OOAD PROJECT DOCUMENTATION

ANALYSIS DESIGN AND IMPLEMENTATION OF DINER DASH GAME CLONE

GROUP 9

- TANGANG MC BRIDE TABUFOR(ICTU20201697)
- Nyeango Ashley Majoma(ICTU 202011034 – asangbeng.brandon@ictuniversity.edu.cm)
- Kiawi Bemsimbom Fien Titi(ICTU20201250–stephane.pele@ictuniversity.edu.cm)
- MOPOUONG WAFO ROBERTO(ICTU20201168–kumo.pascaline@ictuniversity.edu.cm)
- KISSOK MELVIN JEFFERSON (ICTU20201139–Kissok.melvin@ictuniversity.edu.cm)

Course Instructor: Eng. Daniel Moune

Table des matières

INTRODUCTION

Diner Dash is a benchmark task for training reinforcement learning (RL) and imitation learning (IL) polices in the domain of high dimensional action space.

The player is running a restaurant by controlling a waitress to serve customers as many as possible. As shown in the picture, the restaurant has 6 tables with different sizes and up to 7 waiting groups, on the left side and with different sizes, to be served. For each group of people, the player needs to allocate a table for them, collect orders, submit orders, pick up food, serve food, collect bills, clean table and finally return the dish to the dish collection point. There is a happiness value of each group of people, represented in the form of hearts, and the happiness value will decrease if they wait too long. Once the happiness value reaches zero, the customer runs away, and the player loses one star. There is a maximum of 5 stars of each player, and the game ends when the player loses five groups of customers.

Diner Dash is a challenging task, with high dimensional action space, high dimensional state space, infinite horizon, hierarchical structure and requires sub-tasks to be completed in parallel. Such a tough task gives a better training environment which is closer to the real-world problems.

SYSTEMS DESIGN

For the purpose of the design of our system, we will be using UML diagrams. UML stands for Uniform Modeling Language. It is a general-purpose visual modeling language that helps to visualize, construct and

document software systems

USE CASES

Use case 01: Receive customers at the entrance

Actors: player, customer

Description: To move customer from the queue to table

Precondition: Customers must be available on queue

There must be a free table

Post Condition: -Customers are seated on table

-A table becomes occupied

-Customers pass their order

-

Main scenario:

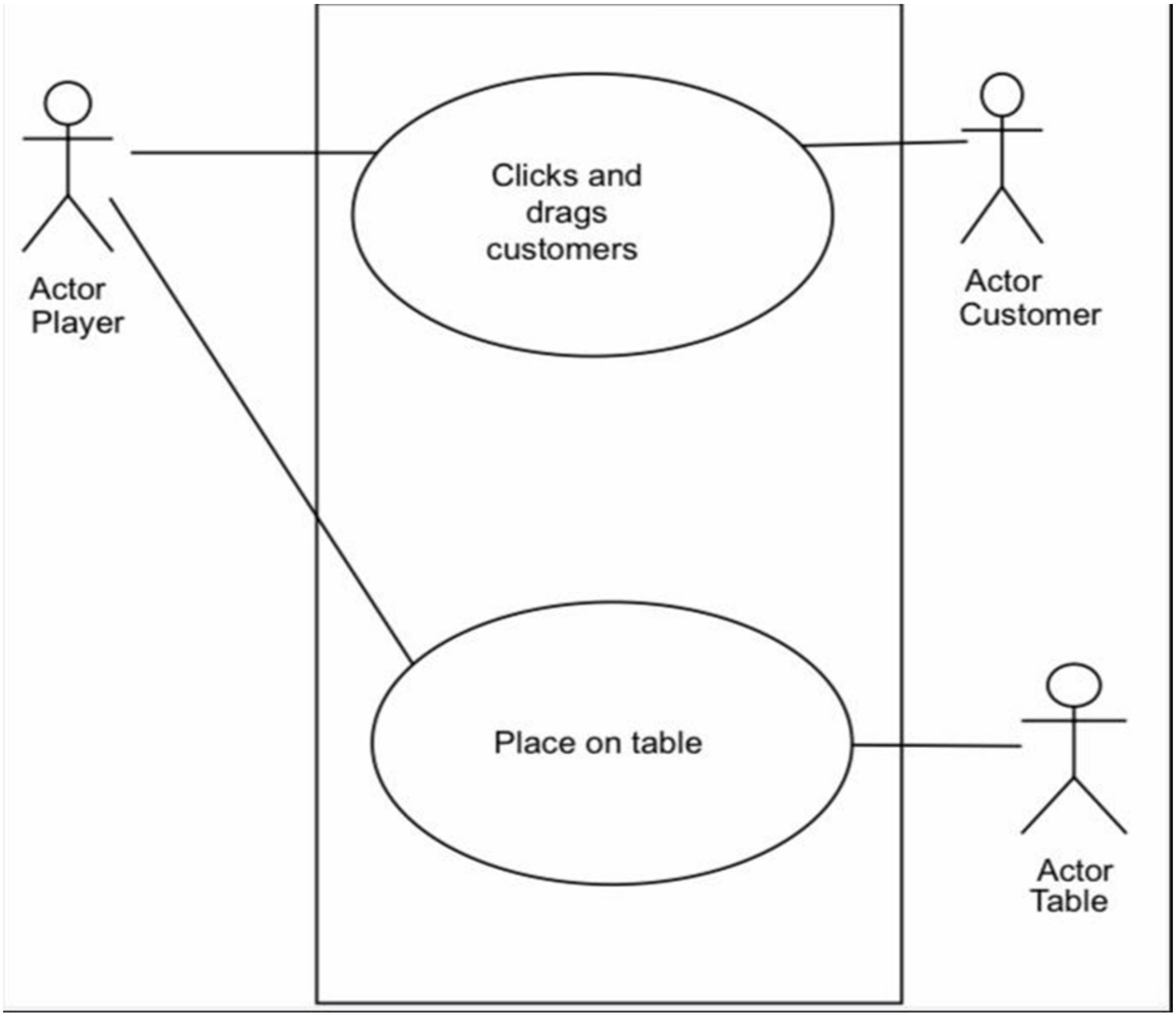
-Game starts

-Customer arrives at the restuarant

-Player drags customer from the queue to table

Alternate Scenario:

If customers don’t come to the restaurant, player wont be able to drag them to the table



Use case 02: Collect customer’s order

Actors: player, waitress

Description: Waitress collects customer’s order

Precondition: Customers must be available on queue

There must be a free table

Post Condition: -Customers must have been seated on a table

-Customers must have been ready to order

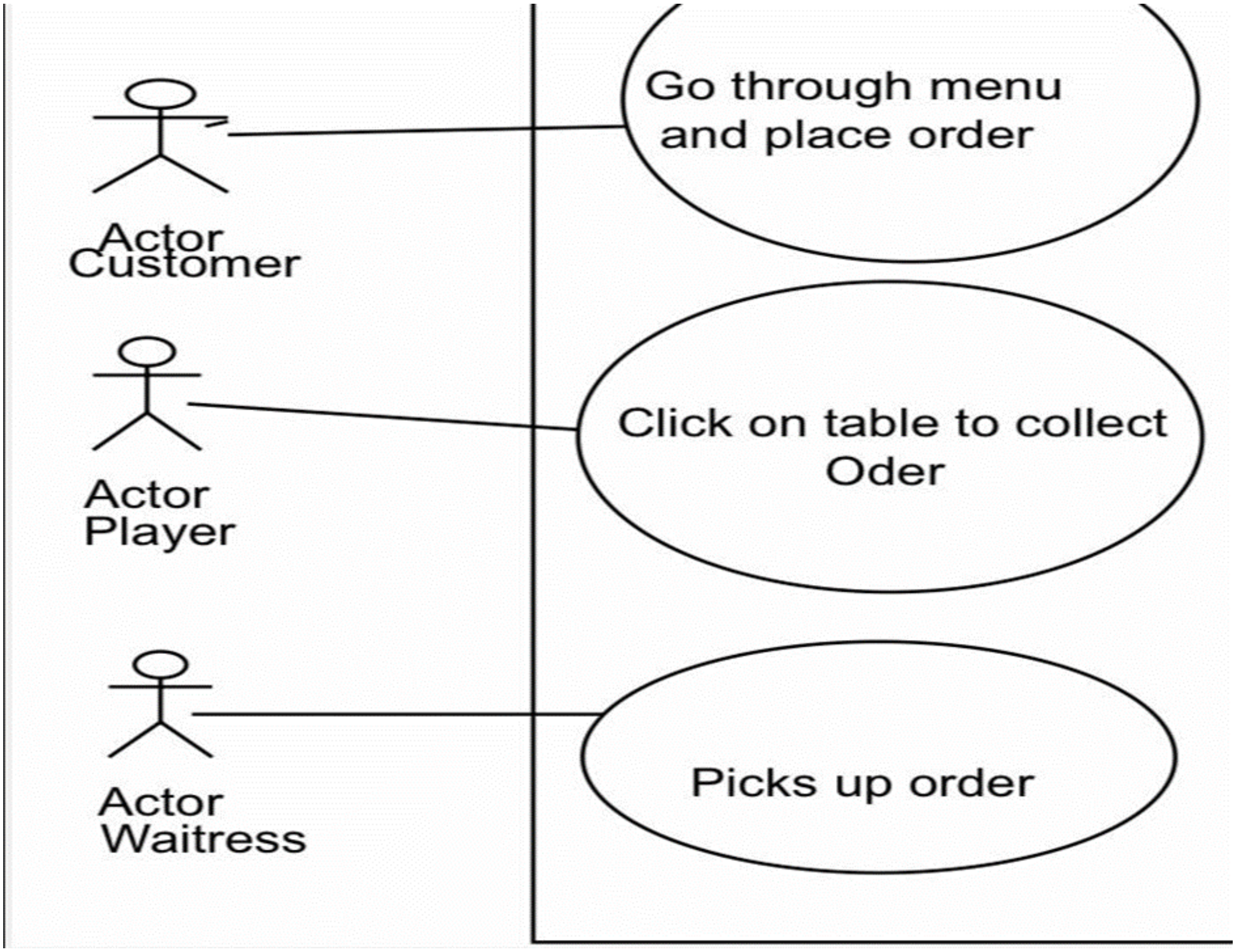
Main scenario:

-Customer go through menu

-Customer makes order

Alternate Scenario:

If customers doesn’t select a meal from the menu, waitress will not be able to serve the meal


Actor
Customer
Actor
Player
Actor
Waitress
Go through menu and place order
Click on table to collect Oder
Picks up order

Use case 03: Place customer's order on the chef table

Actors: waitress, cheftable, player

Description: customer's order is placed on chef table

Precondition: Player should have clicked on order

Waitress should have picked up order

Chef table must be clicked

Post Condition: -Food is prepared by chef

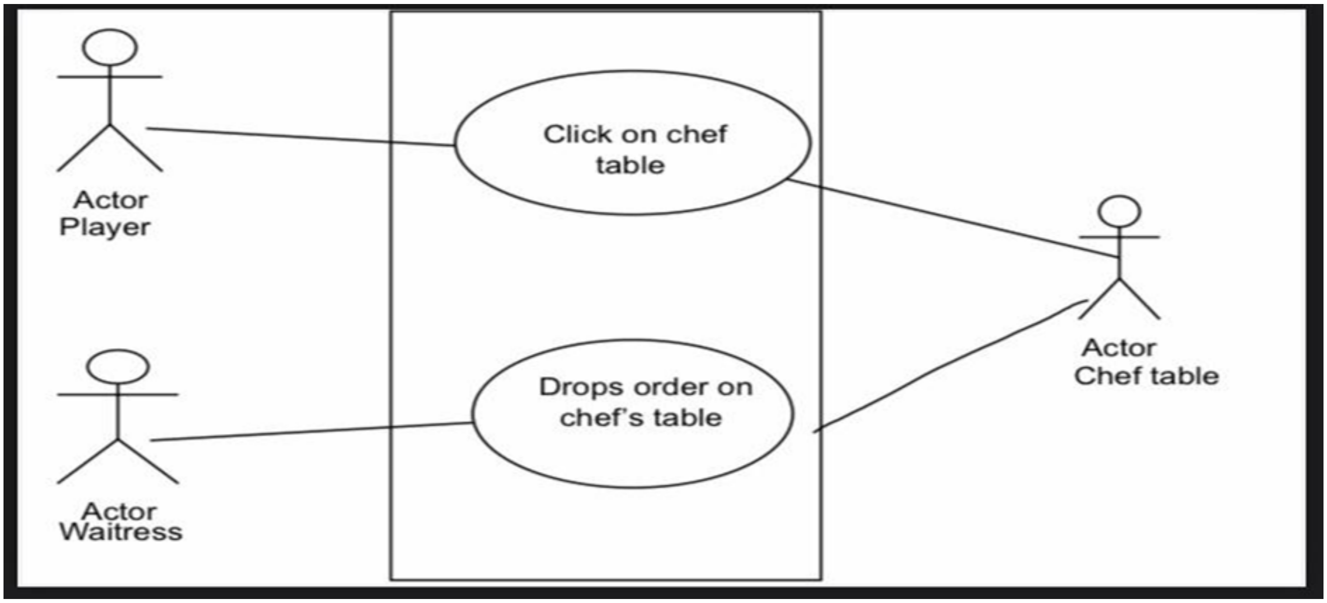
Main scenario:

-Player clicks on chef table

-Waitress places order on cheftable

Alternate Scenario:

If customer's order is not placed, chef won't prepare the meal



Use case 04: Serve meal to customers

Actors: player, waitress

Description: Waitress serves meals to customer

Precondition: meal must be on chef table

Table should be clicked for meals to be served

Post Condition: -The customers are eating the meal

Main scenario: -Chef prepares the meal and place on chef table

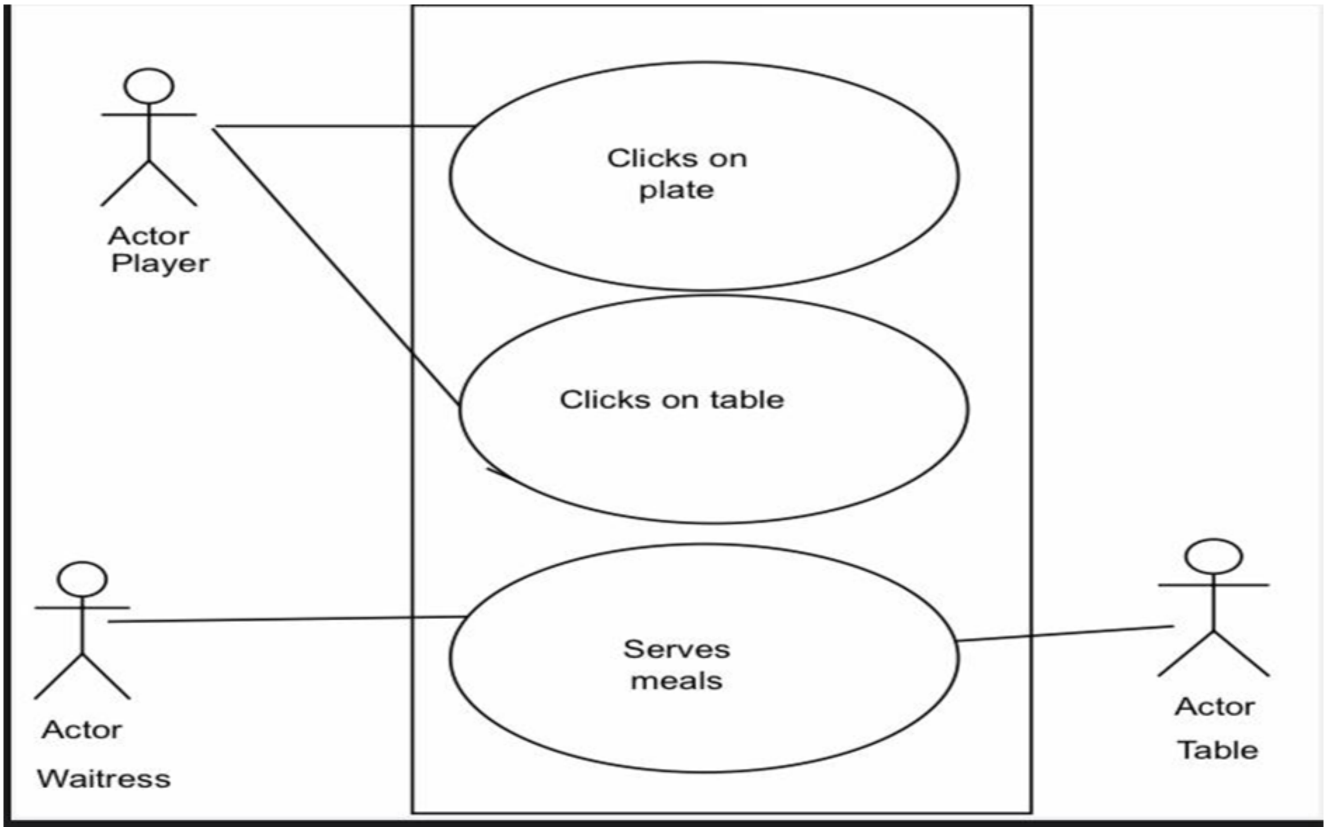
-Player clicks on the chef table for waitress

-Waitress collects meals and serve the customers

-

Alternate Scenario:

If meal is not prepared by chef, customer won’t be served



Use case 05: Gives cheque to customers

Actors: player, waitress

Description: Waitress gives cheque to customer after they have eaten

Precondition: Customers must have finished eaten

Player clicks on the table

Post Condition: -Customers give a tip depending on how happy they are

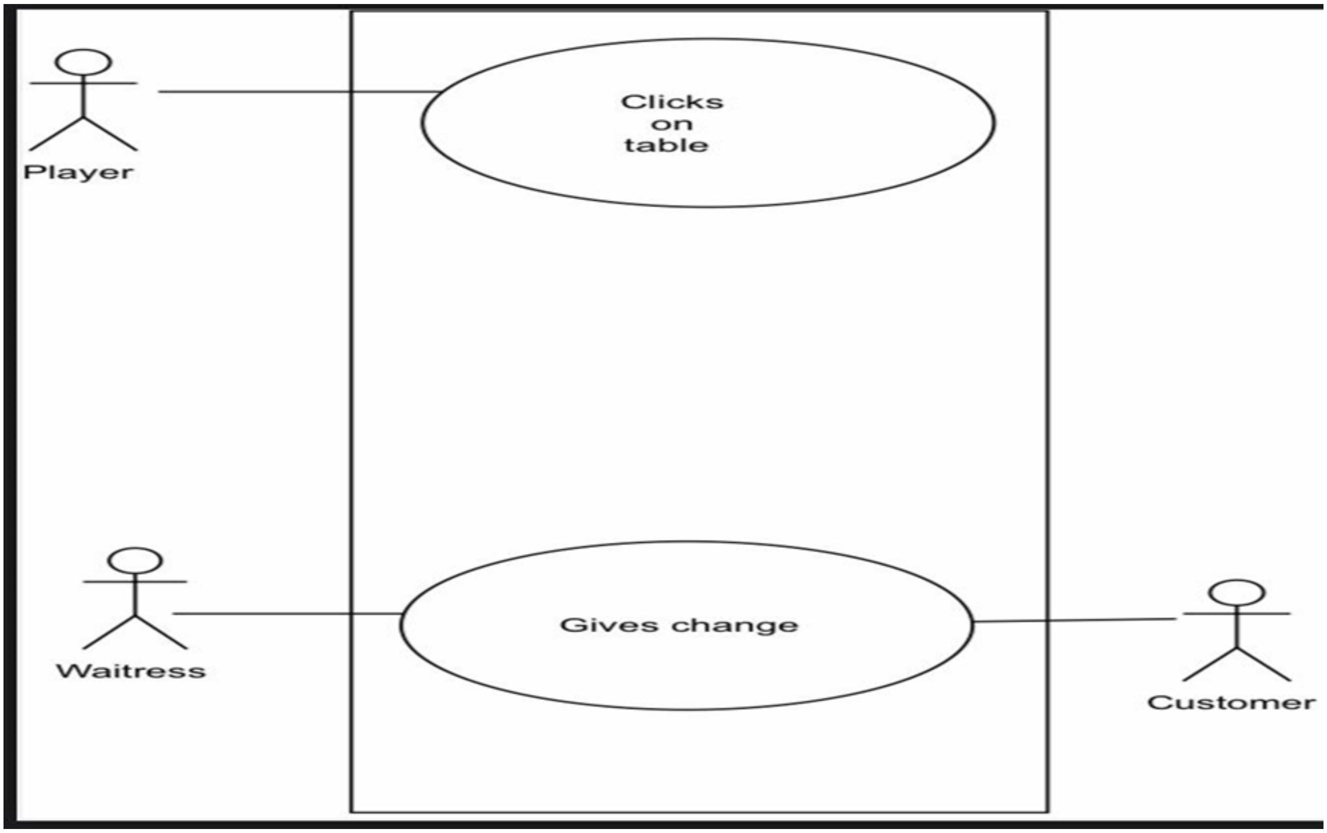
Main scenario:

-Player clicks on table after customer are finished eating

-Waitress places the cheque on the table

Alternate Scenario:

If customers don’t eat, waitress will not be able to give cheque and collect money



Use case 06: Clear up table

Actors: player, waitress

Description: Waitress takes dishes away from the table and drop at the bus station

Precondition: Customers must have left the table

Post Condition: -Table is clean and empty

-Customer cloth’s colour and happiness level can still be seen on the table and chair

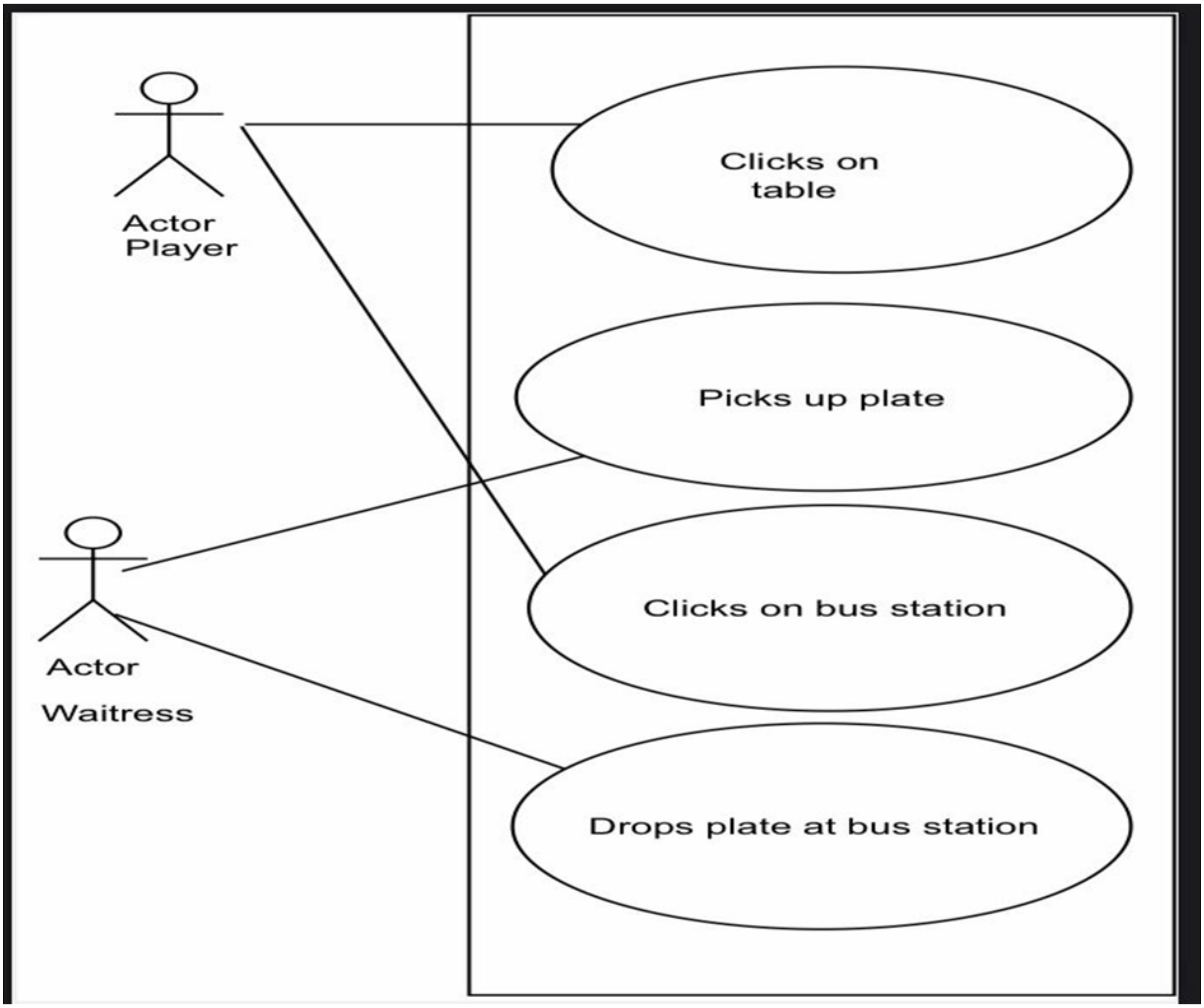
Main scenario:

-Player clicks on table for waitress to pick up the dishes

- Player clicks on bus station for waitress to drop dishes in bus station

Alternate Scenario:

If customers don’t leave the table, waitress will not be able to clean it up



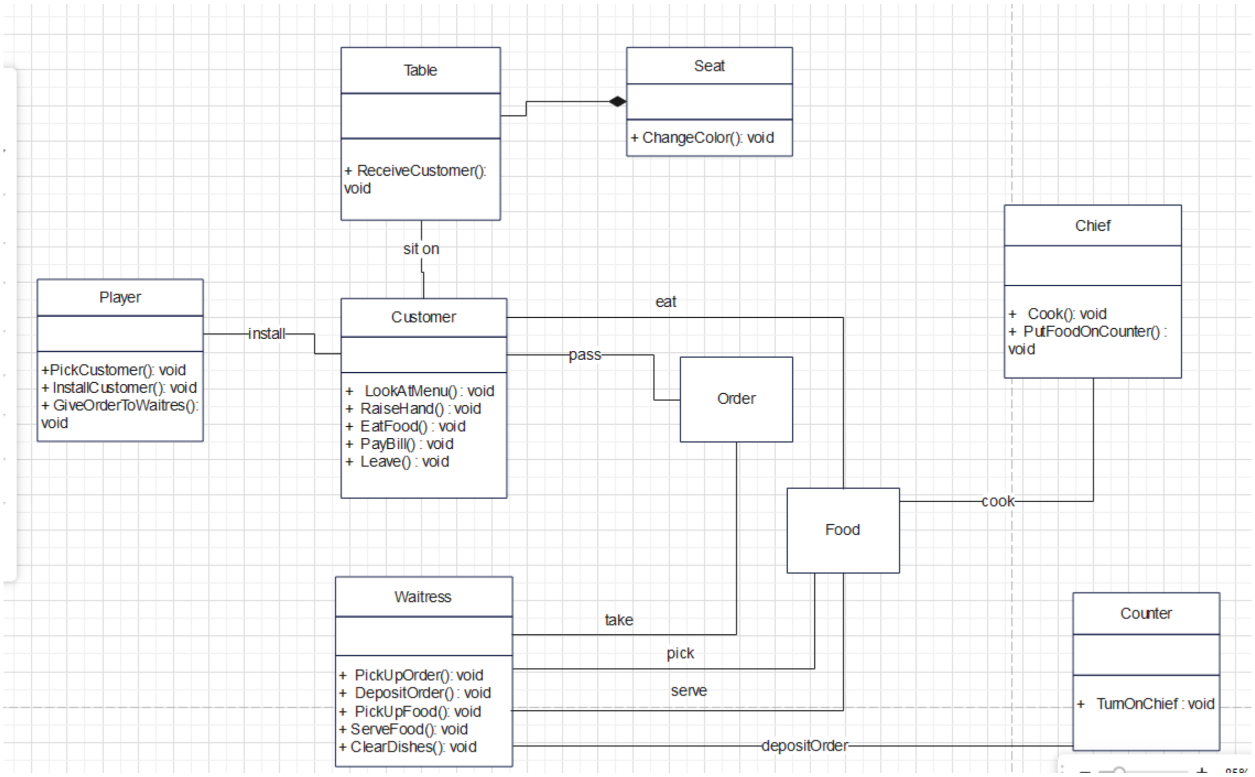
CLASS DIAGRAM

ACTORS :

- PLAYER
- CUSTOMER
- WAITRESS
- TABLE
- SEAT
- ORDER
- COUNTER
- CHIEF

ASSOCIATIONS :

- Install (player, customer)
- Pass (customer, order)
- Eat (customer, order)
- Take (waitress, order)
- Pick (waitress, food)
- Cook (chief, food)
- depositOrder (waitress, counter)



STATE DIAGRAM

ACTORS :

- PLAYER
- CUSTOMER

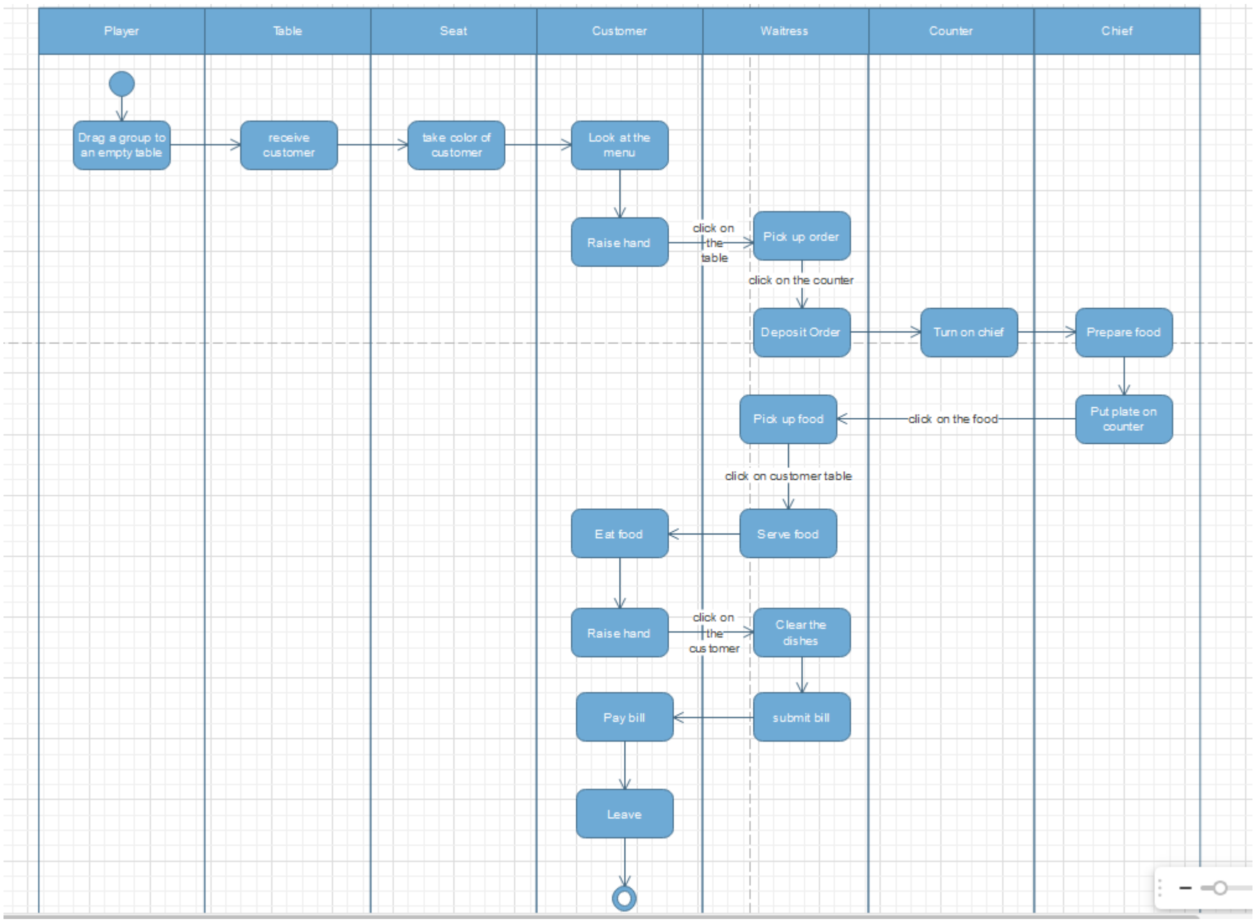
- WAITRESS

- TABLE

- SEAT

- COUNTER

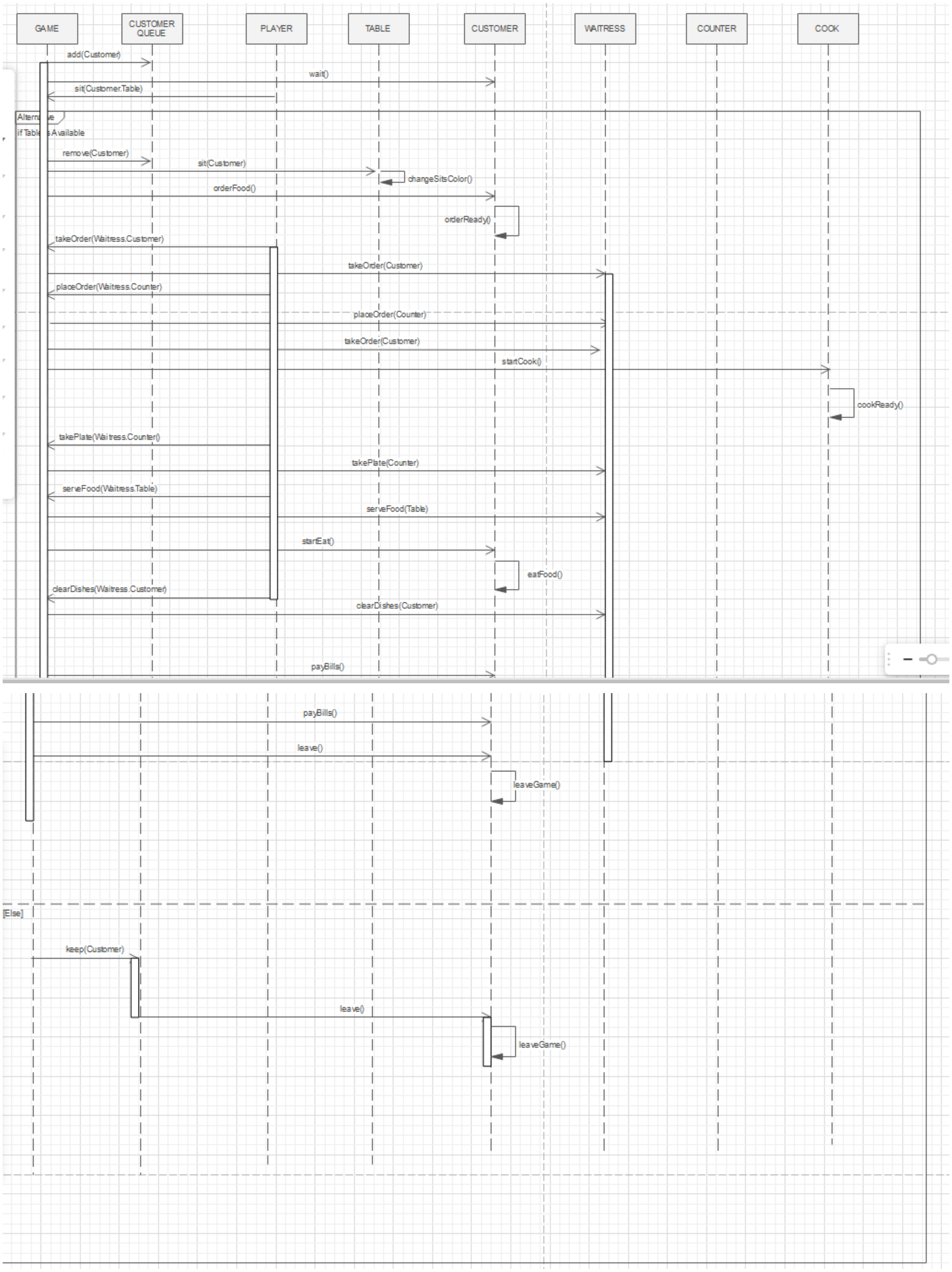
- CHIEF



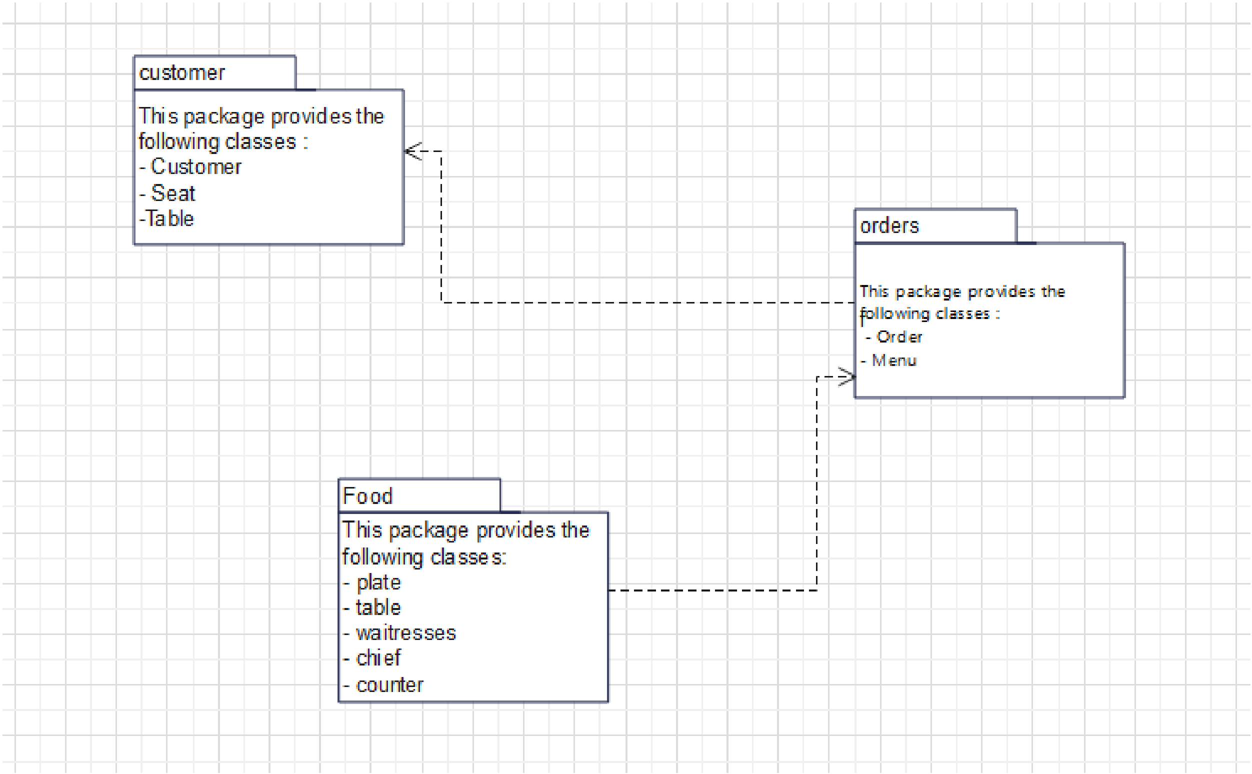
SEQUENCE DIAGRAM

ACTORS :

- PLAYER
- CUSTOMER
- WAITRESS
- TABLE
- SEAT
- COUNTER
- CHIEF cry f


GAME
CUSTOMER QUEUE
PLAYER
TABLE
CUSTOMER
WAITRESS
COUNTER
COOK
add(Customer)
wait()
sit(Customer.Table)
Altern ve
if Table s Available
remove(Customer)
sit(Customer)
changeSitsColor()
orderFood()
orderReady()
takeOrder(Waitress.Customer)
takeOrder(Customer)
placeOrder(Waitress.Counter)
placeOrder(Counter)
takeOrder(Customer)
startCook()
cookReady()
takePlate(Waitress.Counter)
takePlate(Counter)
serveFood(Waitress.Table)
serveFood(Table)
startEat()
eatFood()
clearDishes(Waitress.Customer)
clearDishes(Customer)
payBills()
payBills()
leave()
leaveGame()
[Else]
keep(Customer)
leave()
leaveGame()

PACKAGE DIAGRAM


customer
This package provides the following classes :
- Customer
- Seat
-Table
orders
This package provides the following classes :
- Order
- Menu
Food
This package provides the following classes:
- plate
- table
- waitresses
- chief
- counter