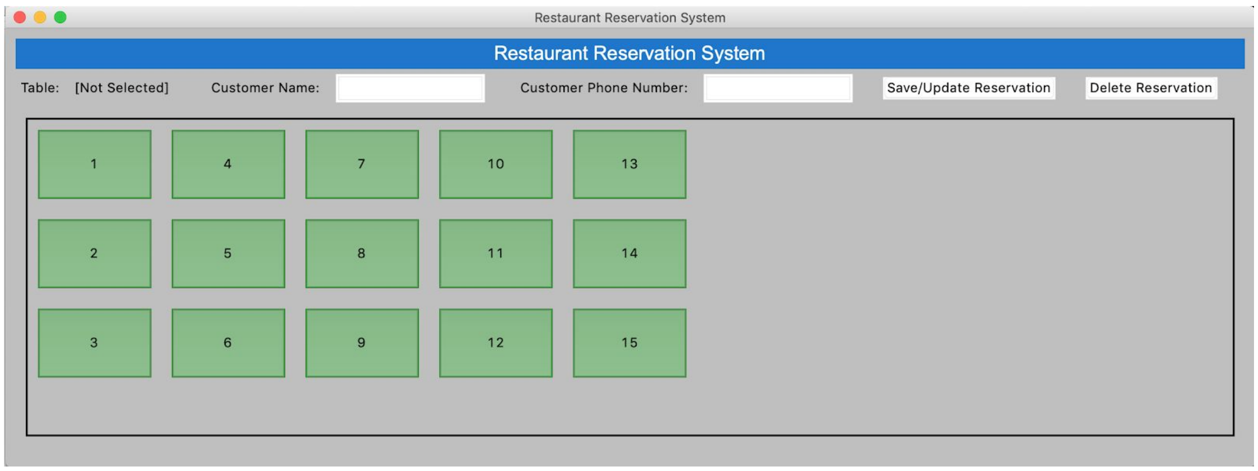
ENGR 102 – Programming Practice
Mini Project 1
Spring 2020

Due on March 12 by 17:00

In order to help reduce the number of open tables and maximize table turnover, restaurants usually take reservations. In this mini project, you are going to develop a software tool to help restaurants manage their reservations in an efficient way.

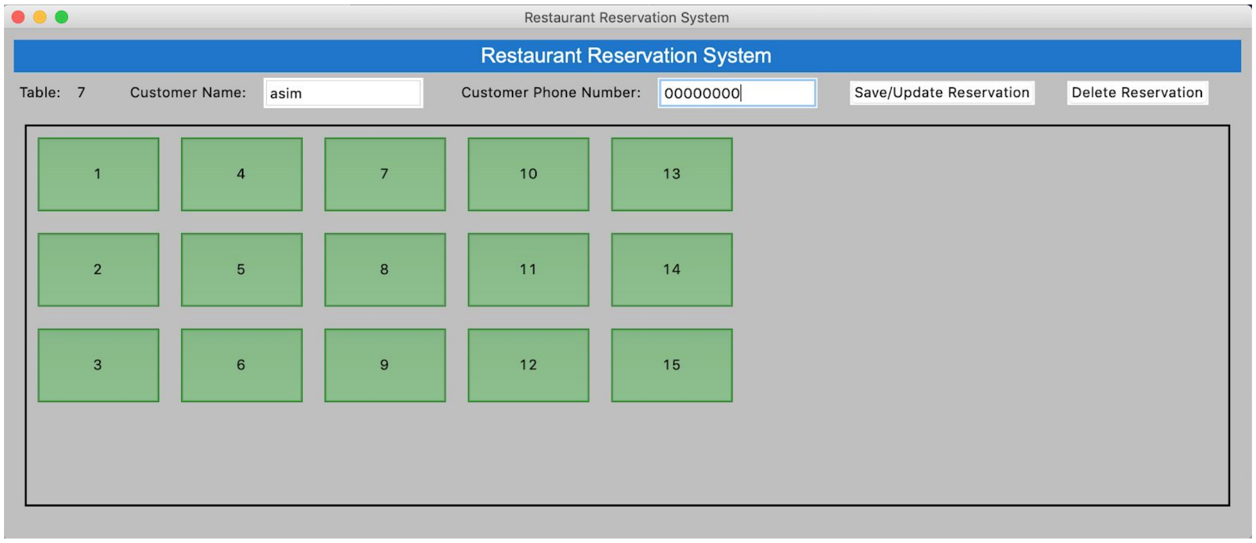
How it should look like: Your graphical user interface should look like the following one.



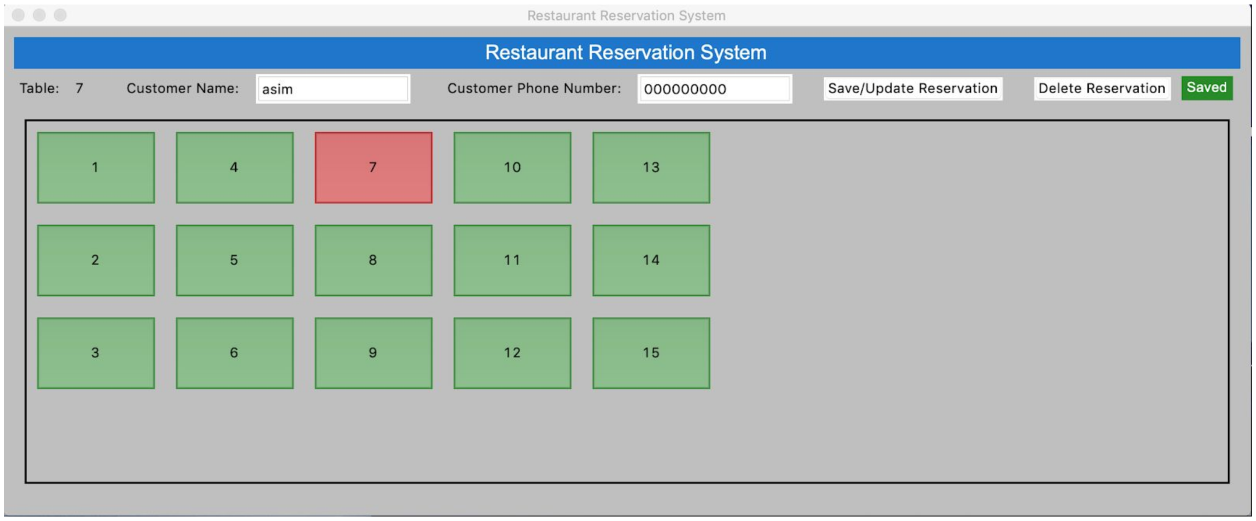
How it should work:

1) When the app is opened, the GUI should look like the figure above. Please assume that each restaurant has 15 tables. **Initially no table is selected**.

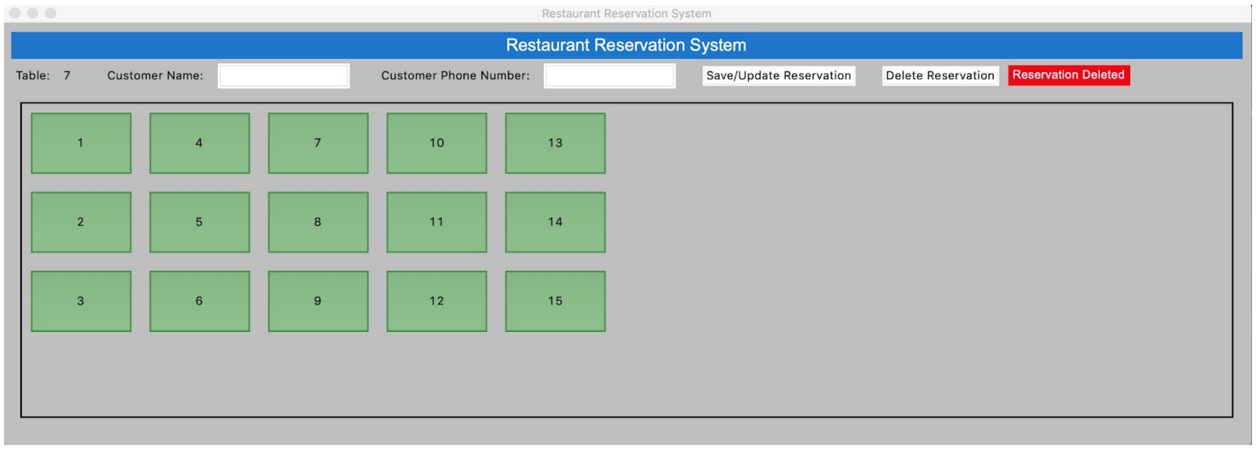
2) When a table is selected, the **Table** label at the **top left** of the GUI is changed as shown in the following screenshot. Then, the information for the selected table **(customer name & customer phone number)** can be entered **but not saved yet.**


Restaurant Reservation System
Restaurant Reservation System
Table: 7
Customer Name: asim
Customer Phone Number: 00000000
Save/Update Reservation
Delete Reservation
1
4
7
10
13
2
5
8
11
14
3
6
9
12
15

3) When the **(save/update Reservation)** button is **clicked**, the reserved table's color should be changed to **red** and a label (with test "**Saved"**) should be added, next to the Delete Reservation button, to confirm the completion of the process (This label should disappear after 2 seconds - you can use the method **'after'** that was used in week 3 practice session). Besides, the data should be **saved to a database**. **Note**: A table shoud not be reserved if the name and phone number is empty. In that case, instead of "Saved", "Incomplete Info" text should be displayed next to the "Delete Reservation" button for 2 seconds. Existing reservations may be updated through the Save/Update button as well (e.g., correcting the phone number of a customer who had reserved a table earlier).



4) A reservation could be **deleted** as well. When the button **(Delete Reservation)** is **clicked**, the table's color should be reset to **green** and a label(**Reservation deleted**) should be added, next to the Delete Reservation button, to confirm the completion of the process (This label should disappear after 2 seconds - you can use the method **'after'** that was used in week 3 practice session). Also, the corresponding reservation should be **deleted from the database** and the customer name and customer phone number fields should be cleared. **Note**: A reservation should not be deleted if it does not exist!



5) Later, when the app is closed and opened, the program should read the database content at the beginning and accordingly color the tables reserved earlier should be **red** and when **clicked** the information about that table should be displayed in the fields **(customer name & customer phone number)**. See the following screenshot. **Note**: suppose we did not delete table 7 in the previous step to understand this.

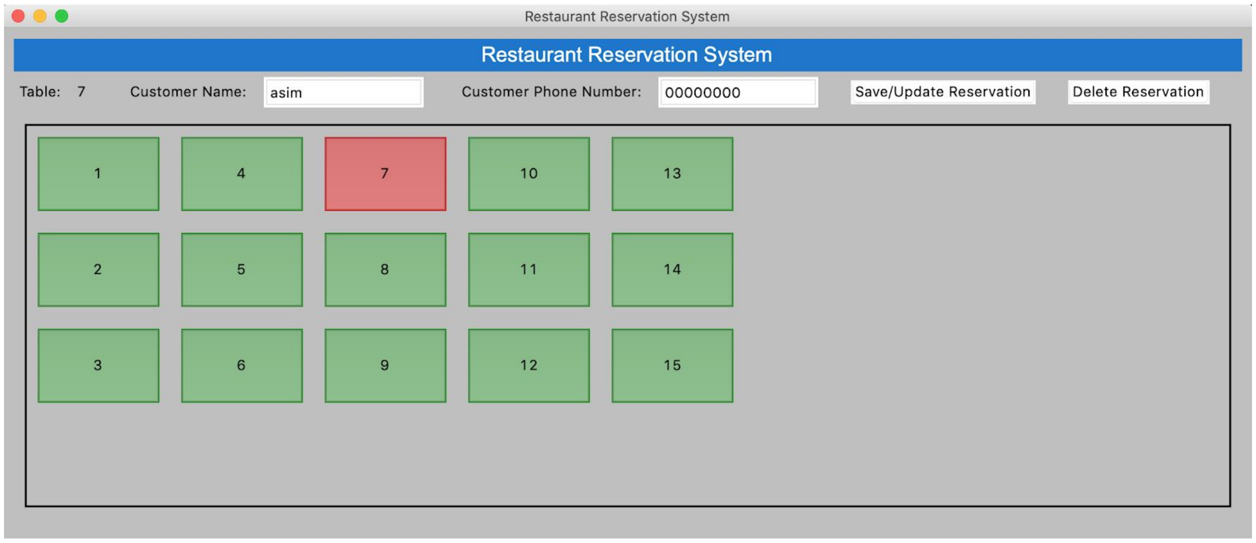
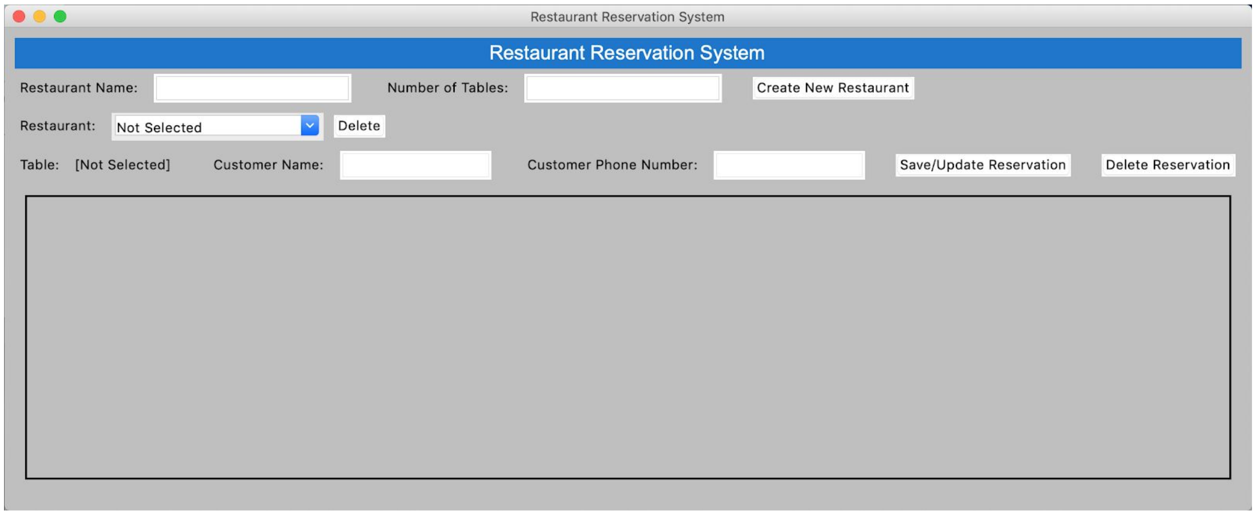


Table: 7 Customer Name: asim Customer Phone Number: 00000000

Bonus Part (50 points extra):

The extended bonus version will add the capability to manage multiple restaurants. Please read on for more details.

1) Initially your GUI should look like the following. **Note**: the shown combobox says (Not selected).



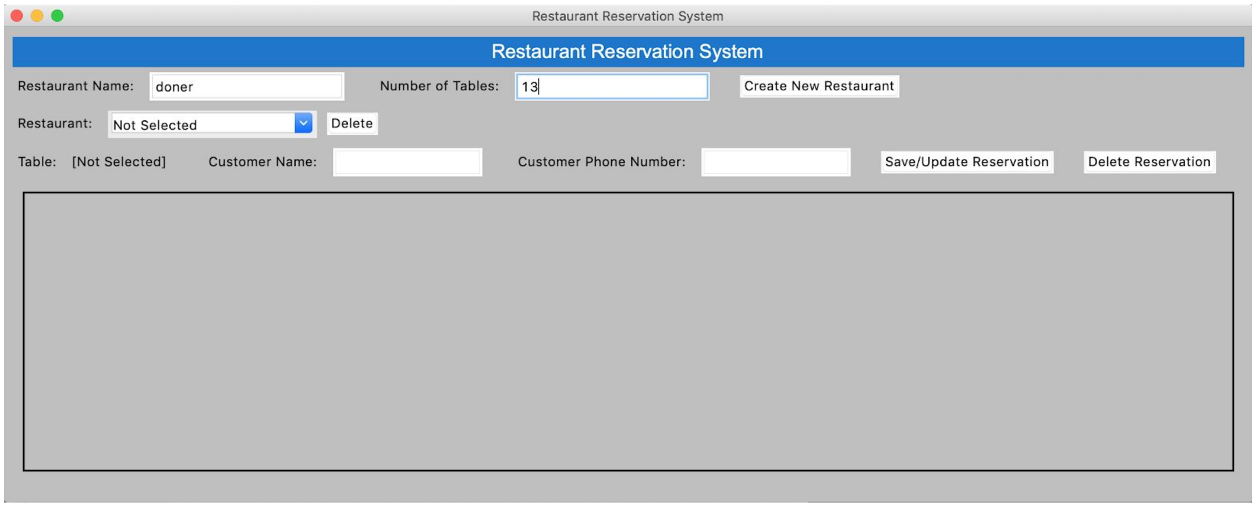
Restaurant Name: Number of Tables:

Restaurant: Not Selected

Table: [Not Selected] Customer Name: Customer Phone Number:

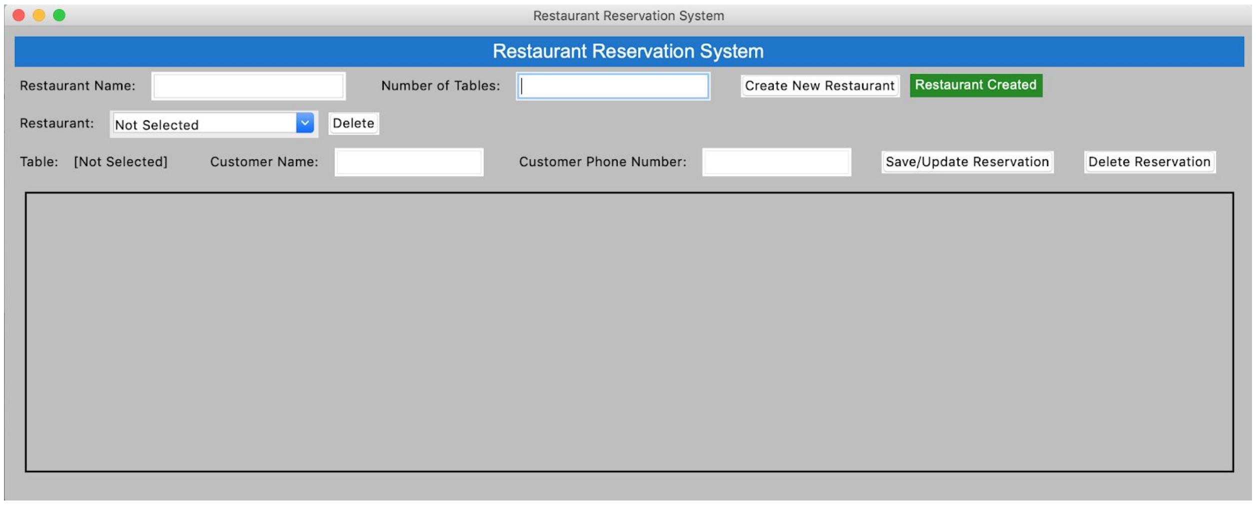
2) At the first run, there will not be any restaurants or tables. A restaurant can be created after filling the required fields: **Restaurant Name and Number of Tables**. When the button **Create New Restaurant** is clicked, a restaurant should be created and added to the

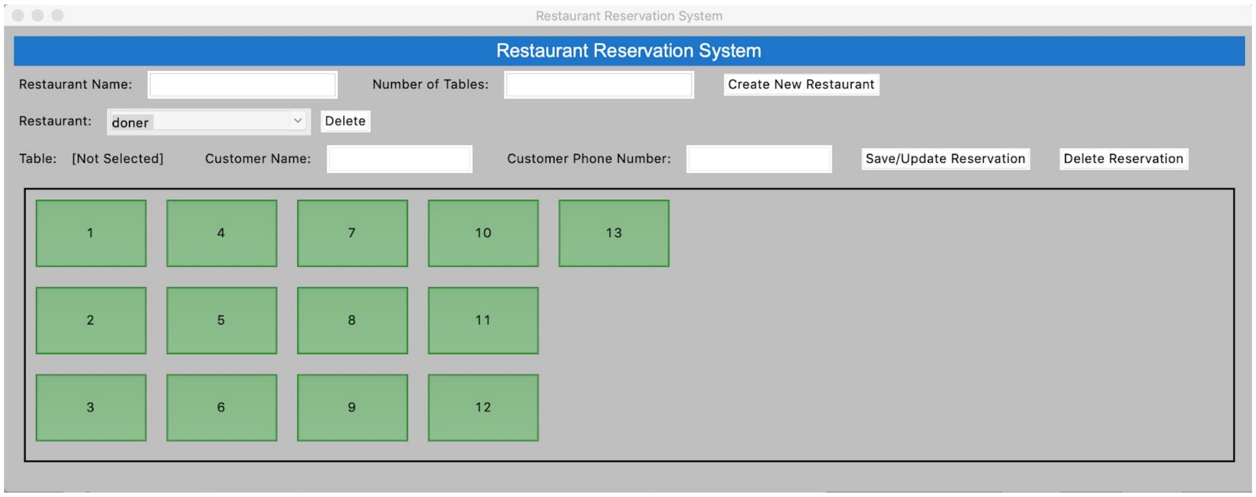
combobox and when selected, its tables should be displayed. **See the following screenshots**. Regardless of the number of tables, there will always be three rows of tables. The number of columns will change depending on the number of tables in a restaurant. The minimum allowed number of tables is 6 and the maximum allowed number of tables should be 24. Your program should check these limits during restaurant creation and warn the user if needed with a message "Num. of tables may be between 6 and 24" next to "Create New Restaurant" button for 2 seconds.



Restaurant Name: doner

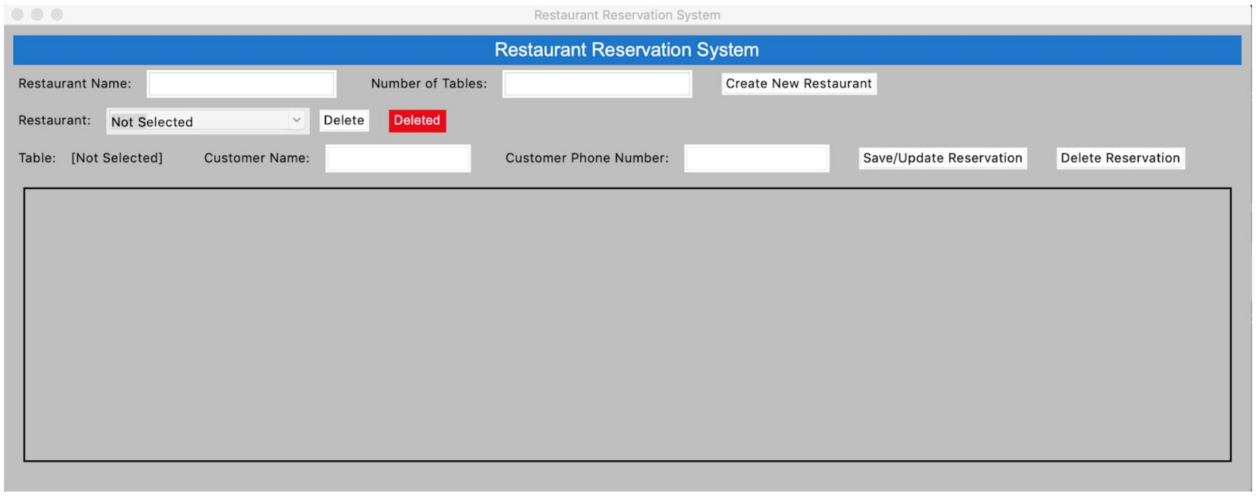
Number of Tables: 13





Restaurant: doner

3) Besides, a restaurant could be deleted when selected from the combobox. And, a label (**deleted**) should be displayed next to the button delete to confirm the completion of the process.



Implementation Notes

- **You need to use at least 3 classes in this project. For instance, you may create classes to represent the following: Restaurant, Table, GUI,..etc**
- For an example of Combobox usage, you may take a look at the following link:

  https://www.delftstack.com/tutorial/tkinter-tutorial/tkinter-combobox/

- You may represent each table with a Button widget.
- In order to set the time between coloring updates, you may use - **after()**- method. Please see the following link for an example: https://www.geeksforgeeks.org/destroy-method-in-tkinter-python/
- You may use events for table reservation handling (please wait for Week 5 lecture next week!). Alternatively, you may use the regular command option that we covered. The second option will require writing more and repetitive code. You may use any of these options.
- Apart from the above, the lecture slides are enough for most parts, still feel free to research and implement. Please provide references if you use outer sources. TutorialsPoint, StackOverflow, and Effbot might be good starting points, but **do not copy codes directly**.

Warnings:

- You **CANNOT** use the **place** geometry manager, only **grid** and **pack** are allowed.
- Do not talk to your classmates on project topics when you are implementing your projects. Do not show or email your code to others. If you need help, talk to your TAs or the professor, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have serious consequences for both parties.

- Carefully read the project document, and pay special attention to sentences that involve "should", "should not", "do not", and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resources at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so may result in plagiarism investigation. Last but not the least, you need to understand code pieces that you may get some other resources. This is one of the goals of the mini projects.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is important to understand all the details in your submitted code.

Submitting the project

- Projects may be done individually or as a small group of two students (doing it individually is **strongly** recommended for best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for file naming details).
- Submit your own code in a single Python file. Name it with your and your partner's first and last names. As an example, if your team members are Deniz Barış and Ahmet Çalışkan, then name your code file as deniz_baris_ahmet_caliskan.py (Do not use any Turkish characters in file name). If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.
- Those who do not follow the above naming conventions will **get** 10% **off** of their project grade.

Late Submission Policy:

- **5%:** Submissions between 17:01 – 18:00 on the due date
- **10%**: Submissions between 18:01 – midnight (00:00) on the due date
- **30%**: Submissions after which are up-to 24 hours late.
- **50%**: Submissions which are up-to 48 hours late.
- Submission more than 48 hours late **will not be accepted.**

Grading Criteria

- GUI design: (**50**)
- Proper **reservation** of a table(changing table's color, saving the information to a database - when app is reopened reserved tables should still be reserved!): (**20**)
- Proper **updating** of a reservation(changing table's color and saving the information to a database): (**15**)
- Proper **deletion** of a reservation(changing table's color and updating the information): (**15**)

From your overall grade, we will deduct points by the specified percentage for the following items:

- **Inappropriate/Cryptic variable names (-10%)**
- **Classes and objects are not used properly (-30%)**
- **Insufficient commenting (-10%).**

Have further questions?:

- If you need help with anything, please use the office hours of your TAs and the instructor to get help. **Do not walk in randomly (especially on the last day) into your TAs' or the instructor's offices. Make an appointment first. This is important. Your TAs have other responsibilities. Please respect their personal schedules.**