# CS62 Lab 11: Coding interview practice / Final Project

Basic SWE skills

## CS62: Final Project

The final project will be an open-ended software development project done in **groups of 2-4** over the course of 5 weeks. Your group will apply methods in human-centered design to identify a problem that software may be able to solve in the real world. Your software project will likely fall into either one of two camps: interactive or data analysis. You will decide on the best data representations and custom data structures to solve your problem. You will make a public Github repo with a detailed README with usage examples for your software project. Lastly, you will create a PDF report that includes run time and affordance analysis for your software project.

The first 1-2 weeks of the project will be dedicated to needfinding and figuring out/scoping out your project and finding the data set. You will then have 3 weeks to implement your project.

The project is purposefully open-ended, which may be daunting, especially if you do not have prior experience in large unstructured software applications before. The instructors are here to help and give constraints and direction! The project is contract graded, which means you decide what grade you get as long as your group completes the promised work correctly by the deadline.

### Learning goals

- Apply the human-centered design process to software engineering
- Apply what you have learned to select (and customize and create) the most appropriate data structure for a problem
- Judge the appropriate scope of problems that can and can't be solved with computer science
- Analyze user-created algorithms for efficiency (time/space complexity), edge cases, and affordances

# Agenda

- Final project details
- Coding interview practice details

# Final project

- Open-ended software development in groups of 3-4 (you choose what you want to make!)
- 5 week period; contract graded
  - C - one “feature”, B - two+ features, A - three+ features
- Weeks 1-2: conduct needfinding interviews & think about how you’ll get data
- Weeks 2-3: get the data, spec the data structure, grading contract (part 1, due 4/22; check-in on lab 4/15)
- Weeks 3-5: (3 weeks after part 1) all code & writeup due; peer evaluation form (part 2, due 5/13; 1 feature done check-in on lab 4/29)

# Final project: interactive example

**Interactive software project example: Synced playlists**

Bella makes playlists they want to share with their friends, but they use Spotify while friend A uses Apple Music and friend B uses YouTube Music. While interviewing their friends and asking for stories about their music sharing experiences, they learned about turntable.fm, a (now-defunct) website that let users take turns live DJing and queuing songs for each other. Inspired by this anecdote, but still wanting asynchronous, any time music sharing in the form of playlists, Bella decides to build a Java app that will take as input a text file of Spotify or Apple Music or YouTube Music links, allow users to re-order songs in their app, and generates text files of the playlist in all 3 music sharing formats.

(Bella originally wanted to take as input just a single link to a playlist to read the songs from, but realized the music sharing services' APIs were difficult to hook up with Java. However, if your group wanted to take on a challenge like that, it would be grounds for an A+ on the project.)

First, Bella's group writes code to automatically extract song information (like Title, Artist, Album) from a given link using the Jsoup library. They use this library to also find the URLs on other music providers through parsing their search query (e.g., once given a Spotify link like https://open.spotify.com/track/6LgJvl0Xdtc73RJ1mmpotq, they scrape the title and artist to be Paranoid Android by Radiohead. They then can format a YouTube search URL like https://www.youtube.com/results?search_query=paranoid+android+radiohead and scrape the first result to get the URL to the YouTube video corresponding to the song).

Bella's group creates a custom Song data structure to contain all 3 links, as well as song metadata. They create a custom equals() and hash function which only looks at song metadata instead of the URLs. However, they realize the best data structure to store their songs in is not a hashtable but a doubly linked list, since the songs need to be ordered as a playlist. Their DLL supports moving songs around, and they also use a queue to implement the "add to queue" feature that asks the user for a new song URL which will output the song before the rest of the playlist.

The interactive main() of their Java app prompts users to input the name of a .txt file containing song URLs in playlist order. It then parses and prints the URLs of the other streaming services. Users can use their keyboards to order songs, add new songs to the queue, or save output.txt files for the other streaming services.

- Creating a synced playlist between Spotify, YouTube, and Apple Music songs
- Custom data structure to store the URLs and meta data of a song
- Doubly Linked List for playlist reordering
- Queue for "add to queue" feature

# Final project: data analysis example

**Data analysis project example: Group personal finance**

Alex, David, and Marina like comparing how much money they spent at the end of the month on their group chat. They wonder: why not just build a shared group finance Java app, that takes as input spreadsheets each of them download from their bank, and outputs some data analysis (e.g., who spent the most on eating out this month? How far apart was each of their spending from each other in each category? If the group set a budget to collectively spend less than $100 on entertainment, did they make it? What was each person's top 3 spend categories, and did it differ between them?)

From their interviews where they asked their friends to share and reflect on their past month's spending, they learn that some friends don't track their spending at all (and wished they did, as they found recurring subscription services they don't use), and some friends wanted to share with other friends but were too shy to overcome the social barrier of talking about finances, while some friends wanted bragging rights on who spent the least amount of money in their friend group. From these interviews, they also decided to implement a leaderboard feature and hone in on the specific kinds of data analysis their software project will do.

The group realizes each of their banks has slightly different categories for spending, so they first write a Java file to automatically clean their data so all the categories are standardized (e.g., one person's spreadsheet has the heading "Restaurants" while another has the heading "Dining Out", but these mean the same thing).

The group implements their app by using a custom data structure to store transactions. The Transaction Class has the spender's name, the name of the merchant, the date of the transaction, the transaction category, and the price. They use a custom Binary Search Tree that allows duplicate keys to store all transactions, where the price is the key (so the BST can have multiple $19.99 transactions, for instance). They debate also storing the Transaction objects in person-specific hashtables, so it is faster to access the Transactions for a single person as opposed to having to filter through searching the whole tree–it is storing redundant data, but they decide the trade off is worth it since many of their data analysis features are person-specific. They decide to use Java's built in mergesort to sort their data.

The group writes in their README what folder to put the spreadsheets in so their Java app can automatically read and clean them. When running the main() method, their software prompts the user to enter a number or letter on their keyboard corresponding to what analysis they want: for instance, pressing L shows the leaderboard of overall lowest spending per person, while pressing R shows each person's spending on restaurants.

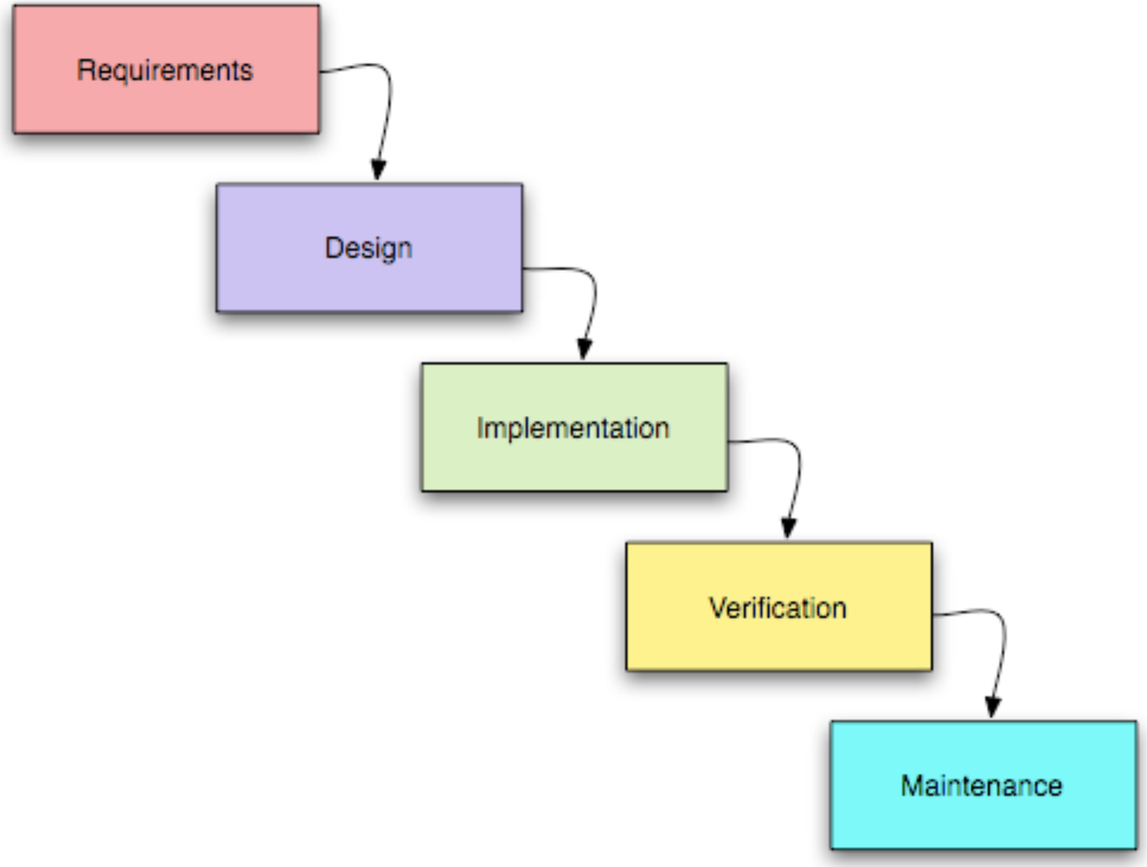
- Group personal finance analysis
- Custom data structure to store transactions
- Custom BST (allows duplicate keys) to store all transactions (key - price)
- vs hashtable to store transactions of a specific person and category

# Q: how do users interact with my program?

- Default: run main() and get user input from console (type 1-5 for options, etc)
- Extra credit: build a GUI (you can look into the Wordle GUI as a starting point)
  - For building a GUI, you can use LLM assistance (since it’s out of scope of this class)

# Models of software engineering

- Traditional waterfall model vs Agile development model



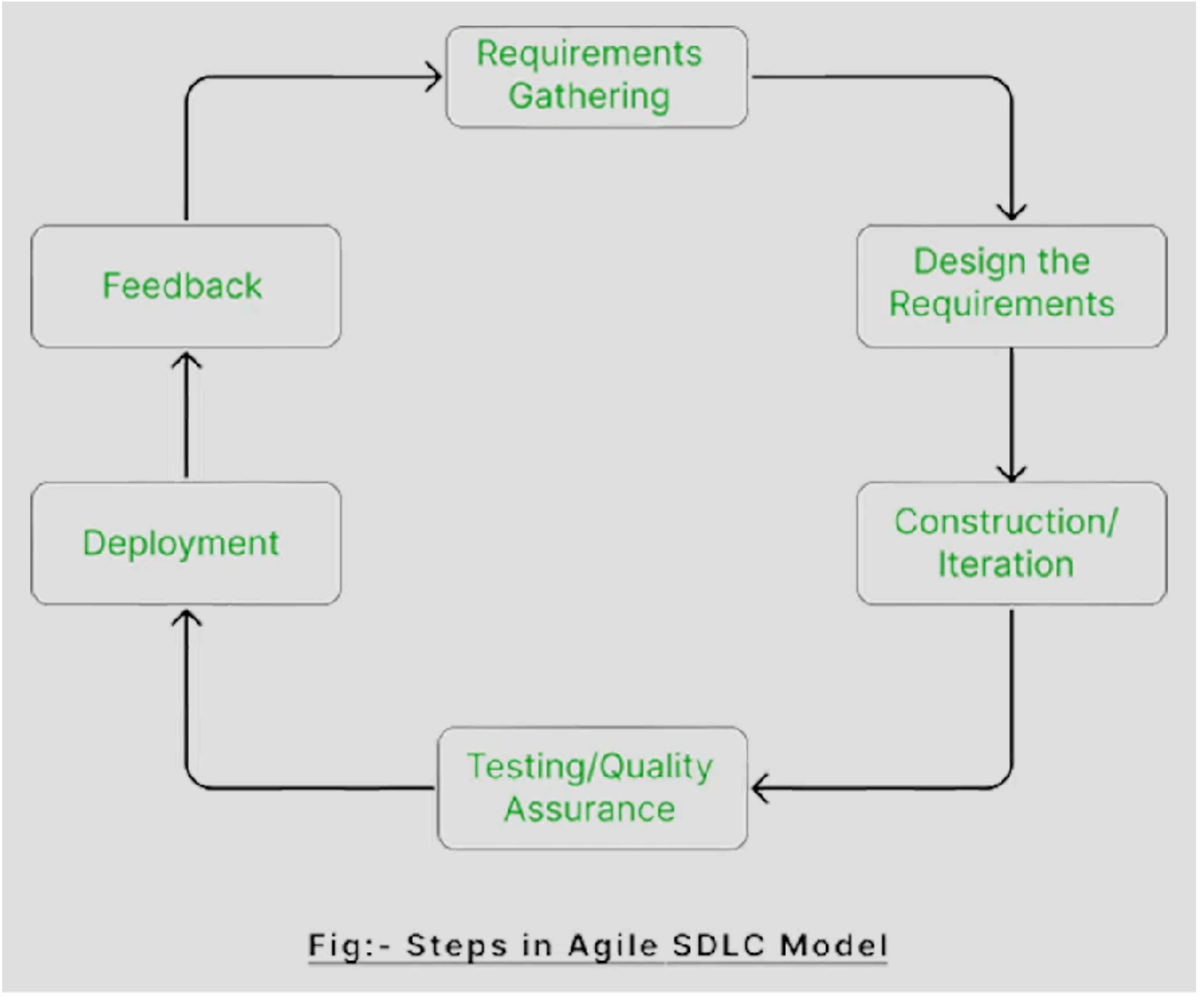
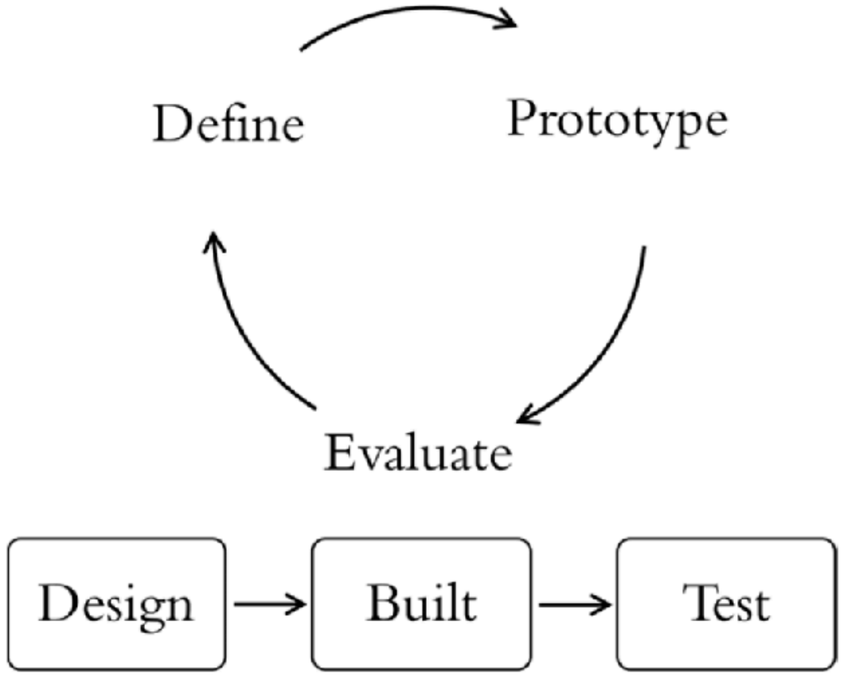
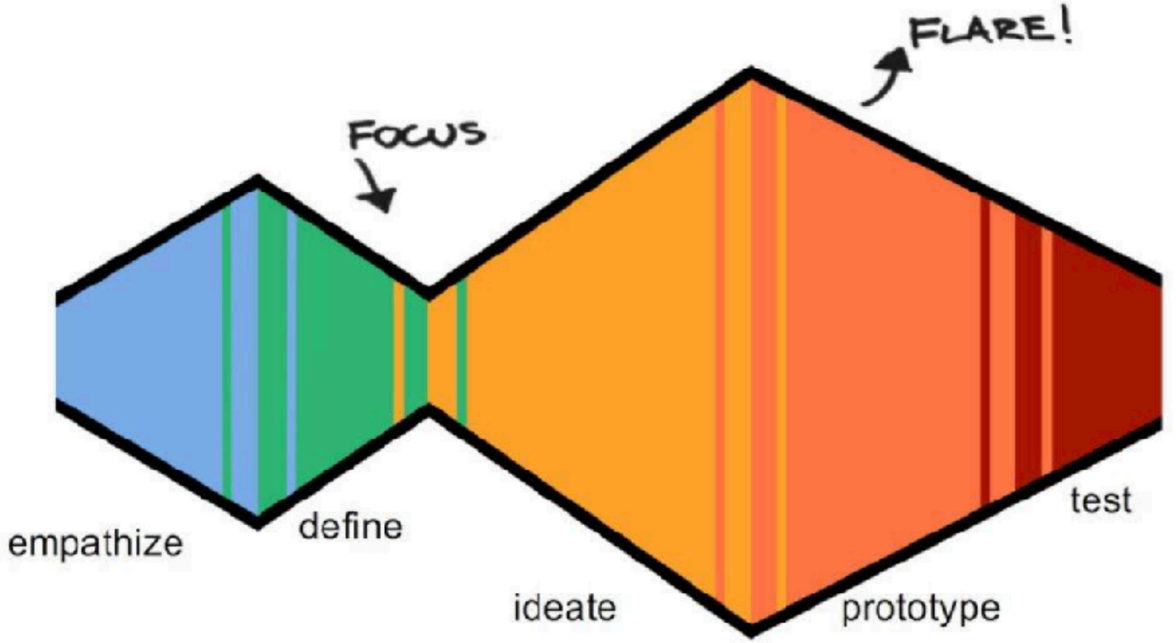




Fig:- Steps in Agile SDLC Model

# Human-centered design

- A *methodology* for building stuff that places the *user* at the heart of the process
- Also called user-centered design, or design thinking



# Needfinding

- Sometime soon (maybe after this lab?), spend 30 minutes with your group brainstorming general directions and interview questions
- Everyone conducts one 20-30 minute interview, come back together to synthesize
  - What were problems uncovered that might not be best for software solutions?
  - What were problems uncovered that you do think could be solved with software? Why is software a good match?

# How to conduct good interviews

- The purpose of your interview isn't to confirm an idea but to surface more problems/assumptions
- **Get participants to be specific**. Ask story-based questions: "tell me about a time you…"
- You can generate abstractions from details, but you can't generate details from abstractions

# Part I

- Hard part: How do you get an input dataset?
  - Does it already exist & can you clean it up?
  - For interactive software projects, can it be user created without too much trouble?
  - Can each member of your group contribute 250 rows to a dataset?
  - Can you manually annotate/create some data, and ask an LLM to generate 900 more rows within a range?

# Next lab (4/15) Part 0 check in

- Prepare a 5 min presentation that shows
  - The problem you want to solve and if your project is going to be more interactive or data analysis
  - Why you decided on that problem (eg from personal experience, from interviews)
  - The proposed format for your data & how you plan on generating the data
  - Initial thoughts about what data structures you will use
  - Any questions you have/advice you want going forward
- **If you don’t have a group by the end of week, message us ASAP! We’ll help!**

# Part I

- Due Weds Apr 22 11:59pm on Gradescope
  - Your grading contract (what 3 features are you proposing, if implemented correctly, will give you an A?)
  - Your dataset
  - Your .java Interface file proposing your data structure
  - A PDF that includes your project overview & grading contract (more details on the course website)

# Part II

- Due Weds May 13 11:59pm on Gradescope
  - A Github repo with your actual code, a README on how to run your code, an API, example usage
  - A PDF report
    - Needfinding summary
    - Results - show screenshots of your code running (e.g., print outs from main() for each feature)
    - Analysis: run time, space, and affordance analysis
    - Reflections
  - Self & peer evaluations

# Last point

- Doing a totally open ended project can be daunting, and it might be hard to scope what is an appropriately difficult feature/project
- That’s why we have 2 check-ins! Personalized learning! We are here to help. Please come to OH/mentor hours any time.
- This lab will also help you develop an intuition of choosing data structures to use

# Coding interview practice

- Get into groups of 2-3
- Each group will do a “systems design” style question on the board with the instructors or TAs
- While you’re waiting for your group’s turn,
  - Brainstorm ideas for final project/interview questions? (See webpage for guidance & details)
  - Do checkpoint 2 corrections
- You can leave once you get checked off (but you are welcome to stay to work on checkpoint corrections, etc.)

# Exit ticket!!!

- https://forms.gle/XR8YmbTyxJDvWgPi7