Matthew Oliphant

CS-319: UI/ UX Design & Development

Southern New Hampshire University

August 15, 2021

**Project Two:**

**Explanation and Lender Cloud-Based System Recommendation**

Based on prior research into borrower and lender profiles from Kivas’ databases, I decided that it would be best to keep the borrowers’ UI as simple as possible so that the embedded device’s hardware doesn’t get overwhelmed. Flip phones lack some of the hardware that a smartphone contains, so it’s important for the UI to be designed to only utilize the hardware that a flip phone would contain.

Navigation is an important consideration when designing a UI. Flip phones have a very limited range of buttons, but enough to get the job done. Users can navigate around the UI by using the up, down, left, right, ok, back, and/ or menu buttons. Navigation between screens is intuitive; the clicked link will take the user to the corresponding page, and because the UI is so simple, the user should be able to easily locate and access a needed resource. I’m not aware of flip phones being able to change screen orientation, but should be designed to fit the screen regardless.

While designing the wireframe for the entry page, I only included pertinent links for the borrower to be able to conduct their business. The main features include a preview window that shows a snapshot of the users’ current loan requests, a menu that displays Kiva information, and several clickable links to handle additional borrower needs.

If the user clicks on the ‘Borrow’ link, they will be taken to a page that describes the process to get a loan. Next to that button is a menu for Kiva information; clicking this menu allows the user to choose which Kiva information they want to read. The last link in the top bar allows the user to sign into their account or sign up if they don’t have one. The top of my UI is very similar to the Kiva website homepage in an attempt to be consistent and guarantee accurate data. My design uses data from Kiva’s database, which aligns with Kivas’ business vision and mission statement by connecting borrowers and lenders to create a financially inclusive world and help underserved communities thrive.

I believe my design benefits the borrower by focusing on what they need. Someone who doesn’t know much about Kiva is able to use this UI to access resources on becoming a borrower. From there, they can click on a link to see all user stories and possibly gain insight. They are then able to create an account and request a loan. Once approved, all loans show up on the preview window, providing quick access to funding progress, and each loan can be clicked for additional details. The user has the ability to share their story with friends or on social media in order to draw potential lenders’ attention.

Adapting the lenders’ UI design to a cloud-based system first requires having an idea of what features the UI needs. Most of the time, lenders have more capable embedded devices than flip phones, so their UI can be a little more robust. A possible feature could include using GPS location if the user story requires traveling and the borrow is willing to share that information. Another feature of the lenders’ UI is the touchscreen navigation, which is more intuitive than button navigation. Similar to borrowers, lenders need to access user stories so they can choose which stories the want to invest in. Lenders will also need to have a link that provides lender resources, and a preview window that displays stories they have invested in. Lender goals may vary from lending a one-time minimum payment of twenty-five dollars to lending a borrowers’ entire goal.

Cloud hosting embraces the client-server model, but the host provides the servers. Storing user stories is a complex task and the data structure used to do so needs to be efficient. When a borrower creates a user story, it needs to be added to the data structure and when a borrowers’ funding goal is reached, the user story needs to be deleted from the data structure. Although a singly or doubly linked-list provides a constant insertion and deletion time complexity, the average access time complexity is linear. A binary search trees’ (BST) average access, search, insert, and delete time complexity is log-linear. This data structure fills most evenly when the data set is random, so if the story Id is simply incremented from story to story, the data set will be sorted. In this scenario a BST would be a poor choice and I would recommend using a doubly-linked list.