VIDZEME UNIVERSITY OF APPLIED SCIENCES

VIDZEME UNIVERSITY OF APPLIED SCIENCES

FACULTY OF ENGINEERING

PYTHON OOP AND MODELLING

GROUP 01

PRACTICAL WORK #4

VALMIERA, 2021

VIDZEME UNIVERSITY
OF APPLIED SCIENCES

Table of contents

Document versions			
Version	Status / Changes	Date	Author
1.0	Initial document release	29.12.2021	A.Jenerts
1.1	First task images added	05.01.2022	E.Sprūdžs
1.2	Content for first task	06.01.2022	A.Jenerts

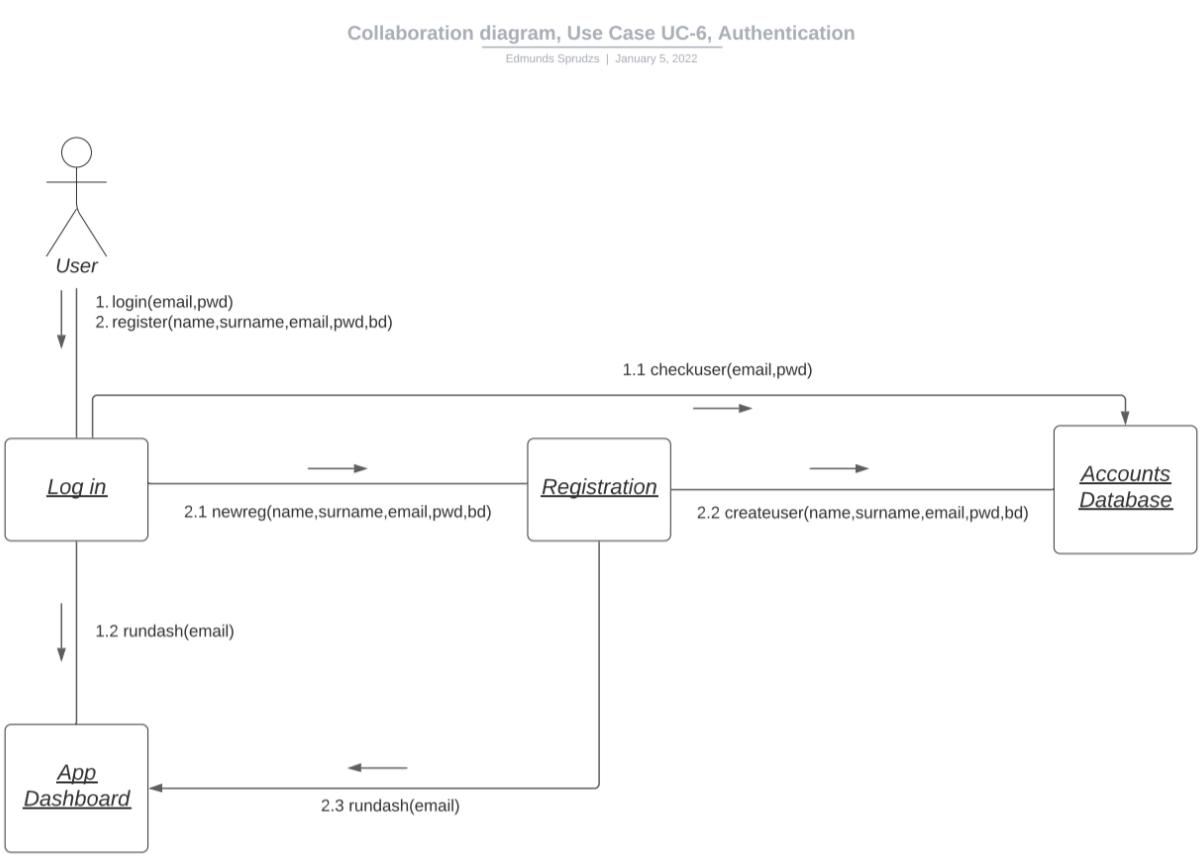
Contacts and responsible (-s)			
Name Surname	Department	Position	Contact information (e-mail)
Signe Grosberga	Group 01	Member	signe.grosberga@va.lv
Edmunds Sprūdžs	Group 01	Member	edmunds.sprudzs@va.lv
Andris Jenerts	Group 01	Member	andris.jenerts@va.lv

1 Sequence and Communication diagrams

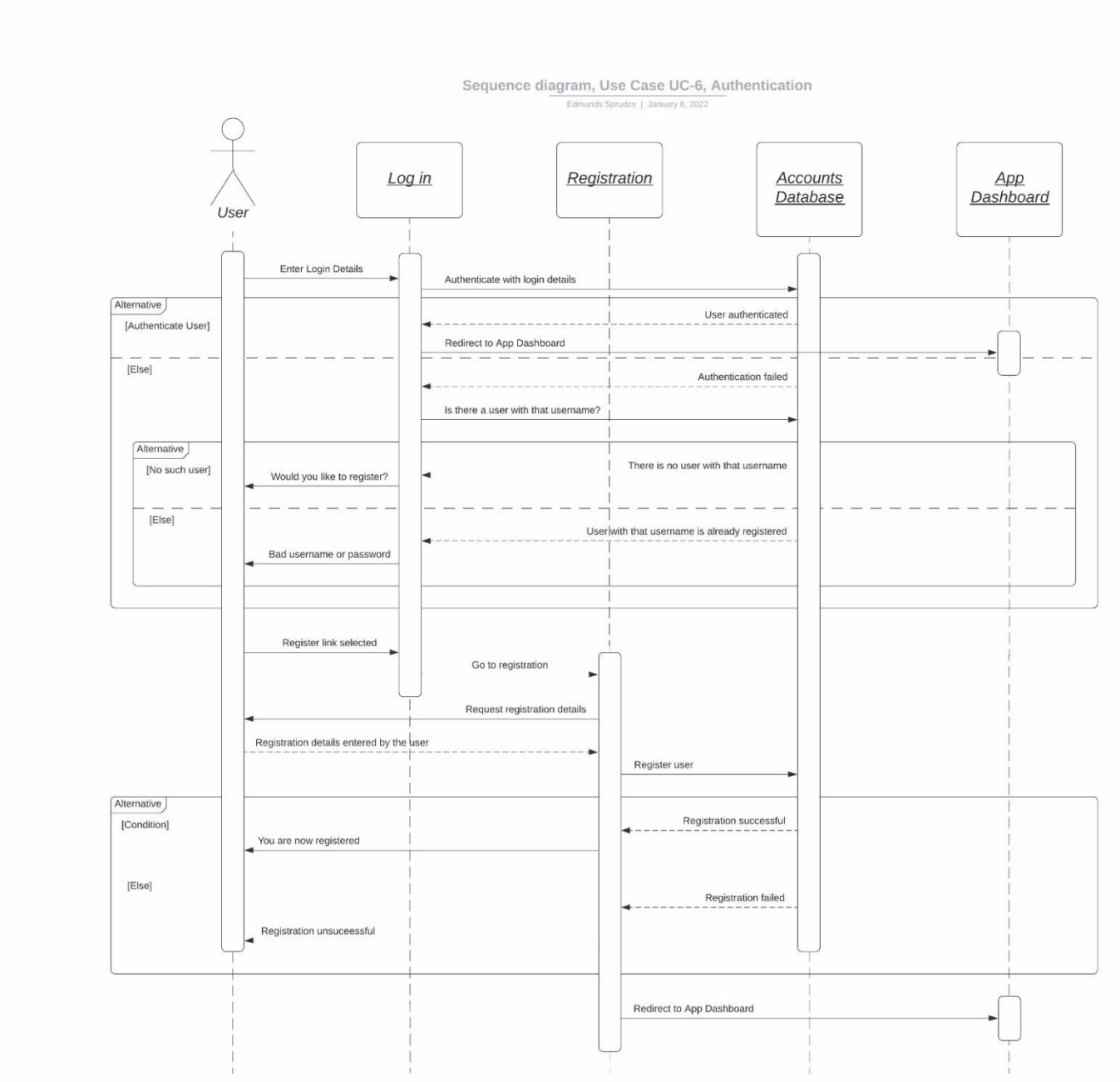
Team selected UC-6 also known as authentication use case. During diagram development some previously unnoticed details emerged. Particularly registration use case connection with the authentication.

Both diagrams were developed with Lucid App Drawing tool.

1.1 Collaboration diagram



1.2 Sequence Diagram



2 Solar System simulation task

Tasks consists of three subtasks:

- Writing method that prints all the planets in descending order from the furthest to the closest to the Sun of the Solar system.
- Make functionality that allows to specify size of the planet in the simulation screen.
- Make functionality that allows to add moons to the planets

2.1 Modifications in simulation method

To incorporate moon simulations and draw objects in more controlled manner, modifications to the simulation method ware made.

Most noticeably Planet class has been renamed Object class (as for celestial object), this class is also defined as subclass of Turtle class, so it automatically has all the turtle methods and attributes. This significantly reduces class overhead. Each object at creation time is assigned six parameters:

- iName : str – Name of the object
- iRad : float – Scale factor for the planet, planet with value 1 is a reference to all the others, e.g., planet with scale factor 2 is going to have 2 times larger radius in the simulation screen
- iDist : float – Distance from the center of the rotation in px of the screen
- iYear: float – How long one year is compared to reference planet. If Earth has value 1 Mercury has value of 0.241, because it revolves around Sun in 88 Earth days.
- iColor: str – Color representing the object in simulation.
- iCenter: obj – Sun or Object object that represents center of the rotation for this object.

As it can be seen from attributes mechanisms for simulation has been changed. In our work we decided to use a different approach. Once every time interval new coordinate for the object is calculated, this coordinate is calculated based on unit circle using sin and cos functions. Angle is increased after each movement and angle is calculated considering specified iYear parameter. Angle is specified in radians and is increased by small amount each time.

```
self.__angle += ((3.14*2)/(360*self.__year))
```

Separate method Move() has been written for the Object class that calculates new coordinate and moves the turtle. Firstly xPos and yPos is calculated based un current angle and distance to the centre of the rotation. As there is possibility that centre is also moving we add our calculated position to the centre position and use goto method of turtle to move to this position.

```
xPos = self.__distance*cos(self.__angle)
yPos = self.__distance*sin(self.__angle)

self.goto(self.__center.xcor() + xPos,
          self.__center.ycor() + yPos)
```

SolarSytsem object stores the Sun object, list of Planet objects and list of other objects, e.g., moons. During SolarSytem object initialization screen parameters are set so that we have fixed sized black screen.

2.2 Planet Sorting

Sperate method *printSystemPlanets()* in *SolarSystem* class is written for printing planets in descending order. First, new list is created so that original *__planets* list is left unmodified. Sorting is done with inbuild method for the list object. Sort method takes two optional parameters:

- Revers – for sorting in reverse order if set True.
- Key – sorting parameter.

In our case *reverse* flag is set to True because we want descending order. And *key* parameter needs to be used because *__planets* list stores pointers to the memory for *Object* objects. We need to sort by specific *Object* attribute *__distance*. Function inside sorting method is written that returns *Objects __distance* attribute. This function is passed to the *sort* method key parameter so that sorting is done by the *__distance*.

After sorting is done for loop is used to print out name of the planets. After all planets are printed name of the Sun is printed as per task description.

```
def printSystemPlanets(self):
        if not self.__planets:
            print("Solar system has no planets")

        planetList = self.__planets.copy()

        def getSortingKey(obj: Object):
            return obj.getDistance()

        planetList.sort(reverse=True, key=getSortingKey)

        for aPlanet in planetList:
            print(aPlanet)

        # print name of the sun
        print(self.__theSun)
```

2.3 Planet size representation

Size of the planet (and any other Object object) is set during initialization. Turtle method *shapesize()* is used to set x and y stretch accordingly to the passed iRad parameter. Also during the initialization turtle is moved to the correct distance of the center object. During all the described manipulations turtle is hided and only showed after they are done using *showturtle()* method.

2.4 Adding moons

Moons can be added similarly to the planets. Firstly, Object is created passing all the necessary parameters, it is important to pay attention to iYear and iCenter parameters. iYear should still be assigned as reference to the Earth year and planet should be specified as the center of rotation. This can be easily done with *SolarSystem* class method *getPlanet()* that returns planet object with corresponding name from *__planets* list.

```
moon = Object("Moon", 0.25, 30, (1 / 10), "green", ss.getPlanet("Earth"))
ss.addObject(moon)
```

Example above show how Earth moon could be added to SolarSytem object ss. Moon will have 0.25 times smaller radius than Earth, located 30 px from the center of Earth and will rotated around Earth 10 times faster than Earth will rotate around Sun.

3 Used resources

1. https://docs.python.org/3/library/stdtypes.html#list.sort
2. https://www.jpl.nasa.gov/edu/pdfs/scaless_reference.pdf
3. https://nssdc.gsfc.nasa.gov/planetary/factsheet/
4. https://www.geeksforgeeks.org/difference-between-sequence-diagram-and-collaboration-diagram/