Compared to our solution for assignment 1, the code base is now more well-structured, and classes are of higher cohesion and lower coupling. As recommended in our assignment 1 feedback, the Driver has been implemented separately from the views files and acts like a bridge between the models and the view, making all data manipulation operations, thus making use of the MVC design pattern.

We have also made good use of packages to separate our classes, will makes it easier for somebody outside to understand the code. All classes that are closely related are kept in the same package.

For the last assignment we had only one class DependentUserModel which was used both for Child and YoungChild. In this one, we have separated this into two classes; ChildUserModel and YoungChildUserModel in the goal to have a better design and separation of concerns.

We have also introduced an interface class (iNotYoungChild.java) to our solution, to declare the methods that are common in AdultUserModel and ChildUserModel classes.

So in brief, we have 4 models classes for the users; UserModel *(abstract)*, AdultUserModel, ChildUserModel, YoungChildUserModel. To have a better understanding, a class diagram in also attached in the design folder.

At initial start-up, if there is a people.txt file and there's no database that has been created, the program reads both the people.txt and relation.txt files and eliminates all wrong entries and then create instances of the users. Then a database is created, and the users are saved to it. After that, for all start-up following that, the program reads directly from the database.