Assessment 2 - Term Paper

UC2PSD051 - Professional Software Development

Kristofer DeYoung
Seminarie gatan 6, 41313 Gothenburg, Sweden

02 - December - 2022

Contents

1 Requirements Engineering and System Design

1.1 Functional requirements

Write four functional requirements for this system:

- The system provides a user with the ability to book a train ticket after their given preferences Sommerville (2016)
- The system will validate a client's payment information/status as a legal entity to process the transaction.
- The system needs to distinguish between personal and organizational bookings, thus allowing access to relevant information in both scenarios
- The system will process the preferred choices and check for availability on the chosen route.
- The user can add multiple e-mail recipients for booking confirmations; the receipt and booking confirmation, by default, always be sent to the client.

1.2 Non-functional requirement

Write one non-functional requirement for EACH of the following categories:

1.2.1 Product Requirements

The product must be portable/compatible with API allowing for integration with business solutions for travel booking and a standalone unit through a booking web page. It must be fast and reliable yet consistent, mistakes such as double booking should be non-existent.

1.2.2 Organizational Requirements

The system must allow high availability and consistency; because of sensitive data such as passport number/s, security the system should not store any excess data on the user after processing the transaction or inputting passengers.

1.2.3 External Requirements

Processing transactions will require connecting external services when validating the client's payment information and the transaction. The service will utilize a trusted external source to check the validity of the passenger's passport number and Name as a legal entity.

1.3 Class Diagram

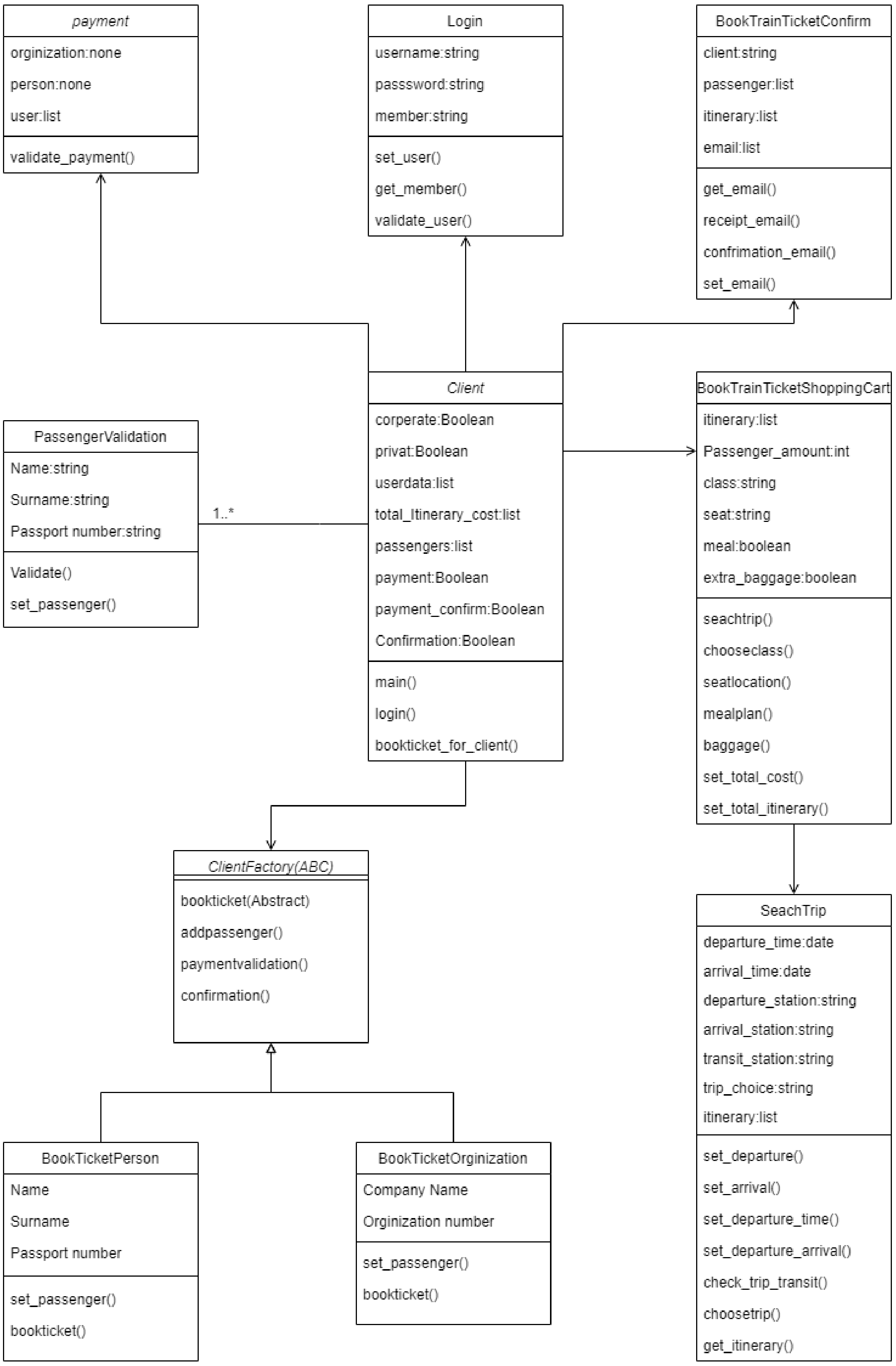


Figure 1. class diagram

1.4 UML Diagram

Explain the difference between one-to-one relationships and one-to-many relationships by designing the UML example for each:

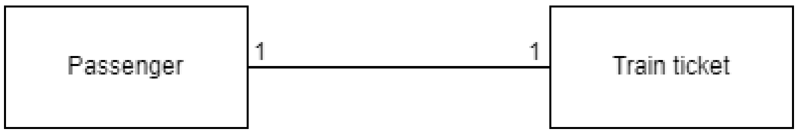


Figure 2. **One to One relationship; One passenger can only have one train ticket**

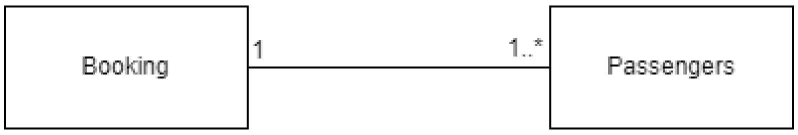


Figure 3. **One to Many relationship; One booking can have many passengers**

1.5 Use case diagram

Draw the composite use case diagram from the client actor with five interactions that can be performed:

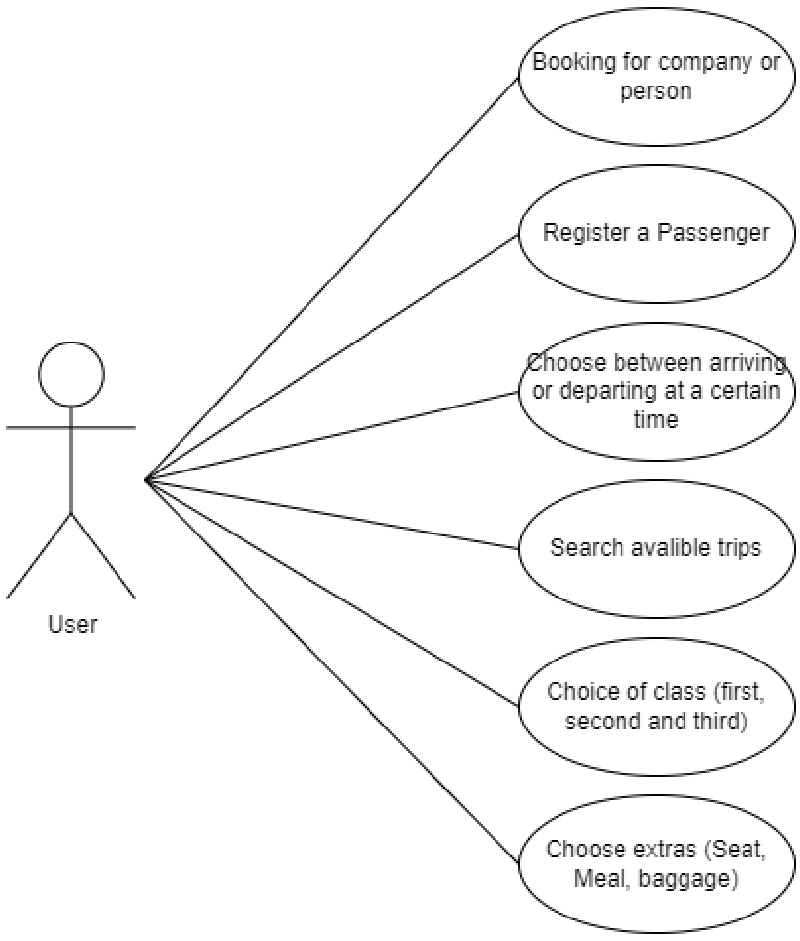
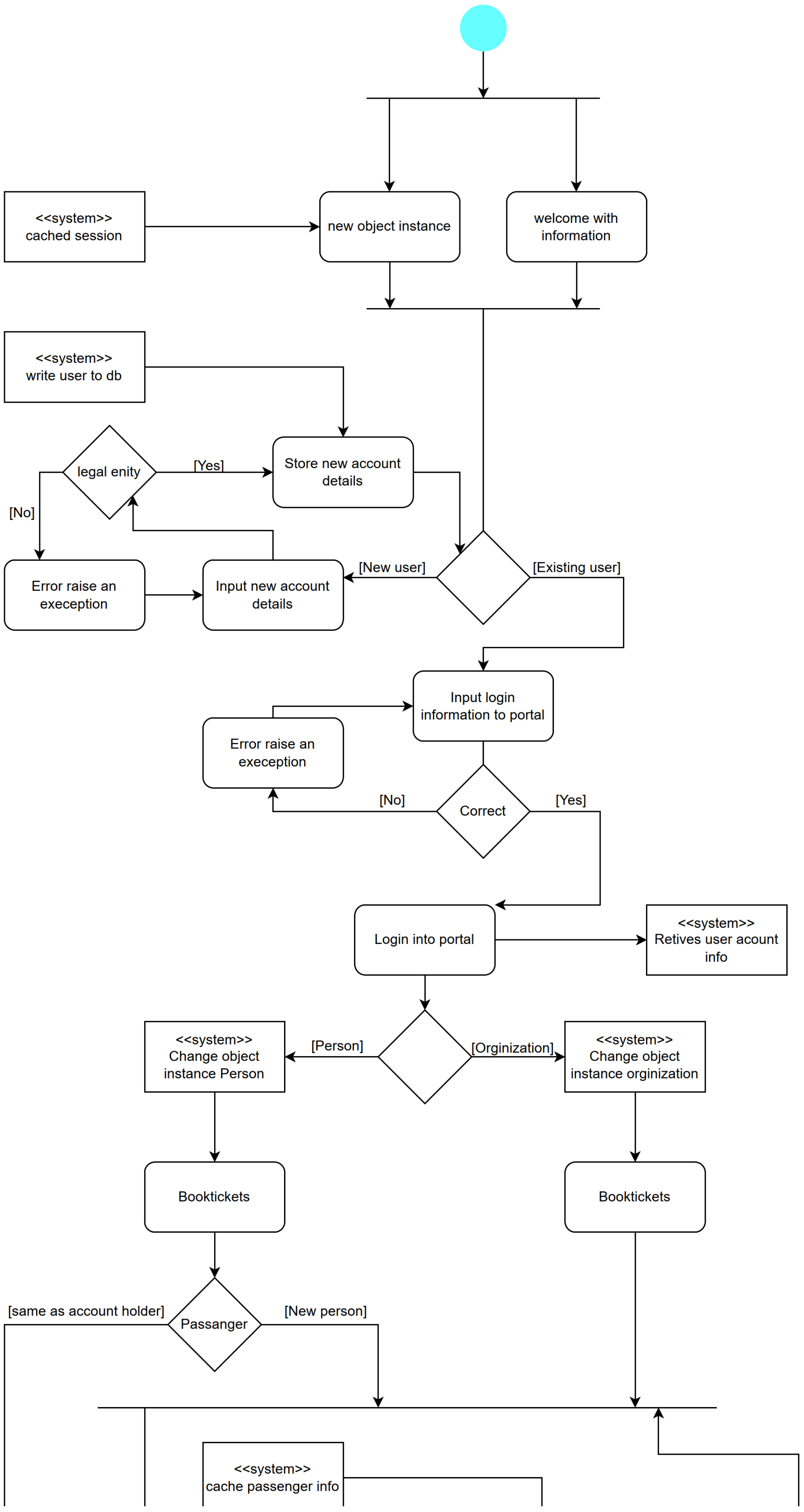

Booking for company or person
Register a Passenger
Choose between arriving or departing at a certain time
Search avalible trips
Choice of class (first, second and third)
Choose extras (Seat, Meal, baggage)
User

Figure 4. **Use case diagram, with 6 activities be performed for the user**

1.6 UML activity diagram



<<system>> cached session
new object instance
welcome with information
<<system>> write user to db
legal enity
[Yes]
Store new account details
[No]
Error raise an exeception
Input new account details
[New user]
[Existing user]
Input login information to portal
Error raise an exeception
[No]
Correct
[Yes]
Login into portal
<<system>> Retives user acount info
<<system>> Change object instance Person
[Person]
[Orginization]
<<system>> Change object instance orginization
Booktickets
Booktickets
[same as account holder]
Passanger
[New person]
<<system>> cache passenger info

Figure 5. UML activity diagram continues on the next page

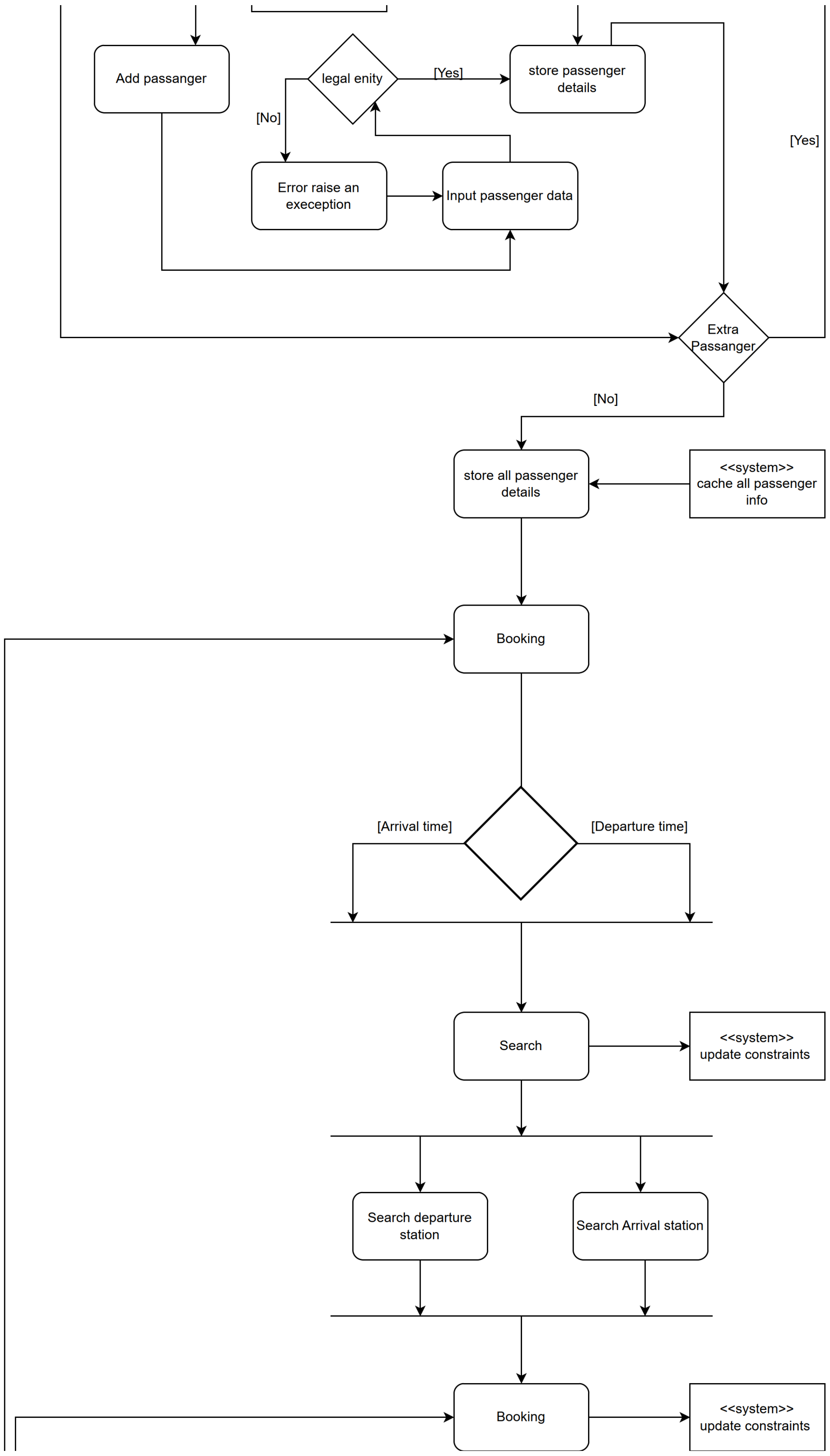
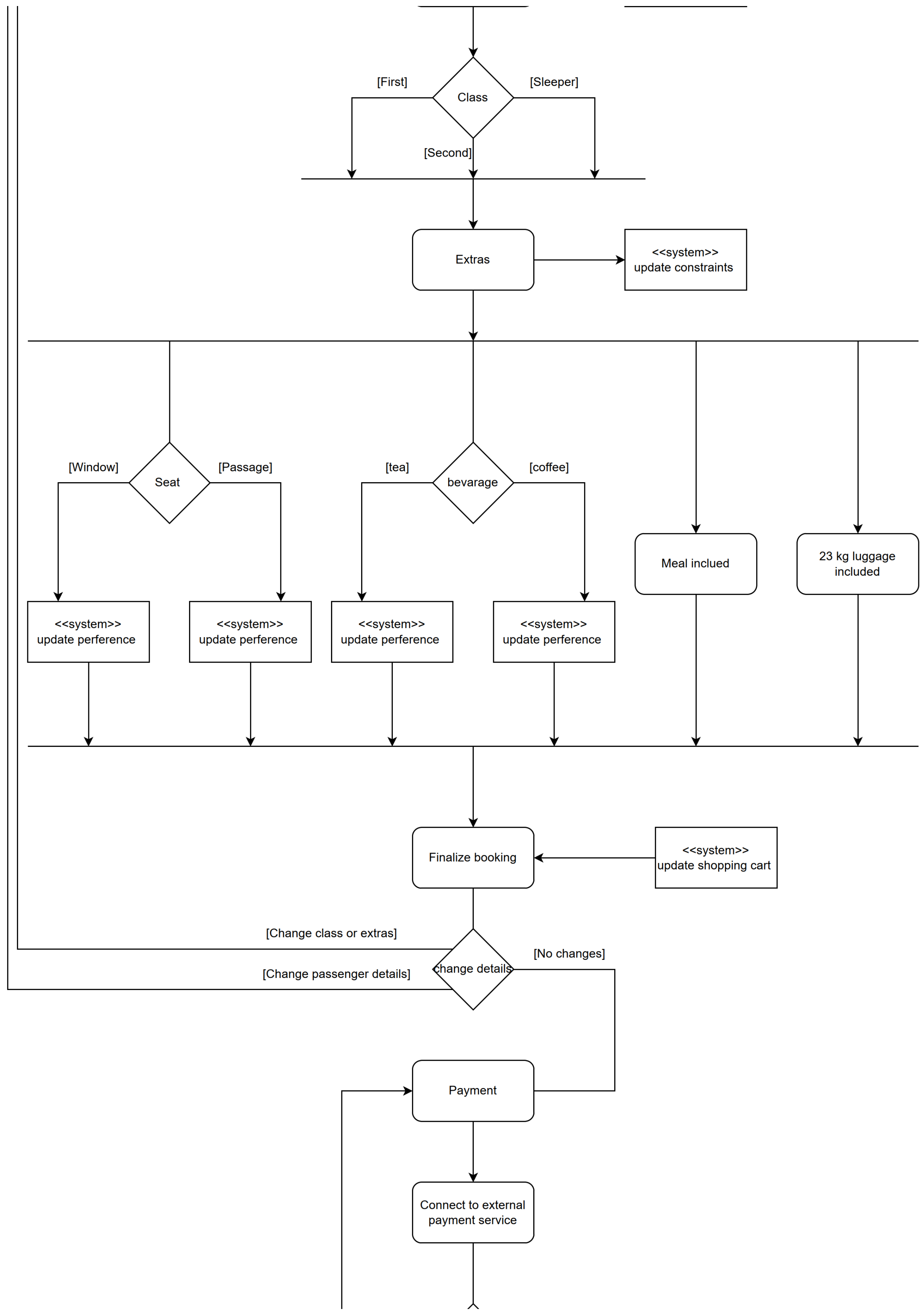


Figure 6. UML activity diagram continues on the next page



[First]
Class
[Sleeper]
[Second]
Extras
<<system>> update constraints
[Window]
Seat
[Passage]
[tea]
bevarage
[coffee]
Meal inclued
23 kg luggage included
<<system>> update perference
<<system>> update perference
<<system>> update perference
<<system>> update perference
Finalize booking
<<system>> update shopping cart
[Change class or extras]
[No changes]
change details
[Change passenger details]
Payment
Connect to external payment service

Figure 7. UML activity diagram continues on the next page

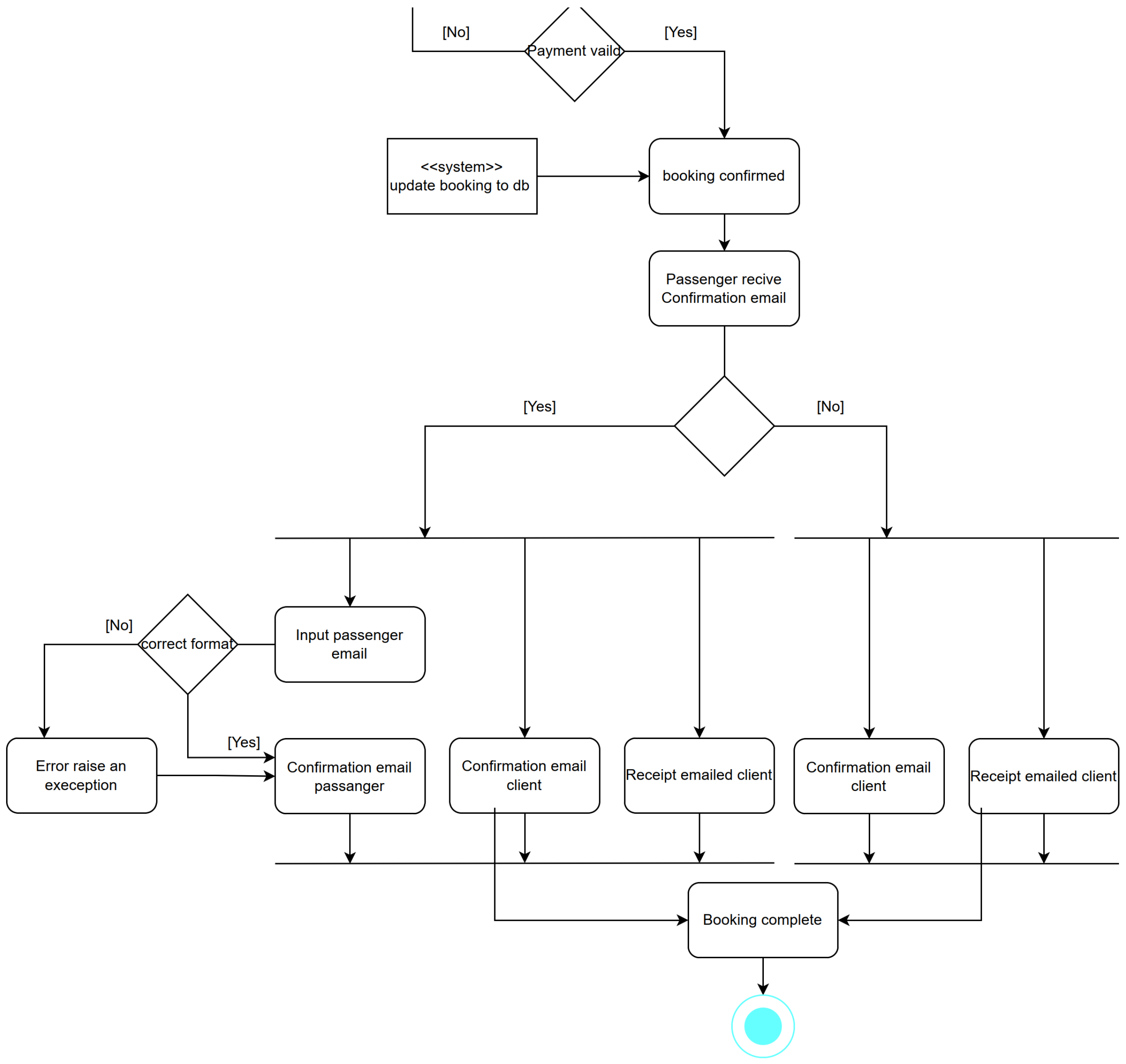

[No]
Payment vaild
[Yes]
<<system>> update booking to db
booking confirmed
Passenger recive Confirmation email
[Yes]
[No]
[No]
correct format
Input passenger email
[Yes]
Error raise an exeception
Confirmation email passanger
Confirmation email client
Receipt emailed client
Confirmation email client
Receipt emailed client
Booking complete

Figure 8. **UML activity diagram last figure**

1.7 Sequence diagram

Draw the sequence diagram, which depicts the client logging into the booking system portal. Write a short summary of your understanding of how they login to aid in your discussion on the diagram:

The login process starts from the main page; if the user chooses a new user, the program opens another object; however, by selecting the option "existing user" the client is allowed to input their username and password on the login page. The information is then passed on to the object handling the member's database. From Members, the database object returns authorization for either a corporate or private account. Thus the login page now instantiates the correct object according to account privileges, and the client receives feedback that the login is OK.

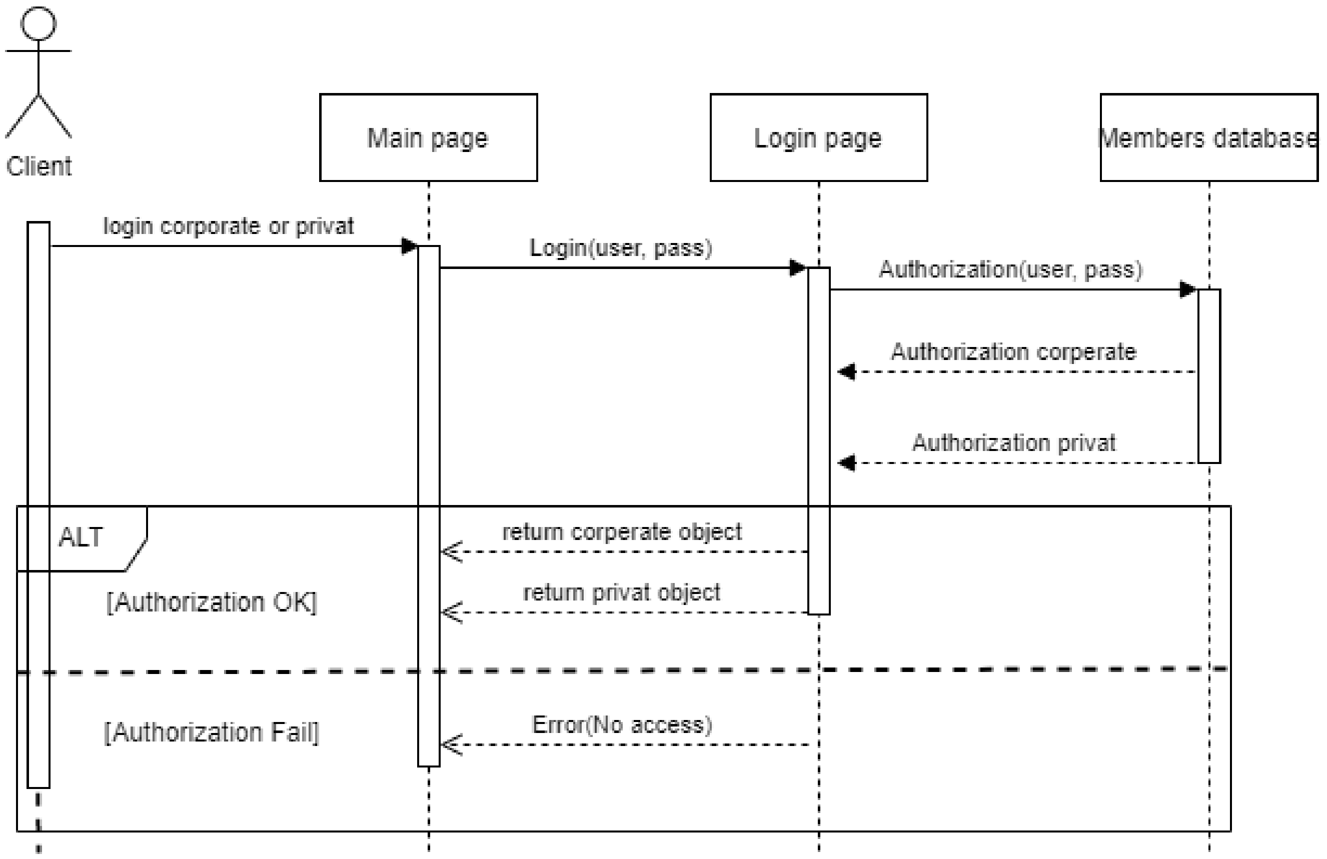


Figure 9. Sequence diagram

2 Software Testing Methodologies

2.1 Scenario Testing

Design 4 scenarios in order to perform "Scenario Testing" of the system from Question 2. For each of your test scenarios create at least two Test Cases (total number of scenarios can be bigger):

2.1.1 Scenario 1 - Search

Case Id	Objectives	Pre-requisites	Test data	Test steps	Expected results
1	Check if the Application search object works	The user is confirmed	none	Press button	search page (object) displays 6 boxes to & from station and departure and/or arrival time.
2	Check valid departure/arrival time input.	The user has chosen either departure or arrival time	Arrival time= 15:00 17.01.2022	Input data and wait.	Validating if data is correct in time/format, and giving feedback to the user.
3	Check valid to & from station input	The user has inputted date and time	From = London To = Berlin	Input from and to data	Search page (object) displays alternative routes within the constraints of departure time or arrival time

2.1.2 Scenario 2 - Person or Company

Case Id	Objectives	Pre-requisites	Test data	Test steps	Expected results
1	To call the object instance for the company or person	software is running	none	Press button	Related page (object) displays relevant input boxes
2	Check for a valid input Passport number, name, and surname	The user has pressed the button for person	Passport nr:9900123131 name:John surname:doe	Input data and wait.	Validating if data is correct in time/format, and giving feedback to the user.

2.1.3 Scenario 3 - Validating passenger

Case Id	Objectives	Pre-requisites	Test data	Test steps	Expected results
1	Check valid input for passenger detail passport nr., Name, Surname, Email	Payment method confirmed	passport nr:9090909754 Name:John Surname:Doe Email: jd@gmail.com	Input data and wait	feedback is given that data either is correctly formatted or too comply with standards
2	Check the registry to Validate person exists as a legal entity	The client has complied with all formatting standards and pressed the booking button	Passport nr:9090909754 Name:John Surname:Doe Email: jd@gmail.com	data automatically transferred from the previous step	Validation OK; the user is moved on to the booking page; if not moved back to the previous page

2.1.4 Scenario 4 - Booking system

Case Id	Objectives	Pre-requisites	Test data	Test steps	Expected results
1	Choose arrival or departure time	The client has entered the booking page	none	Press button	Moves client to object instance to set arrival or departure Search page (object)
2	Passes the chosen trip back from the search object and all available choices for the current trip	The user has chosen a searched trip	A list of available choices	wait	Only the available choices for the current trip should accessible for the client
3	Check if the trip updates preferences on chosen class	previous user test OK	class= First class	button press	limits the choices relevant to first class on seat location, meal plan, and baggage
3	Check if available seat-location, meal plan and baggage update as intended	previous user test with chosen class	seat-location= 1a, meal-plan= Kobe-beef, baggage= super-sized	button press	trigger object confirm trip from "shopping cart" saving to database "booking the ticket"

2.2 Validation testing

You are provided with an Air Conditioner (AC), which should work both for warming and cooling. Provide 2 validation tests and two defective tests that can be performed on AC. Provide assumptions that you have made for this test):

I have assumed that the AC is of standard design and automatically maintains temperature and airflow in the room in accordance with the user's temperature setting.

2.2.1 Validation

- Check if every temperature setting that is warmer(larger numeric temperature value) than ambient temperature, the unit is expected to heat.
- Check if every temperature setting that is colder(smaller numeric temperature value) than ambient temperature, the unit is expected to cool.
- Check if the AC unit is set to auto airflow; it will default to the least airflow possible unless the difference between ambient and user temperature becomes too great, then it will increase airflow appropriately.

2.2.2 Defective

- The temperature of airflow from the AC unit differentiates from user expectation when setting the temperature to either colder or warmer than the ambient temperature.
- The unit does meet expected airflow when the set temperature differentiates greatly from ambient temperature, I.e larger the difference, the greater the airflow. Resulting in it taking to long to regulate temperature.

2.3 Differences

Explain the difference between unit testing, component testing, and system testing using an example.

Unit testing a computer table would consist of multiple individual tests, such as testing the table's surface strength, weight limitation, and durability. In separate cases, unit testing could also be performed on the raising and lowering system of the table or checking the strength, durability, and stability of the table legs.

Through component testing, we will test the lifting and lowering unit whilst connected to the table surface. Will table strength requirements be sufficient, or will the table surface contort, twist, or vibrate under the force of the raising mechanism.

The System test will be encompassing all different components and make sure interact correctly together. Using our previous example, we would instead be testing the complete computer table with legs, a lifting unit, and the table surface and checking if the table is standing with a computer and screens(weight requirements) and if it is able to be raised or lowered when required without unwanted consequences such as falling over or breaking. Will all the components work together as expected whilst still maintaining the performance demonstrated in the initial unit and component testsChirchenkova (2022)

3 Software Development Life Cycle

3.1 Pros and cons waterfall method

Discuss the pros and cons of using waterfall method versus using an agile method with respect to:

- Managing Project Risk
- Project Tracking
- Software Architecture Design

Project Tracking A waterfall approach completes each stage individually, which allows for more transparency into tracking the progression of the project. Sommerville (2016) Despite allowing stakeholders to get more clarity, the waterfall method is often not an optimal method for a lot of software. Often for software, the design and implementation are interleaved with each other in the development stage, where components need to be developed in parallel; thus, an agile approach could allow for further flexibility.Chirchenkova (2022)

Software Architecture Design The requirement flexibility and constant feedback loop that makes the agile approach a stronger candidate for a lot of software also makes it difficult to define every aspect within software design architecture; many projects can often only define a high-level functionality as it still remains unclear to what the full scope of the project will be. In contrast, a project with a clear set of rules and goals will benefit greatly from a waterfall model, and software architecture design, not only for defining the different development timelines; but because it's easier to define the requirements of the end product when following the steps of architectural design being Performance, Security, Safety, Availability and Maintainability.Sommerville (2016)

Managing Project Risk Managing project risk in the waterfall method is a lot easier since the requirements and steps are more clearly defined, and every phase is validated before moving on. Hoory and Bottorff (2022) The ability to foresee the development cycle is a lot more clear; however, as is often the case with software, there will be unforeseen challenges or consequences, making progress with development slower than with the agile approach, where it basically assumes there will be problems. The agile approach allows for faster development but also makes it more difficult to assess the project risk. Though both Agile approaches and waterfall have their strengths and weaknesses, it is ultimately the project that determines what approach is best to use. Though Waterfall clearly has, its strengths, it should only be used when a project can be clearly defined upfront, such as in the case of large-scale software that needs integration with previous systems or a critical system where a detailed plan is present, allowing for cost-benefit analysis. The agile approach will therefore be the "goto" for many other software designs.Chirchenkova (2022)

3.2 Spiral Development Life Cycle

Discuss a project idea where the Spiral Development Life Cycle would be the best fit for the project. Justify why the Spiral Development Life Cycle would work better for this project in comparison to the Validation and Verification Life Cycle Model:

A software application targeting business with a more generalized approach to analyzing data, leveraging Machine learning and artificial would be a perfect candidate for SDLC. Since the product aims for a very broad market where there are variations in data storage and analysis differentiate, the SDLC would allow the team to develop and launch either individual modules or only initially to a more specific audience. Through Verification and validation (VV), both the utilization and user-bases would have to be clearly defined, the upfront time and cost for such product using VV would also be much higher upfront and taken into consideration that this still a fairly new domain, it would increase the risk and make it harder to gain initial capital investment.

The product's main goal is to provide analysis of a data set without en-curing the cost in-house competency. For obvious reasons, it would be very difficult to expect the same results as in-house competency, However, many companies can neither afford nor justify an expense that carries so much risk with it, as they do not yet know if it has any utility. The potential is, however, that one can still leverage some of the data for decision-making with a very easy interface. Whilst utilizing feedback and ideas to improve each iteration of the product in the SDLC.

When considering the VV model in this context, we again have more difficulties considering that the product is targeting clients that are underdeveloped in data-driven decision-making, and the field of AI/ML itself is still only in its infancy. One could argue we will see tremendous changes within the near future, further enforcing the necessity of SDLC so product evolves in a rapidly changing environment.

4 Version Control

4.1 Reasons

Four reasons for using a version control system:

1. Ability to rollback in case of unwanted changes or bugs
2. Transparency and traceability in the changes made
3. Easier Collaboration
4. Quality adherence

4.2 centralized and distributed

Both centralized and distributed version control systems are commonly used.

4.2.1 Advantages

Provide three advantages of distributed version control systems over centralized version control systems:

1. Each developer has a local copy and can commit offline, ensuring further flexibility. Patel (2020)
2. Repository will most likely have several local clones, thus will be multiple points of failure. Patel (2020)
3. Developers no longer have to keep track of merges and branches as a single check-in system. Patel (2020)

4.2.2 Example

Provide an example of a centralized control system and a distributed version control system:

1. CVS, Perforce, and SVN are centralized control systems where a request to pull the latest version from a server. Google has used such a system for over 20 years, and as their team grows larger and the software more complex, they always maintain a "Single Source of Truth"Gehman (2020)
2. Git is an example of a distributed version control where each user can have their own local copy "branching" the main.

4.3 Branching

Branching is commonly used in version control systems.

4.3.1 Branching?

What is branching?

Branching is the ability for developers to create separate copies of programs or objects, allowing them to make changes without impacting the mainline, trunk, or master line.Contributor (2021) Thus, the main branch is kept clean from unfinished changes till the developer is ready to merge them. Git (2022)

4.3.2 Examples

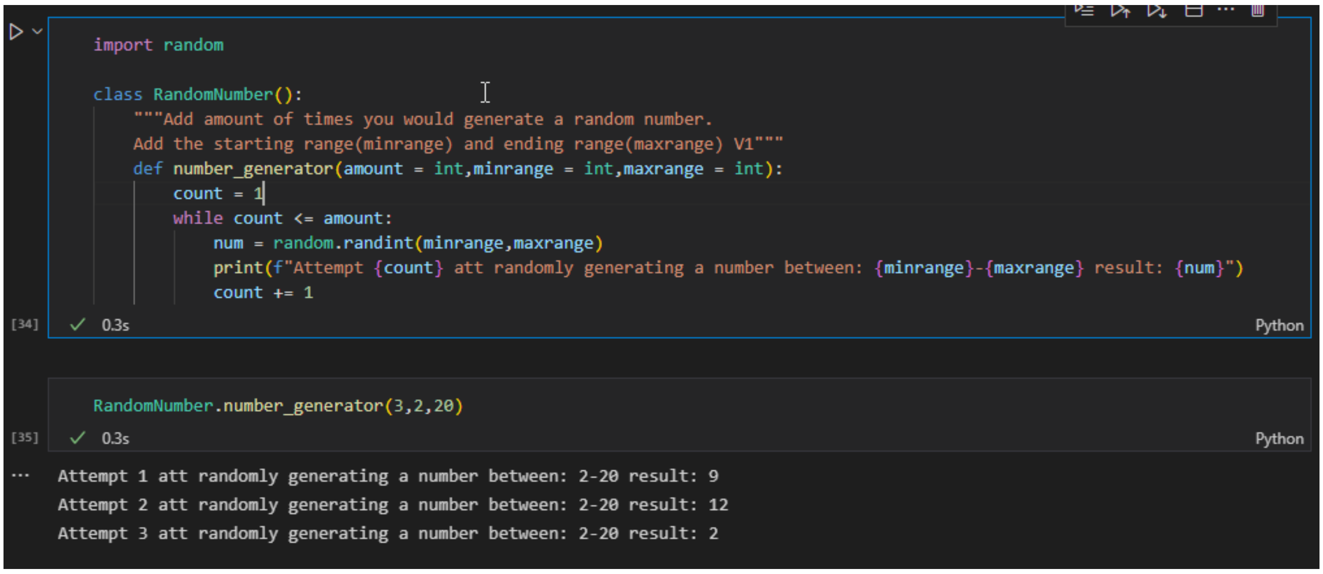
Provide 3 uses of branching with examples:

1. Gits distributed version control system is an example of branching. Essentially the developer creates a branch of the master. When the developer wants to merge the changes, he creates a pull request to the main branch; thus, the main branch is kept clean from unfinished changes.Git (2022)
2. Creating a new code line that may develop individually from a version of a preexisting code line. Sommerville (2016)
3. Storage management in its early days was also a reason for version control with separate branches instead of each developer having a full-functioning copy of the software, the system stored a list of "deltas" which were the differences between different versions.Sommerville (2016)

5 Working with Git

The following sections provide print screens of the python program and committing data to the Git repository, which can be found on the following link: My Repository

5.1 Random number generator



```
import random

class RandomNumber():
    """Add amount of times you would generate a random number.
    Add the starting range(minrange) and ending range(maxrange) V1"""
    def number_generator(amount = int,minrange = int,maxrange = int):
        count = 1
        while count <= amount:
            num = random.randint(minrange,maxrange)
            print(f"Attempt {count} att randomly generating a number between: {minrange}-{maxrange} result: {num}")
            count += 1
```

[34] ✓ 0.3s Python

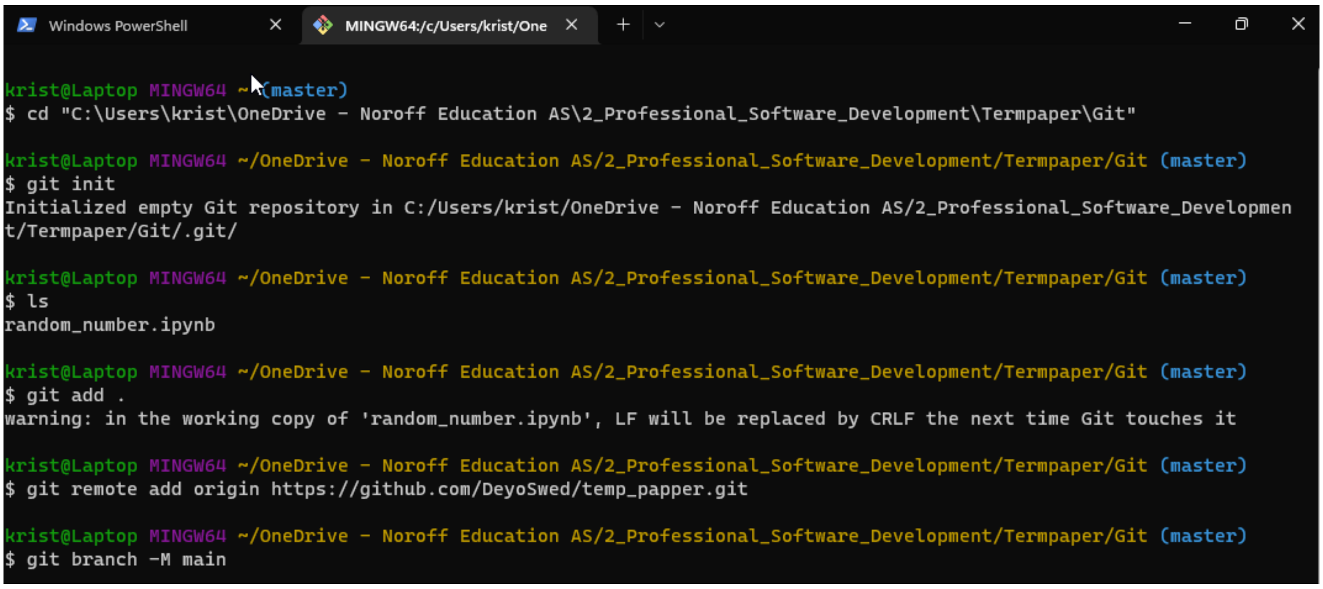
```
RandomNumber.number_generator(3,2,20)
```

[35] ✓ 0.3s Python

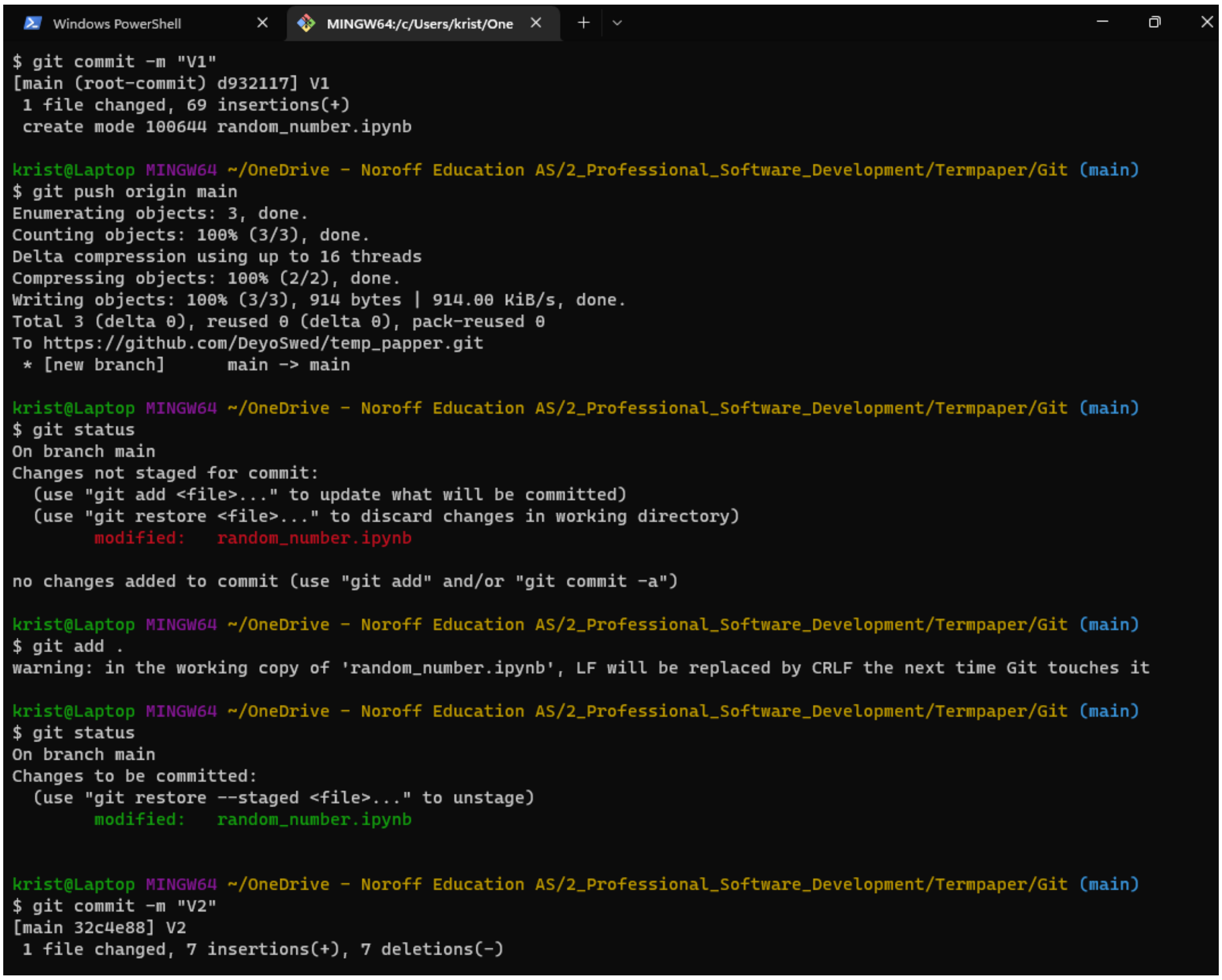
```
Attempt 1 att randomly generating a number between: 2-20 result: 9
Attempt 2 att randomly generating a number between: 2-20 result: 12
Attempt 3 att randomly generating a number between: 2-20 result: 2
```

Figure 10. The random number generator showcasing functionality

5.2 Committing to Git

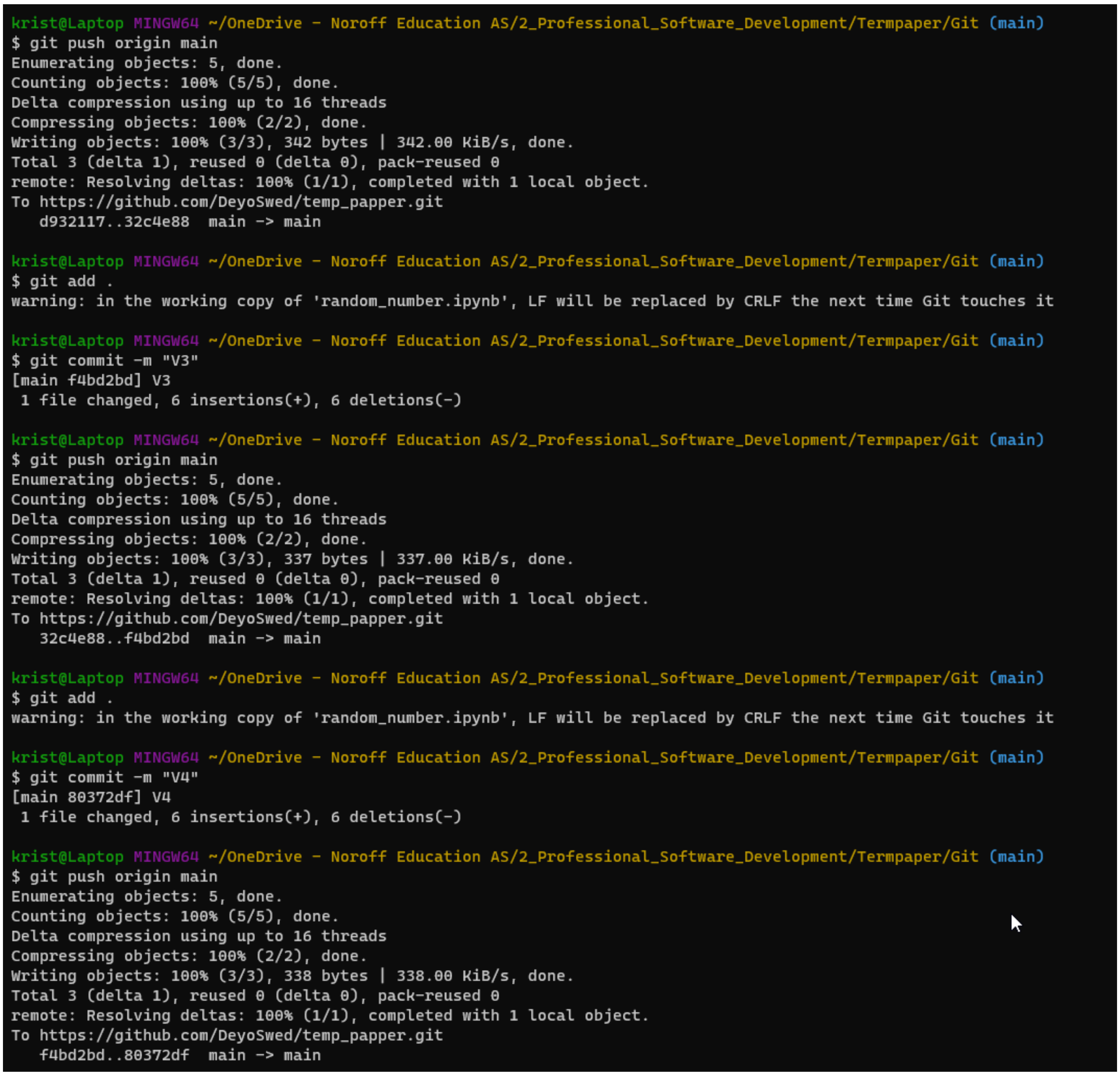


```
krist@Laptop MINGW64 ~ (master)
$ cd "C:\Users\krist\OneDrive - Noroff Education AS\2_Professional_Software_Development\Termpaper\Git"

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (master)
$ git init
Initialized empty Git repository in C:/Users/krist/OneDrive - Noroff Education AS/2_Professional_Software_Developmen
t/Termpaper/Git/.git/

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (master)
$ ls
random_number.ipynb

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (master)
$ git add .
warning: in the working copy of 'random_number.ipynb', LF will be replaced by CRLF the next time Git touches it

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (master)
$ git remote add origin https://github.com/DeyoSwed/temp_papper.git

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (master)
$ git branch -M main
```



```
$ git commit -m "V1"
[main (root-commit) d932117] V1
 1 file changed, 69 insertions(+)
 create mode 100644 random_number.ipynb

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 914 bytes | 914.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DeyoSwed/temp_papper.git
 * [new branch]      main -> main

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   random_number.ipynb

no changes added to commit (use "git add" and/or "git commit -a")

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git add .
warning: in the working copy of 'random_number.ipynb', LF will be replaced by CRLF the next time Git touches it

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   random_number.ipynb


krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git commit -m "V2"
[main 32c4e88] V2
 1 file changed, 7 insertions(+), 7 deletions(-)
```

Figure 11. The process of the original commit to Git repositories and further updating the original with 3 modified versions (continues)



```
krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DeyoSwed/temp_papper.git
   d932117..32c4e88  main -> main

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git add .
warning: in the working copy of 'random_number.ipynb', LF will be replaced by CRLF the next time Git touches it

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git commit -m "V3"
[main f4bd2bd] V3
 1 file changed, 6 insertions(+), 6 deletions(-)

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 337 bytes | 337.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DeyoSwed/temp_papper.git
   32c4e88..f4bd2bd  main -> main

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git add .
warning: in the working copy of 'random_number.ipynb', LF will be replaced by CRLF the next time Git touches it

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git commit -m "V4"
[main 80372df] V4
 1 file changed, 6 insertions(+), 6 deletions(-)

krist@Laptop MINGW64 ~/OneDrive - Noroff Education AS/2_Professional_Software_Development/Termpaper/Git (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DeyoSwed/temp_papper.git
   f4bd2bd..80372df  main -> main
```

Figure 12. The process of the original commit to Git repositories and further updating the original with 3 modified versions

References

Ian Sommerville. *Software engineering*. Always learning. Pearson, 10. ed., global ed edition, 2016. ISBN 978-1-292-09613-1.

Mariya Chirchenkova. Lecture software testing, February 2022.

Leeron Hoory and Cassie Bottorff. What is waterfall methodology? here's how it can help your project management strategy, March 2022. URL https://www.forbes.com/advisor/business/what-is-waterfall-methodology/.

Suri Patel. Why you should move from centralized version control to distributed version control, 2020. URL https://about.gitlab.com/blog/2020/11/19/move-to-distributed-vcs/.

Chuck Gehman. Why did google choose a central repository?, 2020. URL https://www.perforce.com/blog/vcs/why-did-google-choose-centralized-repository.

TechTarget Contributor. branching, 2021. URL https://www.techtarget.com/searchitoperations/definition/branching.

What is a git workflow?, 2022. URL https://about.gitlab.com/topics/version-control/what-is-git-workflow/.