1. **Personal information**

   Reservation system, Hanjemma Jeong, 791623, Digital systems and Design

2. **General description**

   This is a text-based reservation system for restaurant called “Dingo”. The system asks customers to put commands that they can use to check the tables situation, make reservations and cancel them. This is aimed at completing ”easy” level in the project description.

3. **Instructions for the user**

   The program is very easy to launch, just run program.py in the project folder. More instructions are in readme.md.

4. **External libraries**

   The app doesn’t use any external libraries, there are only native python libraries like datetime. It doesn’t need any downloads to run, just to launch program.py for the main loop.

5. **Structure of the program**

   The program has a few parts. Firstly, there is program.py which has the main program of the project, which is an infinite loop that asks user questions and they make commands until they want to finish using the reservation system. The data are represented as classes (restaurant, table, reservation) with their own attributes. The main program loads the restaurant from file, or creates new ne if it doesn’t exist. After that, you can do different things with the reservation system and just save when you finish. The restaurant has 8 tables and is open every day from 12:00 to 20:00.

6. **Algorithms**

   There is some algorithms used by the project. Firstly, there is the infinite loop which runs functions and only dies after it is told to by the user. There is also a loop where time is added to a date every hour and then some things are checked based on the current hour. Displaying the tables and checking reservations is done like this. Essentially this is just the ”brute force” algorithm, used for finding reservations and time windows.

7. **Data structures**

   The only major data structures are the classes that represent restaurant, table and reservation. In addition to this, there are only normal ones like the List (the restaurant has a list of tables, and one table has list of reservations).

8. **Files**

   The project has files for the unique class objects, and also one database type file for storing the restaurant at the end of interaction (restaurant_file). The home of the program is in program.py, which uses the other files to allow usage of the reservation system.

9. **Testing**

   The program was only tested using manual inputs. There are no unit tests as it wasn’t part of requirements. However, manual inputs worked fine since every time I input reservation it is saved to the file and didn’t need to be entered again.

10. **The known shortcomings and flaws in the program**

    Known shortcomings are lack of graphical UI, as well as simple functionalities. Also you can only enter one unique string when making reservations, there is no class for entirely new user (there might be some registration system etc), as that would make everything much more complicated. As far as I know there is not any major bug. One thing is that there is no block for making reservations into time that has already passed.

11. **3 best and 3 worst areas**

    3 best areas are:

    1. Get_params_from_user(), which allows reusing the same input checks. This makes it so when there is repetitive questions and try/except, all those places use the same function.
    2. The table display, which shows clearly situation of every table and hour.
    3. App working smoothly, responds nicely to bad user inputs

    3 worst areas are:

    1 The input is a bit stiff, you have to enter a lot of text every time

    2. There is no check for very old dates, now you can reserve historical time even.

    3. little amount of options

12. **Changes to the original plan**

    Some of the things in project plan were a bit impossible to make with just text interface (table location). The plan was streamlined to allow displaying things a bit better, and to do the operations in the requirements, and table location was dropped.

13. **Realized order and scheduled**

    The project was mostly planned and options researched in the first 4 weeks before tutori meeting, and finished mostly in 6 weeks before deadline.

14. **Assessment of the final result**

    I think the project came out really well, and fills all the requirements. I am happy with how smooth the loop works, and it is not easy to mke it crash. The reservations table looks nice, and there are some filters to make sure bad dates cannot come through. Also, the cancellation method was not part of requirements but is still implemented.

15. **References**

    I mostly used online sources to complete project, and also little bit the course module information, and also things that I already knew and just googled further how to use them.

16. **Attachments**

    Demonstration video is in project folder. (demonstration.mp4)