Additional Feature:

The livable area of fully vaccinated individuals represents the ratio of fully vaccinated people to the overall population multiply the total livable area within a specific zip code.

The calculation for the average livable area of fully vaccinated individuals is determined by the formula:

Livable area of fully vaccinated people =
total livable area * fully vaccinated population / total population

Here, the fully vaccinated population is derived from the highest available data for that particular zip code. The total population is based on the overall population of the same zip code. The total livable area is the cumulative sum of all valid livable areas within that specific zip code. The result is rounded to four decimal places.

Use of Data Structures:

In the implementation of our program, various data structures such as HashMap, HashSet, LinkedHashMap and ArrayList has enhanced efficiency. Below are key instance of their application:

1. **HashMap for Zip code searches:**
   Scenario: The propertiesCache, populationCache, and covidReportsCache utilize HashMaps.
   Reasoning: This choice allows for O(1) time complexity in searching for entities based on zip codes.
   Comparison with LinkedHashMap: While LinkedHashMap could be an alternative, the lack of necessity for maintaining zip code order makes HashMap a more efficient choice. LinkedHashMap incurs additional overhead by managing a double-linked list for ordering.

2. **ArrayList for reports:**
   Scenario: covidReportsCache and covidReportsByDate HashMaps use ArrayLists to store reports.
   Reasoning: Given the need to store multiple data entries for a single zip code, ArrayLists provide a suitable structure.
   Comparison with HashSet: While HashSet could replace ArrayLists, the traversal requirement for the entire collection makes ArrayLists a more efficient choice in this context.

3. **LinkedHashMap for ordered results:**
   Scenario: The getPartialOrFullVaccinationsPerCapita() method involves converting a HashMap to a LinkedHashMap.
   Reasoning: LinkedHashMap maintains order based on zip codes, aligning with the desired ordering of results.
   Comparison with ArrayList or LinkedList: While alternatives like ArrayList or LinkedList exist, they lack the ability to hold key-value pairs directly, necessitating additional classes for result pairs.
4. **HashSet for unique argument names:**
   Scenario: In the Main class, a HashSet is employed to store valid argument names.
   Reasoning: This ensures each argument name is accessed only once during program execution.
   Comparison with ArrayList: The use of HashSet prevails due to its guarantee of uniqueness and the O(1) efficiency of its remove() and contains() methods, surpassing ArrayList's O(n) complexity.

Lessons Learned:

I have drafted the entire framework of the program, including the implementation of UI, processor, database, and utilities. The content of the Parser module was completed by my teammate, Wu.

Unfortunately, during our collaboration, my teammate felt that he couldn't comprehend the architecture of my program, and we have a bad communication to deal the conflicts.
I caught a severe cold midway through and had to take exams for other courses, leading me to neglect more communication with my teammate.
I believe that the lack of timely communication with my teammate during the implementation of the program framework led to these issues.

Most of the time, we communicate through WeChat.

We utilize GitHub for version control.