f2

Assignment 1

Jeremy Martinez
jmartinez91@gatech.edu

1 QUESTION 1

When evaluating interfaces based on the processor model, predictor model, and participant model, Piazza stands out as a tool that does some things well but could gain immensely from putting more emphasis on the other two. Piazza is a tool that is designed to act as the classroom portion of a massive open online course (MOOC). Piazza has two main components to its platform, the sidebar and the main thread which highlights whatever discussion you've selected from the sidebar (defaulting to some discussion if none selected). In order to achieve this the tool must address the following, among other things,

- Facilitate discussion on relevant course topics
- Provide a platform for a student to interact with their instructors/teaching assistants
- Provide a medium in which the instructor can make class-wide announcements

1.1 Piazza from a processor model

The processor model treats the user as another (computer) processor in tandem with the interface.

This asks objective questions on functionality like whether a user is able to complete a given task, and in what amount of time or how effectively.

An interface that does well in the processor model will fit within human limits. In terms of an interface, it is physically visible and comprehendible to the user. This is evaluated with quantitative measures, like whether they can complete some task and how quickly? If A/B testing, with what percentage of completion does one method (method A) have over another (method B)?

Objectively speaking, Piazza accomplishes the task of hosting a virtual classroom because all of the functionality is there.

- The main thread facilitates discussion on whatever thread the student is concerned with. However, this discussion does not happen in real time, and so engaging in a debate-like conversation is lost in this platform or is drawn out over several hours/days.
- The student has the ability to post questions/discussions/polls/etc. for the entire student body, only instructors, or only teaching assistants.
- Piazza has a priority queue that only the instructor is allowed to post in which serves as a logically separated area for class-wide announcements

1.2 Piazza from a predictor model

The predictor model takes into account cognition of the user of the interface and how well they are able to predict aspects of the interface. This evaluates how well an individual can predict what the outcome of an action will be, whether that outcome reflects what you were expecting, and if the next call to action follows intuition. This involves getting inside the user's head, what they're thinking, what they're seeing, and feeling during a given task. Measuring an interfaces success at this often entails what a user-researcher might evaluate on a UI/UX design team.

An interface that does well in the predictor model will map user's inputs to outputs in an intuitive way. The interface must fit with the knowledge of the user. This means, it must help the user learn what they are trying to learn, and efficiently leverage what they do already know. Contrary to the processor model, we evaluate this with qualitative results

- It is not clear to a user if/when a user will respond to their post, and it is difficult for the user to predict when they will be able to engage in that discussion (having been responded to). There is not an efficient alerting system with this either.
- The interface in which to post a new discussion is intuitive, not too cluttered, and easy to edit making it straightforward to use. Piazzas interface with this task matches existing discussion platforms, which leverage the users existing knowledge well.
- A user will take the path of least resistance, so having higher priority discussion (announcements from the instructor) always queued to the top allows the user to predict what they should view first and foremost.

1.3 Processor model vs predictor model

From a purely functional standpoint (processor model), Piazza does many things well. The real time discussion is a limiting factor, and so adding a feature or required online period could address this. From a predictor model argument, enhancements to this feature would include a more rich discussion environment, which I will use Slack as an example

- Adding profiles of students in another sidebar with some indication that they are online would allow students to leverage real-time discussion with students they can see are currently online
- Adding thought bubbles/typing bubbles to show another student is in the process of responding would allow the user to anticipate when their question will get a response
- Giving the user some indication that their posts have received activity (intelligent alert system) would enhance these discussions as well

While some of the discussion above in regard to the predictor model discusses other students, I want to emphasize that this differs from a participant model, since I am only referring to a user's interaction with the discussion board interface. The discussion board inherently encapsulates a response to a user's post, so it is impossible to contemplate the discussion board without considering another post in which the user is responding to.

2 QUESTION 2

2.1 Rosetta Stone

Rosetta Stone is a web-based application for learning a foreign language. They're primary platforms are designed for desktop browsers and a mobile application. Contexts that this tool is used in include, but are not limited to,

- Non-distracted modern browser use
- Non-distracted legacy browser use
- Distracted browser (both modern and legacy) use
- Distracted mobile use
- Non-distracted mobile use

2.2 Context analysis

Non-distracted modern browser — Perhaps the optimal context to use Rosetta Stone in terms of memory retention, speed, and ease of use would be with a modern browser without any distraction. In this context, the user has no distraction, can pay full attention to the task at hand, and maximize their retention in the vocabulary, pronunciation, grammar and speech comprehension.

One could make a similar evaluation of non-distracted mobile use in a secluded environment. While the interface will change significantly, the user will still be able to accomplish the same modules. One (and possibly the only) area the experience may be slightly less desirable would be with the writing portion and not having a keyboard at their disposal could make this cumbersome and time consuming.

Non-distracted legacy browser — Slightly tweaking this the previous context to accessing Rosetta Stone via a legacy browser introduces some issues from a functionality standpoint (more toward an analysis of the processor model in this context). If one were to access this application in an old version of internet explorer, the application may not function properly rendering certain tasks unachievable. This will become even more apparent after 2020 when Firefox and Chrome cut support for Adobe Flash (the platform Rosetta Stone is written).

Distracted browser/mobile — Distracted use (whether this be mobile or desktop) introduces a problem with this task since not having one's full attention inhibits memory retention. This becomes an even larger concern when considering distracted use on mobile. When one is seated and using a laptop/desktop computer, there are fewer options for distractions. However, with a mobile phone, this can be used on the go, which invites many other forms of distractions.

Case study - mobile — Consider a scenario where someone has a long commute to work (via the bus). They decide they want to spend this commute time more efficiently by learning a new language, and so they (pay for and) download the Rosetta Stone app (which has amazing reviews 5 stars with 72k reviews on the Apple app store). They start working through the modules on a bus ride and (unexpectedly) come to the speaking portion of a given module. However, they think it will look strange to start speaking out loud in a foreign language to

themselves at 7 AM on the bus, and so they skip this portion of the module. If this is the only context in which they use this application, they may develop their understanding of this foreign language in a lopsided manner with their speech lagging far behind speech comprehension, reading, and writing. This UI/UX is not ideal and will result in their users not developing well rounded foreign language skills which could reflect poorly on their reviews.

2.3 Context recognition/adaptation

As far as context recognition is concerned, this is very easy to do, and I am sure Rosetta Stone is doing so already. This is achievable via HTTP headers, and other operating system recognition tactics that exist in most development platforms. In order to prevent their users from having a bad experience in a browser, they can gate the functionality of their application and prompt the user with a message, "please upgrade your browser version to continue." They could also attempt to support legacy browser's, however, this may require them to rewrite their platform from scratch (without Adobe Flash).

For enhancing the mobile experience, it is difficult to circumvent the speech portion, since you don't want to write it off entirely. However, they should be able to toggle a setting that says suppress these modules for now and come back to them later. As it stands now, sifting through unfinished modules and repeating sections is cumbersome and unintuitive. My guess is that many users never revisit these and just move on to the next module leaving the previous one incomplete.

3 QUESTION 3

The gulf of execution can be summarized in the question, "how do I know what I can do?" How the user figures out what actions to take in order to accomplish a desired outcome are at the center of this. How difficult is it for a user to accomplish their goal. The context under analysis is the goal of submitting an assignment through Canvas.

- Identify intentions: What their goal is in the context of the system
- Identify actions: Once their intentions are clearly laid out, what actions are necessary to accomplish this goal?
- Execute in interface: Once they've identified those actions, how do they execute these through the interface?

Identify intentions — A user's intentions in this goal (submitting an assignment through Canvas) is to upload and submit some deliverable (outlined in the assignment description) through a submission portal. In the context of Canvas, there is a sub context for each course you are enrolled in. In my experience, the landing page has always been clear in that I am already in my course's section of the website, or I need to select a tile to narrow in on that course. Once there, the sidebar offers a list of options to choose from, none of which say assignment submission. There is a tab that says "assignment," and so the user is required to further refine that context. Providing common actions associated with tab will help associate intentions with tabs

Identify actions — Intentions and actions here are terms that are being overloaded a bit, since our goal/intention is the action of submitting an assignment. Canvas has identified the intention and reduced the number of tasks for the user to situate themselves in the proper context down to two/three clicks. Once they are able to navigate to the proper sub-context, the call to action must be clear and follow the intuition of the user. Considering the user in the predictor model, we can assume they are familiar with boilerplate submission forms. So the only real task of the UI on the submission page is to ensure the actions lead to accomplishing their goal.

Execute in interface — Once the user has identified and comprehends the call to action on the submission page, they need to be able to execute these without any friction. Canvas assumes their audience has used a submission form before, and so this is relatively straightforward given web standards. However, Canvas falls short fairly significantly in one key aspect of this. Part of executing an action in an interface deals with recognizing your action was consumed by the interface. This is often indicated through a dramatic change in the UI. Canvas fails to do this in two ways:

1. Attachment upload. The only indication to the user that their attachment was received (actually attached) is that the file name is listed above the submission button. This is relatively anti-climactic and is confusing to the user whether this attachment is still loading or only staged to be sent via an API call later. Further, the change in UI is so subtle, that if their eye did not catch it the first time, they may have missed this altogether. There needs to be some symbol putting the user's mind at ease that the work on

their end, in terms of attaching the necessary file, is complete and they can move on with their task

2. Assignment submission complete. The indication that this assignment is submitted, after attaching files and clicking submit, is some text (which is not emphasized in any way) and a green check on the right-hard sidebar. This is again, anti-climactic in that it is not the main call to action of the page. The user should be given a dramatic reassurance that the task they were just trying to complete has gone through successfully.

3.1 Communicating to the user their task is complete

While this was touched on in the previous sub-section, we will shortly recap here on how Canvas assignment submission performs in terms of the gulf of evaluation.

- Identify intentions: this can be assumed successfully complete if the user is able to navigate the UI to the proper context and identify overall intentions. Canvas does a decent job at this, however, it could do a more thorough and concise job by leveraging common icons and creating an association in their users mind. They can build associations of specific tabs/sub-navs by always correlating icons/symbols with intentions/actions
- Identify actions: Re-iterating the point above, Canvas can improve upon their UI in a concise manner by leveraging icons and correlating them with actions. Creating this association in their user's brain will lead to quicker comprehension the more familiar they become with their suite of products.
- Execute in interface: Putting more emphasis (and dramatization) on the fact the assignments are attached, and assignment have been successfully submitted will help convey the success of a user's accomplished goal.

4 QUESTION 4

For question four, we will compare a single intention across two applications and evaluate the gulf of execution. The intention is to purchase one stock of Apple (AAPL). The first interface in question is Robinhood and the second is ETrade. Before diving in, it is worth mentioning to ETrade's credit, that their application is immensely more complex than Robinhood which makes retaining a short gulf of execution difficult.

In the context of a stock trading application, identifying the user's intention is

captured. And so, we will breeze past this and hone in the second two phases of the gulf of execution. We will also skip past the set up of ACH bank accounts for deposit, since legality enforces a time-consuming process here.

In Robinhood — the array of actions to take is specific. The platform's primary intention is to buy/sell stocks. The user has one of three tabs at the bottom to choose from to navigate with clear, industry standard icons. The magnifying glass indicates search. One must navigate to this page, search AAPL, and select a number range (integer), and click (the only call to action) to confirm purchase. If funds are insufficient, the user is temporarily redirected to add more funds, then navigated back. The funds submitted to their account are immediately available to reduce friction and ensure conversion.

In ETrade — A user has an extremely cluttered browser application and must identify which action in the navigation tab will take them to a page to trade stocks. Prior to this, they must ensure their funds are in the proper subaccount as to make them available for trading. They must search for the stock, and tune parameters to specify every detail of their purchase (quantity of stock, purchase minimum, maximum, and many others). If funds are insufficient, the user must deposit funds, wait 3-5 business days, return, and try again.

Lessons to be borrowed — One would have to look at conversion rates between each app to know for sure, however, there are a significant more number of steps required in ETrade to accomplish the same task in Robinhood.

Part of what makes Robinhoods gulf of execution so much narrower is the overall applications narrower focus. It has a limited number of intentions baked into the product, so the user has fewer distractions/confusing options to choose from.

A major take away is that simplicity is key in comprehension (among many other things, brand/icon recognition and consistency was discussed earlier). Intuit has accomplished this by breaking Turbo Tax out into a separate application and keeping the UI's intent narrow and clear. ETrade could learn from this and build another ETrade branded application/interface with the sole intent of buying/selling stocks. This way, their product can leverage all the advantages of a large corporation while remaining agile like Robinhood.