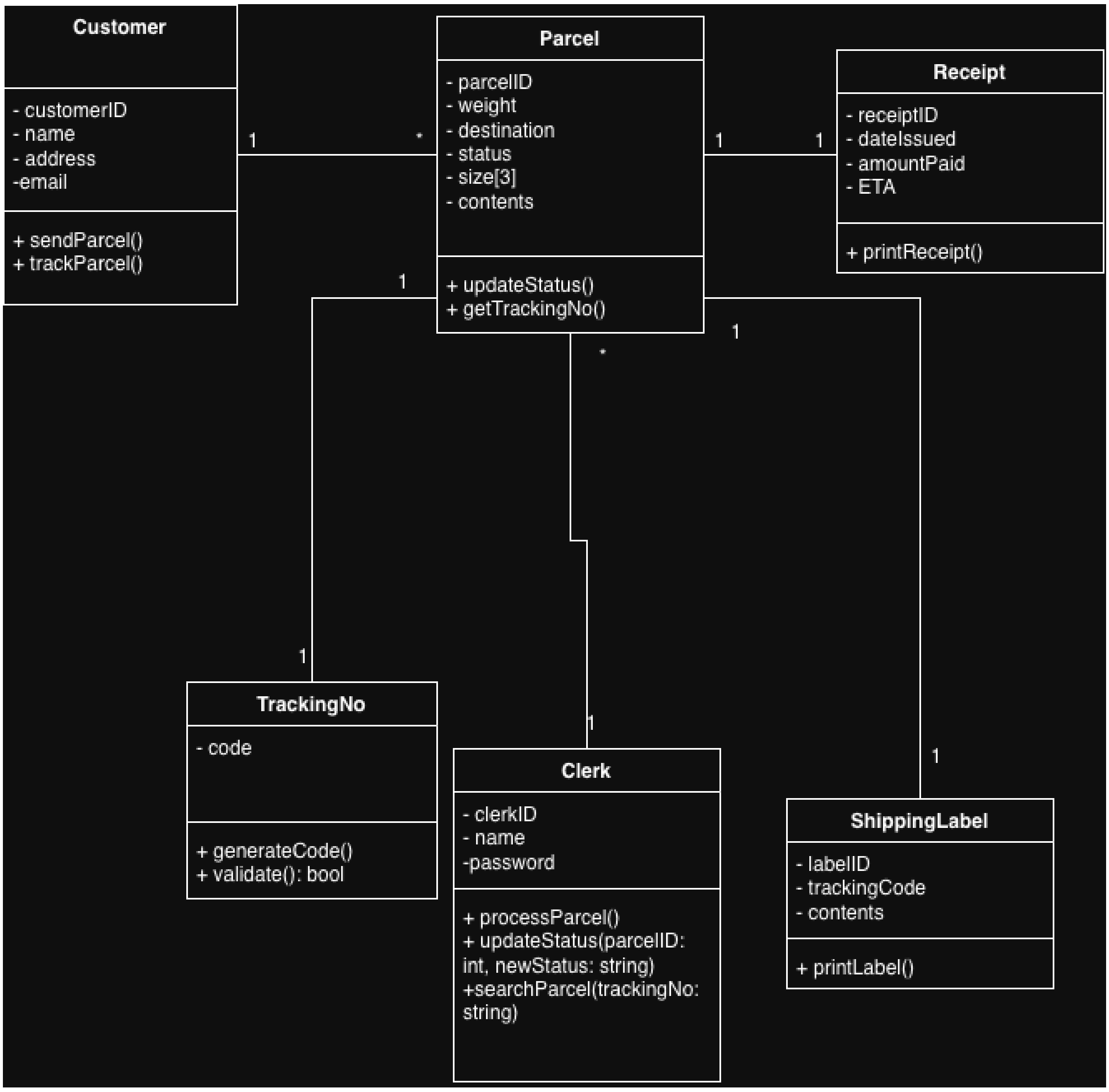
EE2010 Software Engineering
Software Engineering Report, Part 2
Group 7:*Rhys, Ivan, Dimitris, Vadym*

1. Introduction

This report provides a reflection of the design and requirements testing for the project ***mail_system***. It contains an evaluation of each of those phases and the experience gained through the project.

2. Reflection on Design



7

The UML diagram that I created was inspired by how real post offices work. In a real post office, you would have a clerk working there, a parcel being processed, and of course the customer. These were the first three classes I created. The UML made it easier for me to visualise the connections between these classes. For example, you can have one clerk handling many parcels, but not many clerks for one parcel. After that, I added the TrackingNo, ShippingLabel and Receipt classes, which are all associated with the Parcel. Having the Parcel in the centre of the diagram helped me understand how the process flows from class to class. The hardest challenge for me was deciding which attributes belonged to which class. At first, I put the ETA inside the TrackingNo instead of the Receipt, but then I realised the ETA needs to be written on the receipt so that the customer can see it.

One of the biggest benefits of using UML to draw the diagram, is that it allowed me to visualise multiplicities early and therefore did not cause any confusion in the code

3. Reflection on Requirements Testing

Testing has been a very crucial step in the design process of this project. It allowed us to evaluate the final program and see how loyal we have stayed to the specification. To ensure a thorough requirements test, we have used some software testing methods. One of the methods we used was boundary testing. Boundary testing checks how the program behaves at the limits of valid input ranges, because errors often occur at these edges (GeeksforGeeks, *Software testing - boundary value analysis* 2025). We tested this with the main menu, where the program asks you to input a number corresponding with the 3 options. We tried entering zero, negative numbers and numbers bigger than three and I could see that it passed this test. Another method we used was equivalence partitioning, which basically means I test one value from each input group (GeeksforGeeks, *Equivalence partitioning method* 2025). For example when we were checking the login system, we tested a valid username, an invalid username and one that has been already registered. All these inputs represent a different input category. The final testing method we employed was the scenario-based testing method, which essentially attempts to follow complete user stories such as how the customer would behave when using the program (PEGA Academy, 2020).

Note: I would not expect to see a report of more than 3 pages.

4. References

- GeeksforGeeks (2025) *Software testing - boundary value analysis*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/software-testing/software-testing-boundary-value-analysis/ (Accessed: 08 December 2025).
- GeeksforGeeks (2025a) *Equivalence partitioning method*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/software-engineering/equivalence-partitioning-method/ (Accessed: 08 December 2025).
- PEGA Academy (2020) *Scenario testing | Pega Academy*. Available at: https://academy.pega.com/topic/scenario-testing/v1 (Accessed: 08 December 2025).

7