User Experience (UX) Design Process, Paper Prototyping & Schema's for Firebase Realtime DB

User experience encompasses all aspects of the end-user's interaction with the company, its services, and its products.

How hard was the week 11 assignment?

A) Easy

B) Moderate

C) Challenging

D) Unreasonable

How long did week 11 assignment take?

A) Less than 2 hours

B) 2 to 4 hours

C) 4 to 6 hours

D) 6 to 8 hours

E) More than 8 hours

Not much time left

- **November 15th:**
  - Technology demonstration of novel feature
  - 4 Use cases
- **November 29th:**
  - Paper Prototype of user interface
  - Real-time Database implementation as tests
- **December 6th:**
  - Full GUI implementation
- **December 15th:**
  - Final integrated app

Technology Demonstration

- **Show us that you know how to use your novel feature**
- **Build simplest possible app that does this**
- **Use the internet (docs/examples), but CITE any code used!**
  - Also, make sure you understand any code used
- **Start early; you need to get it to work**

Code Reviews and Break.

- **If your code review is on Saturday, your moderator will contact you about re-scheduling your code review.**

User Experience Design is Hard

- **Most users are not like you**
- **Users can’t always tell you what they want**
- **But, they can sure tell you what is wrong.**
  - Consistent problems are the system’s fault

User-centered Design

- Big picture: What does your program do?
- Who are your users?
- What specifically do they want to accomplish?
- How should the interface be designed?
- do { Implement, Test, Refine } while (!done)

Identifying users

- **Often many types of users (many dimensions)**
  - Sophisticated vs. Novice computer user
  - Social vs. Private
  - Individual vs. Group
  - Time/Money trade-off
  - Beginner, expert (with your application)

- **Characterize space using “personas”**

Pizza ordering personas

- **College student:**
  - Not much money, eats at irregular times, no car, orders for self or shares with group.

- **Busy professional:**
  - Money >> time, eats at standard dinner time, has transportation, ordering for whole family

What do these people want to do?

- **Ask them!!!!**

- **Characterize as “tasks”**
  - Receive notification of special deals, order pizza for delivery, be able to have toppings on part of pizza.
  - Order pizza from office, pick up on the way home.

Decide how users will perform these tasks

- **Describe as "Use Cases"**
  - Descriptions of sequence of actions to perform a task

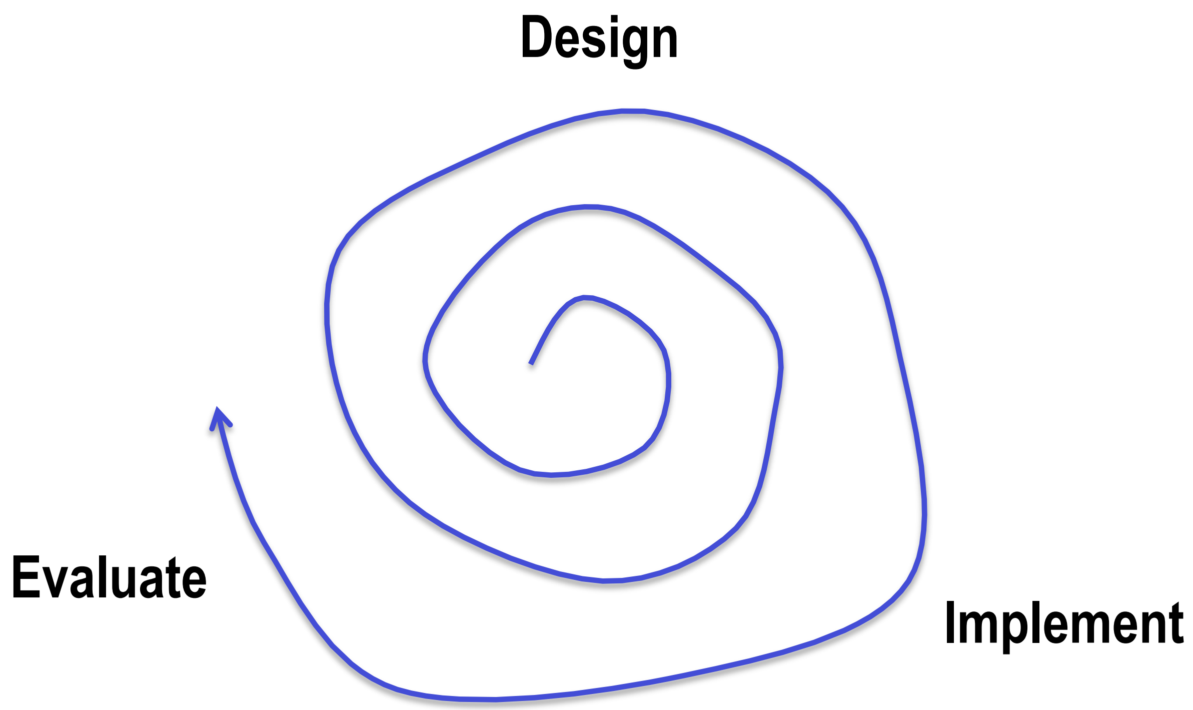
- **Name:** Contributing to a chat room
  - **Brief Description:** User appends message to the chat room.
  - **Preconditions:** The user is authenticated and has selected a chat room.
  - **Basic Flow:** User is presented with time-ordered series of chat messages. The user selects the text entry widget and types in a chat message. The user then clicks the "Contribute" button and the chat message is appended to the chat log and is visible by all participants in the chat room.

Then, Design -> Implement -> Evaluate!

- **Fresh subject each time**
  - No preconceptions
- **Give user a task to complete, with no explanation of how to accomplish the task**
  - Written down (avoid variation in communicating task)
- **Have them manipulate the interface**
- **Observe their decisions**
- **Ask for additional feedback.**
  - What is missing, frustrating, etc.

Spiral Model of Design

- Use throwaway prototypes and cheap evaluation early in the cycle.



Paper Prototyping

- **Cheap, low fidelity prototype**
  - Enables rapid iteration, 15-20 minutes to produce
- **Task prompt(s)**
  - Written down (avoid variation in communicating task)
- **Subject performs task (Wizard of Oz)**
  - Human simulates system (**doesn't explain!**)
  - Ideally, separate observer monitors subject; takes notes
- **Post-interview**
  - Any confusion? What is missing?

Real-time DB Schemas

- **Schema = Organization of Data**
- **A users table: This table is world readable, but only writable by the user and contains their current alias.**
  - users: [userID: user]
  - user: { "alias": String }

```
/users
    |--user1
    |     |--alias: "George"
    |
    |--user2
          |--alias: "Wilma"
```

Designing your Final Project's Schema

- **Questions to ask yourself:**
  - What data needs to be stored in the database?
  - What portions of the data need to be read together?

Real-time DB Schemas

- **Important Firebase Behavior for Organization Rules**
  - Think of the database as a tree
  - Can listen to sub-trees: **values** or **children**
    - Every DB reference either holds an **object** or an **array**
  - When we listen, we're notified if anything below changes
  - When we're notified, the DataSnapshot contains sub-tree
- **We want to minimize:**
  - Notification frequency
  - Size of DataSnapshot

Example Chat App

- Activity1: list chat room names
- Activity2: show all of the messages in that chat

- Problematic Candidate Schema:

```
/chats
   |--Places to eat in Chambana
   |     |--KWE-IUawhwmH5HoPBBz
   |     |     |--author: "user1"
   |     |     |--message: "I like Cravings even though they keep getting sanitation violations"
   |     |
   |     |--KWE-IUawhwmH5Hof801
   |           |--author: "user2"
   |           |--message: "Yes, their Fish Masala is notoriously good!"
   |
   |--Good CS classes
         |--KWE-IUawhwmH5Hof833
               |--author: "user1"
               |--message: "CS 233 teaches all about how computers work.  It is cool!"
```

Better Chat Schema

```
/directory
   |--KWDl7XvsgpU4FcGzo0y
   |    |--name: "Places to eat in Chambana"
   |    |--key: "KWDl7XvsgpU4FcGzo0x"
   |
   |--KWDl7Xvsgp3Mz71rSov
        |--name: "Good CS classes"
        |--key: "KWDl7XvsKguPe1wzruw"

/chats
   |--KWDl7XvsgpU4FcGzo0x
   |    |--KWE-IUawhwmH5HoPBBz
   |    |    |--author: "user1"
   |    |    |--message: "I like Cravings even though they keep getting sanitation violations"
   |    |
   |    |--KWE-IUawhwmH5Hof801
   |         |--author: "user2"
   |         |--message: "Yes, their Fish Masala is notoriously good!"
   |
   |--KWDl7XvsKguPe1wzruw
        |--KWE-IUawhwmH5Hof833
             |--author: "user1"
             |--message: "CS 233 teaches all about how computers work.  It is cool!"
```

Avoid Synchronization Issues

- **Example bad case: two users incrementing the same integer**

- **Avoid situations where multiple users modify same value**
  - Append only structures
  - Per-user data

Example Solution: Likes/Upvotes

- **Instead of keeping a count of likes…**
- **Keep an array of the users that have liked a given thing**
  - Each user appends their name when they like it
  - Removes their name from the list when they un-like it
  - Number of likes = number of entries in the array

- **No synch. problems; each user touches only their data**
- **Also can track whether a user already liked it**