This project focuses on handling incoming animal records using File I/O and keeping everything organized. I created the program to read a text file with details about each animal, create objects for them, and sort them by species using intelligent data structures. The main parts of the system include a File Handler to read data, an Animal Class (with subclasses) to store details, a Data Storage system to keep track of categorized animals, a Report Generator to summarize everything, and a simple User Interface to display info. I am using vectors to store animals dynamically, unordered maps for quick lookups, and optionally linked lists or queues if needed. The process involves reading the file, pulling out details, sorting animals by species, generating reports, and cleaning up memory. Various components of the program work together, including file handling, data storage, and report generation, to keep things running smoothly. Error handling considers bad file formats, makes species matching case-insensitive, and prevents memory leaks with smart points. In the future, I could add a graphical interface, better search, and filtering, or even a database to store everything long-term.

I revised the main() function so it would be more concise and cleaner. It directs readAnimalFromFile() to the file path and uses a range-based loop to display animal details and clean up memory. This reduces unnecessary lines and makes the code more accessible while keeping everything functional.

// Main function  
int main() {  
 vector<Animal\*> animals;  
  
 // Read animals from file and process them  
 readAnimalsFromFile("C:/Users/NCG/CLionProjects/zoo/cmake-build-debug/arrivingAnimals.txt", animals);  
  
 // Display information, generate report, and clean up in one go  
 for (auto\* animal : animals) {  
 animal->display();  
 }  
  
 generateReport(animals);  
  
 // Clean up dynamically allocated memory  
 for (auto\* animal : animals) {  
 delete animal;  
 }  
  
 return 0;  
}