive repercussions later, when changes are more difficult to make.

Step 4. Refinement and evaluation, to be started at the final assignment (that is, starts only at part 2 of this project)

Refine your prototype by considering the nuances of each task, the functions that must be incorporated, and the expected reaction of each user. You may want to start considering the more subtle aspects of interface design at this point (e.g., The Psychology of everyday things, principles of graphical screen design, as well as the other design principles discussed over the course of the term). You should be continually evaluating these prototypes as appropriate. Real users should be commenting on them. You should be using walk-throughs, heuristic evaluations, and various observational methods.

If you follow this process, your prototypes will evolve from a competing series of design ideas, to a crude single design representation, to more realistic looking screen designs, to functioning systems. As your design is refined, you will move from very low fidelity prototypes done on paper to medium fidelity prototypes done on-line, likely through an interface builder. You will have detected and corrected the major interface design problems, and will be concentrating on fine-grained details. Eventually, you will create vertical prototypes with back-ends that simulate (i.e., fake) some of the system functionality. To the user, however, it will look like the real system.

Appendix 3: Projects List

This page lists several project ideas and system situations that we may ask you to use for grounding your design project. Of course, other projects are possible. The project will be assigned to you by your project advisor (if you feel strongly about a specific project topic please touch base with your TA directly).

Important: Ideal projects are ones will have users you can talk to, requires some transaction processing, and will require a modest level of design sophistication. Ideal projects will also deal with fairly simple situations - avoid overly complex domains (that rules out air-traffic control systems) and ones that will allow you access to the users and their place of usage. Ideal projects will also be something you can relate to. Look at things that you and your friends do (e.g., in part-times jobs), or that family members do (e.g., as part of their work) or non-work fields: recreation; systems for kids; education; games. Have fun!

<u>Remember: focus your efforts on the interface design</u>

Project ideas:

Your company is about to design, build, and market a new interface based upon one of the following ideas:

1. Cooking Instructor: design a system that can be used to prepare and cook a recipe.
2. The city of Calgary hired you to design a public information kiosk for people new to the city. For example you can decide to focus on new immigrants.
3. Bus ticking kiosk, designing a system for travelers in the terminal building. A bus company would like to decrease line-ups for clerks at a bus terminal. Their idea is to allow travelers to look up major route schedules, book tickets, and pay for them via debit/credit cards on the computers located in the terminal building.
4. Cross-family communication system: design a system to allow families with children to communicate with their remote grandparents.
5. Electronic course registration system, can replace components of the U of C Student Centre: make sure you narrow down and tackle only a limited and focused part of the registration system.
6. A company wants to develop a personal fitness tracker, an interface that will allow users to track their physical activity, training and fitness program progress.
7. A company wants to design an ordering system for restaurants/bars, allowing users to interact with the menu and order their food from their table directly.
8. Design an HTPC (Home Theater PC) interface which is addressing special populations, for example, the elderly, or young viewers. Users want to be able to easily navigate, view and record various streaming media such as netflix, youtube, cable TV, music files and photos.
9. Create an interface for a social network based on a specific set of users with a narrow common interest, for examples, FPS gamers, D&D players, golfers, English horse riding amateurs, Karatekas, e-sport fans, etc. Users would like to be able to generate,

build upon and relate to their social connections, within an overall theme that is reflecting the narrow specific common interest of the group.

10. A photographer with a huge set of photos in her library wants a system to help her track the photos (which includes adding and deleting photos) and to manage sales of them.
11. A medical clinic would like to give its front-line staff a tool that will help them manage patient requests for appointments with particular doctors.
12. A public library wants to provide a system that lets customers browse the inventory and for staff to check out and check in books, blurays and audiobooks items. (Hint: You should focus on either the customer end of the system or the staff functions, not both).
13. A travel company would like to have an app that helps their clients during their trip (post-covid). The program helps clients view their itinerary, book upcoming local events, and find local amenities (tourist hotspots, emergency contacts, etc). Hint: Choose Calgary, or a specific travel destination you are familiar with, for the scope of the design.
14. Design a kids-friendly Web Browser: design a web browser for young children.
15. A local convention hosting company would like a system that helps their patrons navigate their convention. The program allows patrons find/view scheduled events, find amenities (help desks, ATMs, rooms), and plan their day. Hint: Choose a multi-day convention/festival with a common interest to you (e.g. Otafest, assume we are post covid…)
16. A real-estate company wants to let customers browse through home listings from their homes. The program lets customers indicate properties of homes they are interested in, view details of homes that fit these criteria, and select ones that they may be interested in visiting with a realtor.
17. A theatre wants to develop a new ticketing system for their clerks. Through the clerks, customers order tickets for various events. The clerk responds to customer requests for information as well as actual ticket reservations and sales via the system. Variants on this can include a system allowing customers to buy movie tickets online, or at the entrance to the movie theatre (seat selection, etc.)
18. A gas station wants to computerize its manual system, where attendants can use the machine to observe the pumps, to collect payment for gas and other supplies, to track special events and promotions that may be going on, to track and visualize sale and inventory information and so on.
19. Redesign the ticketing system for city transit. There are different aspects that can be focused on in this area, think about the different ways of buying a ticket and using a ticket. City transit wants to know how technology can improve the experience for transit passengers.
20. A dance company would like an application that would help them develop and refine the choreography of their dance routines. Develop an interface that would allow the group to pre-visualize a dance routine, record iterations of the choreography or develop staging in conjunction with the dance.

Task Centered Design and Prototyping Grading Sheet: Be sure to include it in your portfolio

Student Names and emails ____

Note: The list below is a set of guidelines, or a "convenience" checkpoint. <u>Getting many satisfactory checks does not necessarily indicate a good project (or vice versa)</u>.

Structure and format

	Included	**Not included**		
Portfolio in PDF	1	0		
Section separators	1	0		
Name on outside cover	1	0		
Name and contact information on the first page	1	0		
This grading sheet included in portfolio	4	0		
	Complete	**Missing portions**	**Not included**	
Table of contents	2	1	0	
	Great: no problems	**Good: a few minor problems**	**Poor: Problems throughout (your mark in other sections may also be affected as well)**	
Appearance (organization, layout and whitespace)	6	4	0	
	No typos, grammatical or spelling errors, clear writing style	**Minor typos or grammatical errors or spelling mistakes or some writing may be a bit vague**	**Problems in two areas (spelling, grammar, style)**	**Problems in all three areas**

Language and writing style	7	5	3	0

Setting the stage

	Clear and complete (yes)	Clear and complete (no)
Background	1	0
Expected uses of the system	1	0
System constraints	1	0

	Lists user groups along with relevant skills and experience	Lists user groups with no additional information	Information not included
Expected users	2	1	0

	Clear & complete	Some information missing or unclear	Information not included
Work context	2	1	0

	Spoke directly with actual users	Spoke with a representative of the user	Made it all up
Approach for getting background information for tasks	2	1	0

Tasks

	Appropriate No. (~5-7)	Fewer than what's needed for the usage of the system	No tasks were included in the portfolio
Number of tasks	2	1	0

	Covers all relevant activities	Missing a few important tasks	Missing many important tasks	No tasks were included in the portfolio
Coverage of the tasks	8	6	2	0

	No violations	**A few minor violations**	**Many violations throughout**	**No tasks were included in the portfolio**
Do the tasks follow the properties of a good task?	8	6	2	0

Prototypes

	Two or more	**One**
Number of versions/iterations	2	1

	Marked improvement from version to version	**Few and/or superficial changes from version to version**	**No evolution between prototype versions**
Evolution of prototypes	6	2	0

	Provides clear idea of how prototype changed from version to version	**Describes changes but some parts are unclear**	**None**
Description of how prototypes evolved	4	2	0

Requirements

	Requirements are grouped into categories with clear and detailed explanations based on the users and their tasks	**Requirements are grouped into categories, no indication of how functions were put into particular categories**	**Requirements are shown in a single list, no attempt at prioritization**	**No requirements listed**
Description of system functions to be implemented	5	2	1	0

Walkthroughs

	Walkthroughs for all relevant tasks	**One**	**Zero**
Number of walkthroughs performed	4	1	0

	Walkthroughs conducted, all or most usability problems were caught	**Walkthroughs conducted, some minor problems were missed**	**Walkthroughs conducted, many minor or several serious problems were missed**	**Walkthrough not performed**

Results of conducting the walkthrough algorithm	10	8	4	0
	Walkthrough results summarized for each scenario/task **An analysis conducted that summarized for all tasks what are the high level and major problems**	**Walkthrough results summarized for each scenario/task but not for all tasks**	**Walkthroughs conducted and results shown in table but no additional analysis, summarizing problems**	
Analysis of walkthrough results	6	3	0	
	Walkthroughs easy to follow (e.g., included diagrams at all relevant points of walkthrough, diagrams are annotated)	**Some points of the walkthrough difficult to follow (e.g., walkthrough description didn't match interface, additional diagrams would have made things clearer)**	**Walkthroughs not conducted**	
Ease of following/tracing the walkthroughs	6	3	0	

Tutorial presentations

	Provides clear background information, good tasks presented, requirements properly categorized	**Minor problems: some background information unclear, minor violations in the descriptions of the tasks, requirements could better justified**	**Poor: task violate many properties of good tasks, or background missing or largely incomplete, requirements are not justified**	**No presentation**
First presentation: Phase one and two	4	3	1	0
	Walkthrough: caught most problems, clear indication of what future improvements should be **Prototype: Gives a good feel for how the interaction unfolds, covers main system functions**	**Walkthrough: Missed a few minor problems in the walkthrough** **Prototype: Some parts of the interaction unclear, a few minor system functions (relevant to task) or a major function is missing**	**Walkthrough: Missed many minor problems in the walkthrough or a few major usability problems** **Prototype: several main system functions missing**	**Walkthrough: Many serious problems were missed in the walkthrough** **Prototype: main system functions were missing**
Second presentation: Phase three & four	4	3	1	0
All team members completed all weekly MVP surveys to portfolio due date	Complete	Incomplete		