Final Project

PERATURAN UJIAN:

Exam Regulations:

- Mahasiswa tidak diperbolehkan berdiskusi dan/atau bekerja sama dengan peserta ujian lainnya
  Student is not allowed to discuss and/or work together with other exam participants
- Mahasiswa tidak diperbolehkan menyalin jawaban dari intenet
  Student is not allowed to copy answer from the intenet
- Asisten **BERHAK** memberi nilai 0 **(NOL)** bagi peserta ujian yang melakukan segala bentuk kecurangan
  Assistant is able to give 0 (ZERO) score for exam participant who does any cheating actions
- Kumpulkan jawaban tepat pada waktunya di GitHub
  Submit the answer on time at GitHub
- Bila Anda tidak membaca peraturan ini, maka Anda dianggap telah membaca dan menyetujuinya
  If you have missed to read these regulations, so you are considered to have read and agreed on it

---

SOFTWARE YANG DIGUNAKAN:

Software will be used:

- Windows Subsystem Linux (Ubuntu 20.04 LTS)
- Git & GitHub

FILE YANG DIKUMPULKAN:

File must be collected:

- Folder Project:
  - .pdf
  - .png
  - .cpp

PERHATIAN!

Attention!

- Bagi yang mengerjakan tidak sesuai dengan soal, maka akan diberikan nilai **NOL (0)**
  *For those who do not work in accordance with the exam case will be marked as **ZERO (0)***
- Bagi yang mengerjakan tidak sesuai dengan software dan versi yang telah ditetapkan, maka akan tetap dikoreksi dengan software dan versi yang telah ditetapkan
  For those who do not work in accordance with the software and specific version will be corrected by the predefined software and version
- Kompres semua jawaban yang akan diunggah. Pastikan format pengumpulan nama file dan ekstensi sesuai dengan format berikut: **[FINALE]-[NAMA].zip**
  *Compress all file that will be uploaded. Make sure the format for collecting file name and extension according to the following format: **[FINALE]-[NAME].zip***

Soal
Case

Obscure Binary Search Trees [7.5%]

Items, such as names, numbers, etc. can be stored in memory in a sorted order called binary search trees or BSTs. And some of these data structures can automatically balance their height when arbitrary items are inserted or deleted. Therefore, they are known as self-balancing BSTs. Further, there can be different implementations of this type, like the B-Trees, AVL trees, and red-black trees. But there are many other lesser-known executions that you can learn about. Some examples include AA trees, 2-3 trees, splay trees, scapegoat trees, and treaps.

Common Data Structure Operations

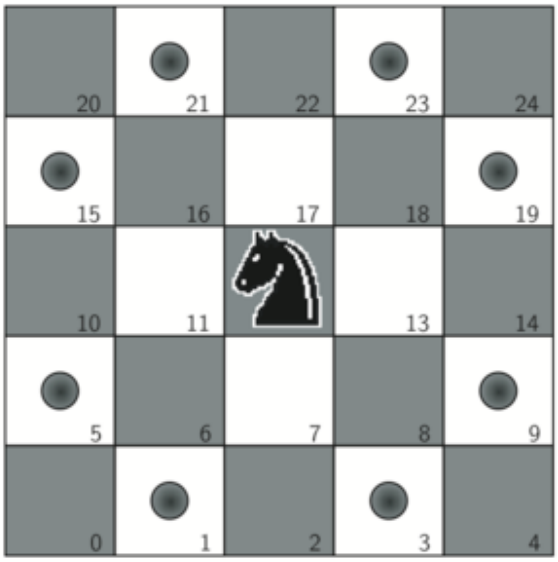
Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

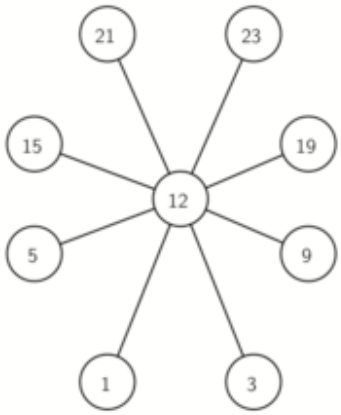
Choose one of these alternatives and explore how they can outperform other widely-used BSTs in different scenarios. For instance, splay trees is faster than red-black trees under the conditions of serious temporal locality.

Knight's Travails [7.5%]

We will understand two algorithms in action – BFS and DFS. BFS stands for Breadth-First Search and utilizes the Queue data structure to find the shortest path. Whereas, DFS refers to Depth-First Search and traverses Stack data structures.

Now, suppose that you have a standard 8 X 8 chessboard, and you want to show the knight's movements in a game. As you may know, a knight's basic move in chess is two forward steps and one sidestep. Facing in any direction and given enough turns, it can move from any square on the board to any other square.





Explain how to know the simplest way your knight can move from one square (or node) to another in a two-dimensional setup! You might want to draw the simulation in a tree data structure.

Text Editor [15%]

Regular text editor has the functionality of editing and storing text while it is being written or edited. So, there are multiple changes in the cursor position. To achieve high efficiency, we require a fast data structure for insertion and modification. Your project should include the functionality to insert or remove a character.

Tree Simulations [10%]

Simulate the following insertions and deletions using AVL Tree, 2-3 Tree and Red Black Tree:

a. Insert 80, 35, 20, 100, 25, 30, 45, 40, 50, 37
   Remove 35, 25, 30, 45, 80
b. Insert 40, 75, 50, 90, 60, 65, 63, 100, 95, 30
   Remove 63, 75, 40, 60, 95

Disjoint Set & Graphs [10%]

Find the minimum spanning tree of the following figure using Prim and Kruskal. Also, find the shortest path from I to A and F to C using Djikstra's Algorithm!

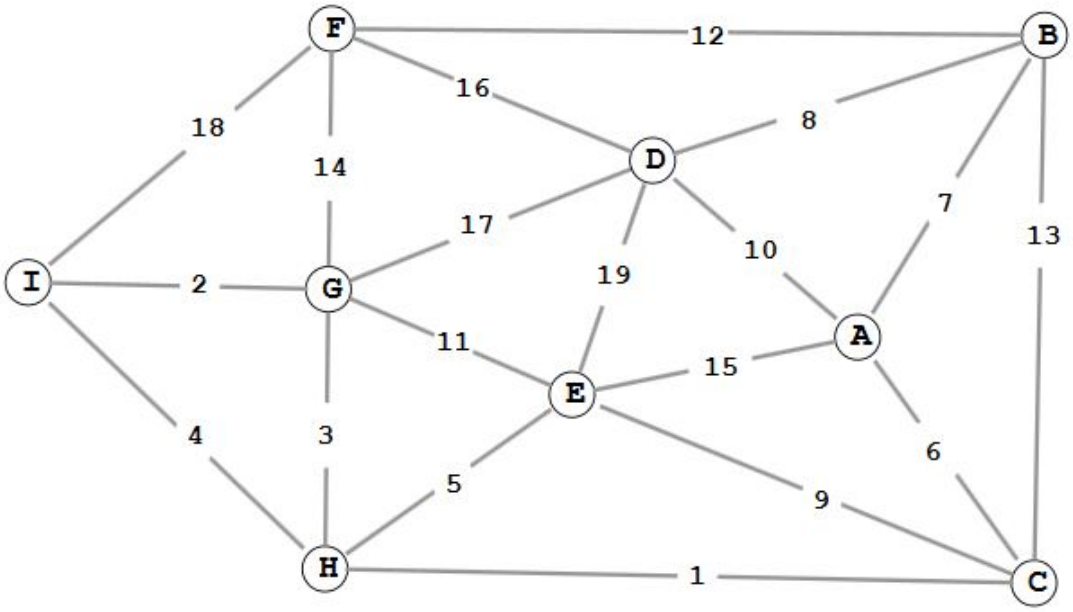
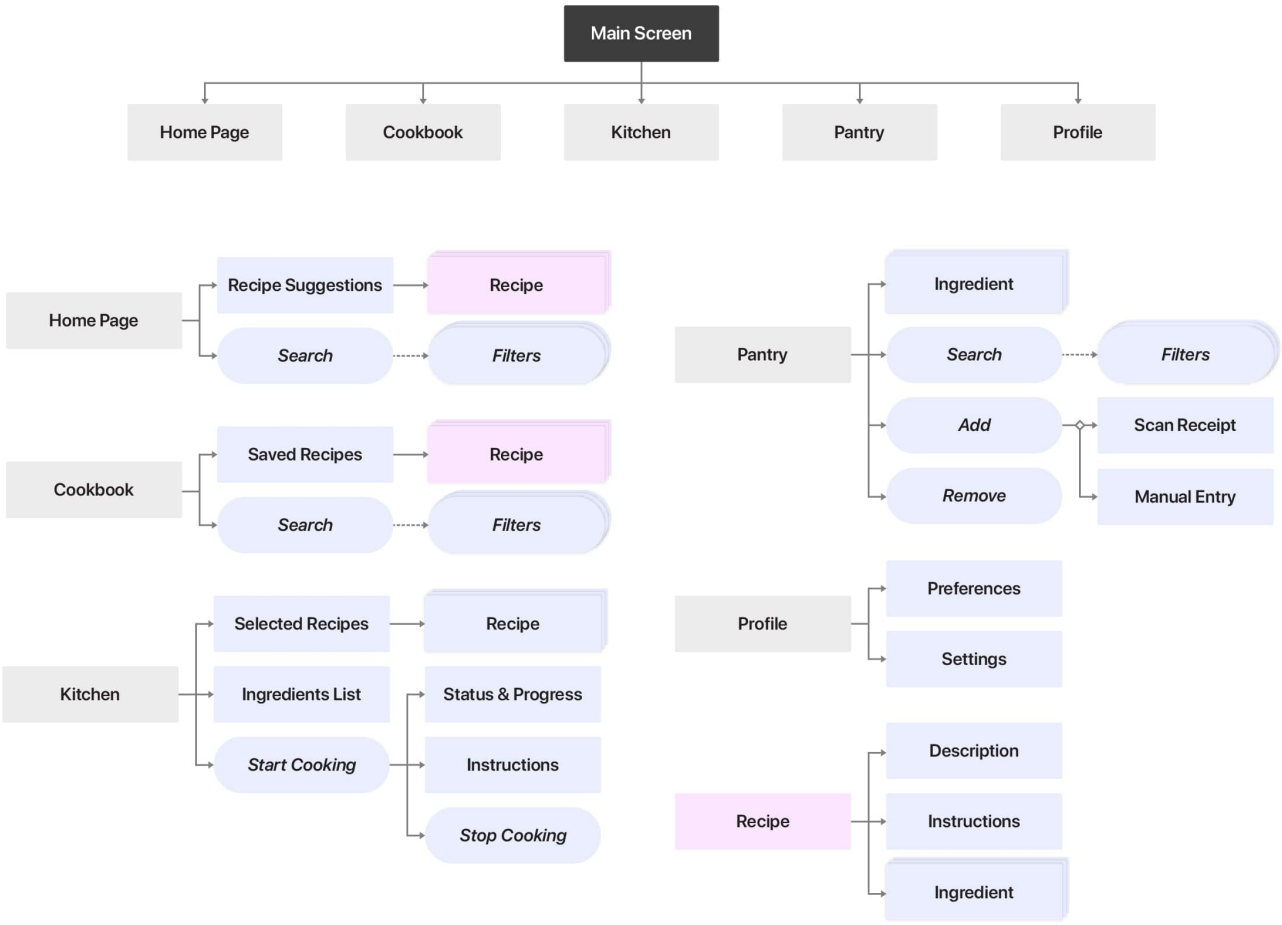


Figure 1

Case: Whisk [50%]

Whisk is an app that helps busy home cooks plan and cook multiple dishes efficiently for their friends and family. It keeps track of your available ingredients, provide relevant recipe suggestions, and simplifies the cooking process by merging all the instructions into a single list of steps.



There are several menus in this program:

- Home Page — A place for users to browse and discover recipes that are relevant to their preferences.
- Cookbook — A library of users' saved recipes which could be manually inputted or saved from online sources.
- Kitchen — A place for users to see the dishes they want to cook, the ingredients they need, and a simplified list of instructions.

- Pantry — A place for users to keep track of their available ingredients. Users can add new ingredients to the pantry by manual entry or by scanning purchasing receipts. It uses your food purchasing habits and average times of food decay to predict the things you have left.

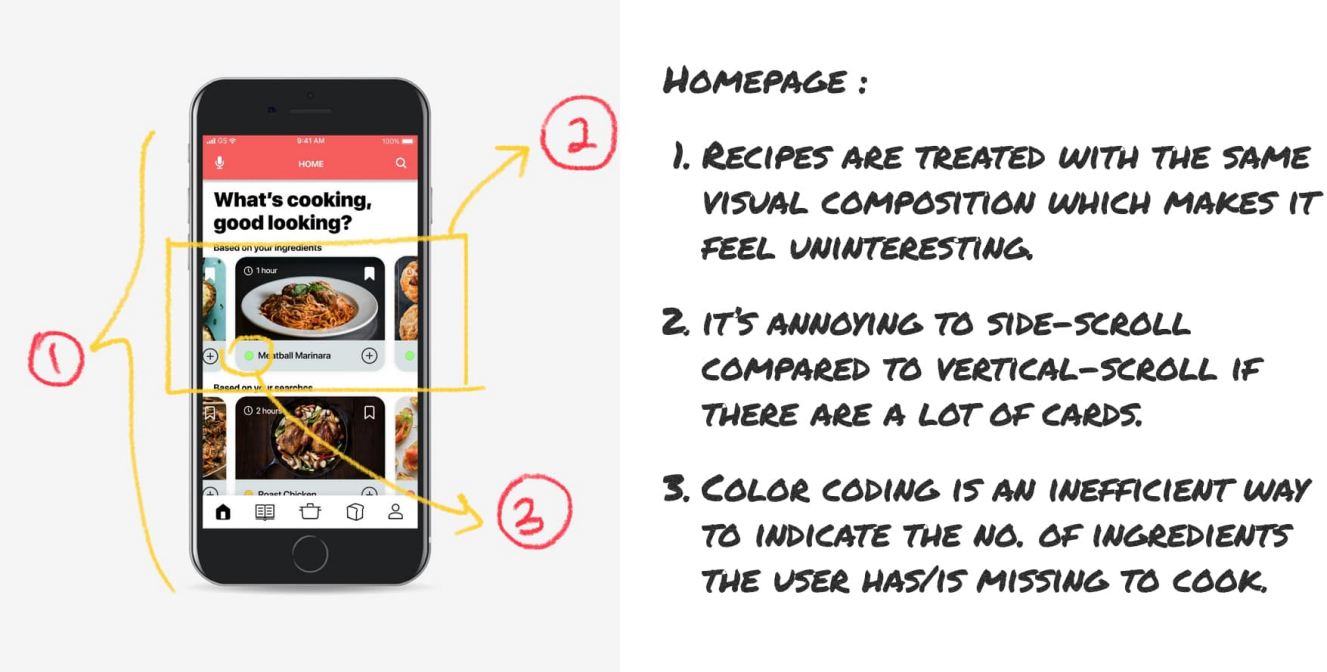
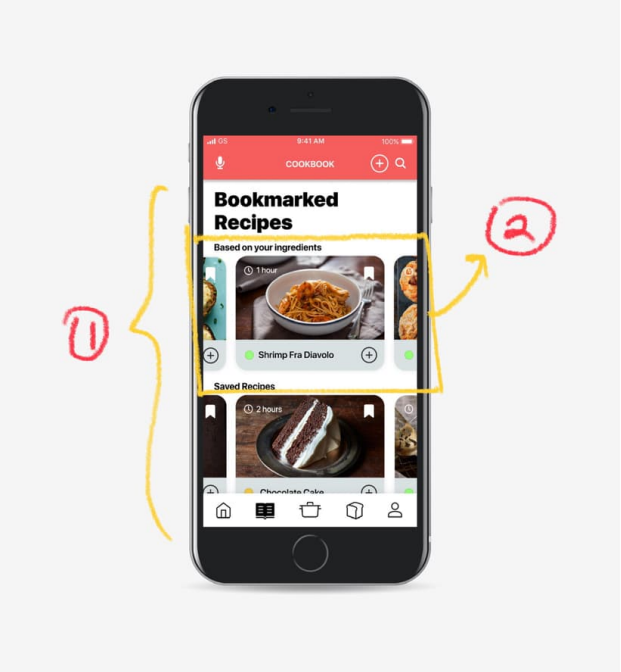


Fig 1. Homepage

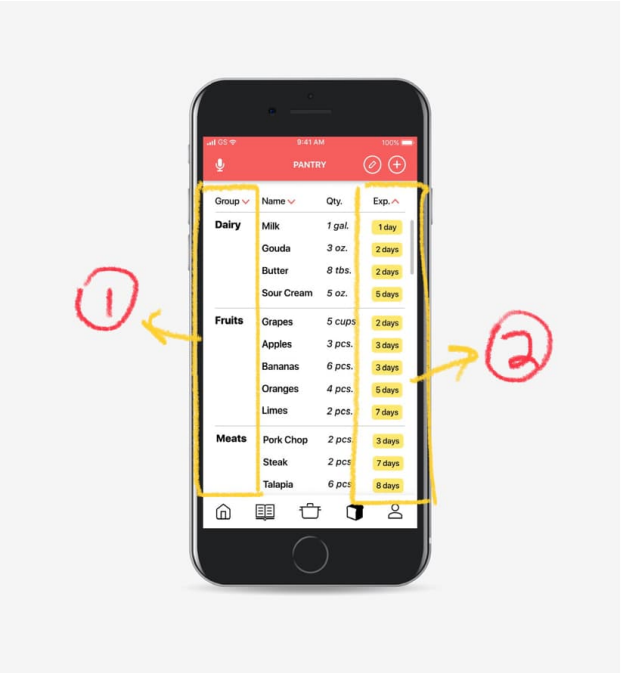
Fig 2. Kitchen



COOKBOOK :

1. LOOKS EXACTLY LIKE THE HOMEPAGE. IT NEEDS TO BE DISTINGUISHABLE SO THAT THE USER CAN BE EASILY AWARE OF WHERE THEY ARE.
2. USERS COME HERE TO BROWSE RECIPES THEY ALREADY KNOW. SO WE SHOULD EMPHASIZE ON THE INFO RATHER THAN THE IMAGE. SIDE-SCROLLING IS ALSO A PROBLEM HERE.

Fig 3. Cookbook

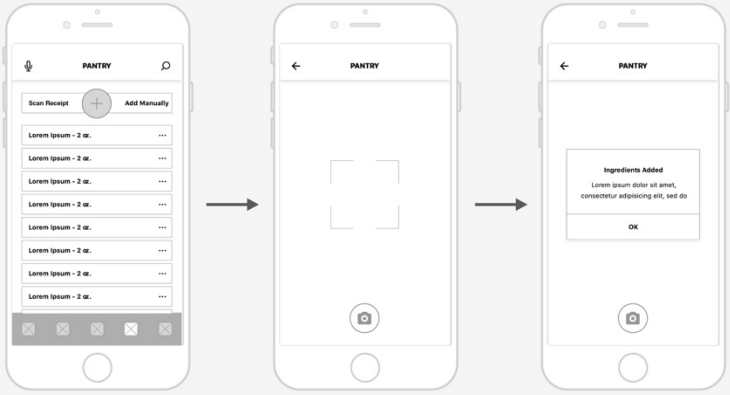


PANTRY :

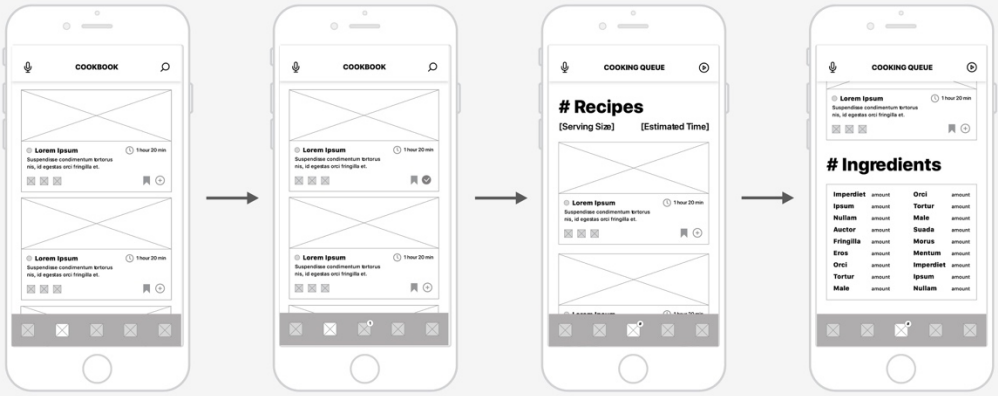
1. THIS CAN SIMPLY BE A FILTER AT THE TOP INSTEAD OF TAKING UP A COLUMN OF VALUABLE SPACE.
2. THE EXPIRATION DATES SHOULD ONLY BE COLOR-CODED WHEN SOMETHING IS EXPIRING SOON INSTEAD OF DIVIDING THE ATTENTION BY COLOR-CODING EVERY DATE.

Fig 4. Pantry

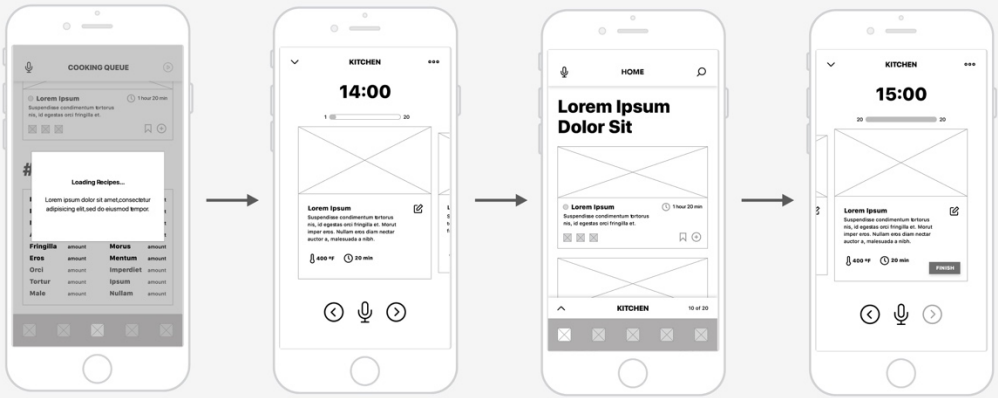
① Adding ingredients with receipt.



② Adding recipes to the cooking queue.



③ Going through the cooking process.



Sources:

https://www.upgrad.com/blog/data-structure-project-ideas-beginners/

https://www.khanglee.com/project/Whisk

Good luck everyone!