Mid-Module Assignment: System Design

Introduction

Information systems are a critical aspect of operations that take place in society, especially within the business environment where they are heavily relied on to perform a multitude of tasks (Sommerville, 2016). Business processes and procedures are intricately coupled in various workflows through computer systems, which acquire data by accessing centralised platforms, and thereafter manipulate it in the required manner before it is forwarded for further processing (Bourgeois, 2014). The ongoing use of the information through its representative data is automatically performed by the information system, or employees who need to interact with it (University of Essex Online, 2021a). One such information system is that of an online store which has various functional and systemic components that need to operate in unison to tie a customer to an end-product through a transaction, in a safe and secure manner (Sommerville, 2016; Bourgeois, 2014). This document presents the design of a conceptual online store, where various of the aforementioned aspects are modelled in a mock software application. The design contains aspects of the client-side Object-Oriented (OO) program logic, required to create a large information system that would support the business of selling goods (Bourgeois, 2014).

Limitations

Due to the design being an individualistic and once-off presentation, some of the usual aspects of system design could not be used (Sommerville, 2016). For instance, the requirements analysis and design phases of the software development lifecycle are

often performed in an iterative manner, where system aspects are changed, added, or removed, and the design perfected (Sommerville, 2016). Therefore, there may be some basic elements in the actual code implementation which need to be slightly changed to facilitate any fixes to the design (University of Essex Online, 2021b). Furthermore, a team setting is often used to produce a collaborative effort, which can assist in producing a better product as different skills and views are pooled and utilised, as well as various levels of oversight applied (Sommerville, 2016).

Considering that a system of this nature would incorporate the components which fall under the information system categories of Hardware, Software, Data, People, and Processes, it would be fairly expansive in reality, containing a lot of software functionality to connect and coordinate all of the system elements (University of Essex Online, 2021a). Therefore, some of the usual functionality like the important Authentication, Authorisation, and Accounting (AAA) principles will need to be decreased, because of time and resource limitations (Nieles, 2017). Additionally, an online store would usually have a browser-based internet application, with a Graphical User Interface (GUI) for interaction (Brookshear & Brylow, 2018). In this mock system there will not be a GUI front-end but a Command Line Interface (CLI) one (University of Essex Online, 2021c). Furthermore, none of the system's data will reside on a remote database which uses a Database Management System (DBMS) (Bourgeois, 2014). Some data will be hardcoded, and the rest added during runtime by the user. Finally, there is no web server being used, as the program has been designed to be located and executed on a standalone local machine (Bourgeois, 2014).

Assumptions

All data attributes have been defined in classes as private to enforce encapsulation of the related object's data structure (University of Essex Online, 2021d). Therefore, "getter" and "setter" methods will be required to access these properties (University of Essex Online, 2021d). Some of the required getter and setter methods related to various classes may not be shown so as to prevent diagram clutter, but are assumed a given (Ambler, 2003). Furthermore, none of the constructors have been displayed as they are considered a requirement for all classes (Ambler, 2003). Additionally, because all object interactions have been viewed by interfacing abstract data structures, which have not actually been programmatically interconnected yet, there may be reactions which were not considered (Ambler, 2003). Therefore, a need for further method parameter and return values to be added to classes may arise in the programming phase (Ambler, 2003).

Design

From the requirements analysis various application functionality was determined, in addition to data attributes that would be used to reflect relevant elements and key information (Phillips, 2018). Specific properties and operations were then tied together to produce several user-defined datatypes in the form of OO classes (University of Essex Online, 2021e). As one can see below in figure 1, only seven classes were defined, so as to keep the number of objects and their respective interfaces to a minimum (Fowler, N.D). A brief class overview follows:

1. Customer objects will be created through a UserInterface object and used to store data relevant to a customer (University of Essex Online, 2021e). The

class's methods provide a way to access and manipulate a user's information (Phillips, 2018; University of Essex Online, 2021f).

2. Employee class instantiations are also made by the UserInterface object and store relevant employee details (University of Essex Online, 2021e).
3. A UserInterface object will be instantiated in the main code body and used to store user objects with their respective security details, so as to provide validation and authorisation features (University of Essex Online, 2021g). Furthermore, the object will be used to interface system users to further system functionality in the Store and Warehouse objects (University of Essex Online, 2021d).
4. The Store class will be created in the main code body and contains details relevant to a normal real-world store, as well as Warehouse objects allowing for product details to be displayed, products to be searched for, and an order to be placed (University of Essex Online, 2021f; Fowler, N.D). Additional Stores may be created by a programmer and linked to the UserInterface if the system needs to be expanded (Jasminder, 2014).
5. Warehouse instantiations will also be made in the codes main body, wherein ProductType objects will be created and stored, and relevant locations in the warehouse appended (Jasminder, 2014; University of Essex Online, 2021d).
6. An Order object will contain relevant details pertaining to an order in progress, and provide the functionality to take a user through the steps required to purchase and item (Phillips, 2018).
7. The ProductType class will be used to create objects that represent any item available for purchase in the store. An object will store all the details relevant to

a specific item model, as well as a list of all the available related serial numbers so that a stock count can be calculated, and units tracked (Phillips, 2018).

Class instantiations share relationships to produce functional flows, and potentially cascade data abstractions (Folwer, N.D). Relationships were created around the principle of loosely connected objects are best (Project Prototype, 2014). A brief object relationship summary of Figure 1 follows:

1. Employee objects inherit the features of Customer objects (University of Essex Online, 2021h). Instead of creating a third class of “Person”, and making Employees and Customers inherit from there, it is assumed that all Employees will be Customers, thereby limiting the number of required classes (University of Essex Online, 2021h).
2. Customer and Employee objects will be created and stored in the Userinterface instantiation, thereby making them owned by and reliant for life on the UserInterface object, hence producing a composite relationship with it (Jasminder, 2014; Fowler, N.D; Ambler, 2003).
3. Store objects will be created outside the relevant UserInterface object, but allocated to it, therefore producing an aggregation relationship (Jasminder, 2014; Fowler, N.D; Ambler, 2003).
4. Warehouses will be created outside all other objects, but may be referenced by multiple stores, forming associations (Jasminder, 2014; Fowler, N.D; Ambler, 2003). This allows stores to share warehouses, which may be the case in reality.
5. Orders will be created by Stores and placed in Warehouses, therefore making a dependency connection between Store and Order objects (Jasminder, 2014; Fowler, N.D; Ambler, 2003).

6. Warehouses will be composed of Order and ProdutType objects, and destroy them when required, hence a composition relationship type will be maintained (Jasminder, 2014; University of Essex Online, 2021d).

Figures 2, 3, and 4 flesh out different aspects of placing an order in the system (Fowler, N.D; Ambler, 2003).

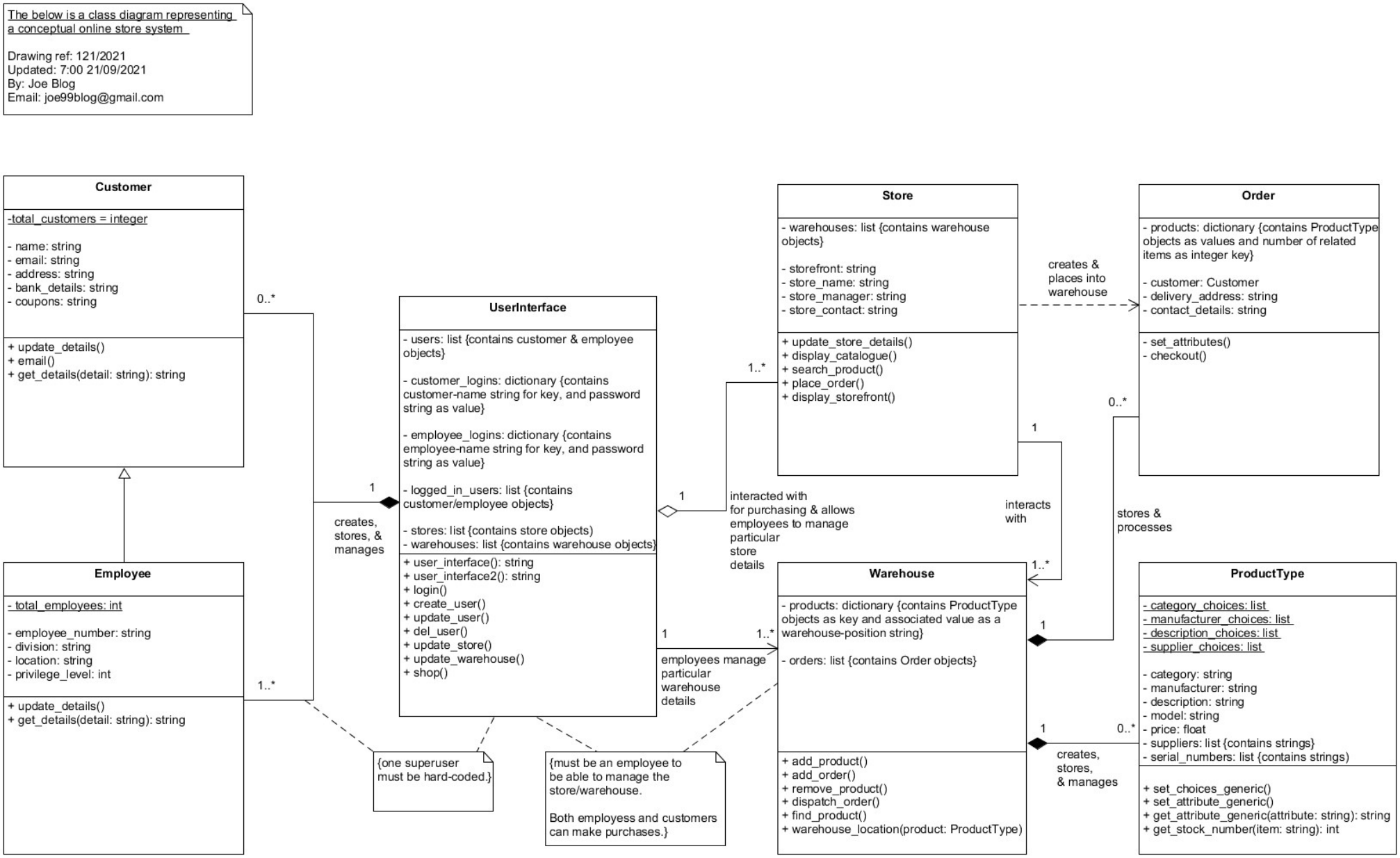

Figure 1 - Class Diagram

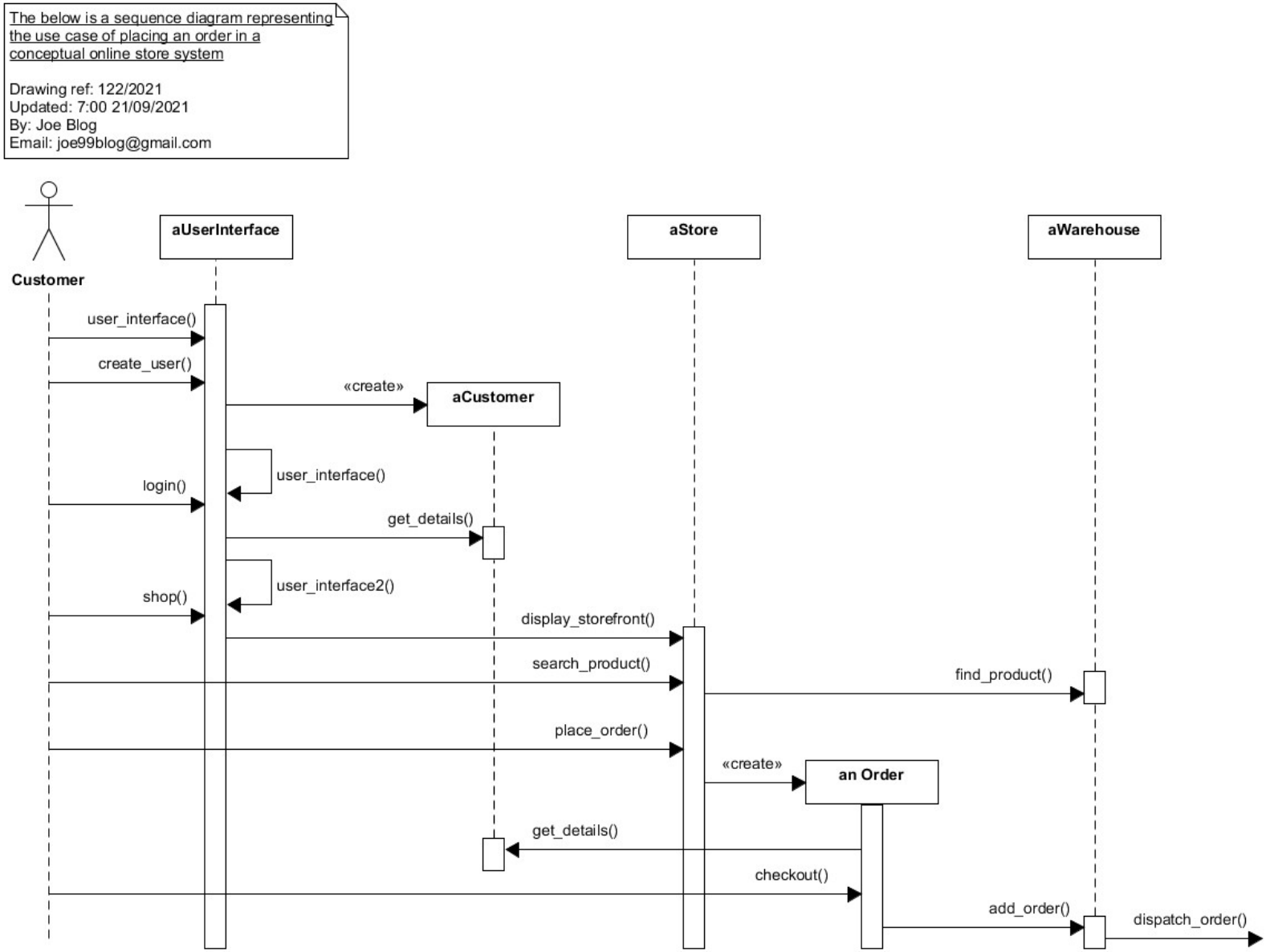

Figure 2 – Sequence Diagram

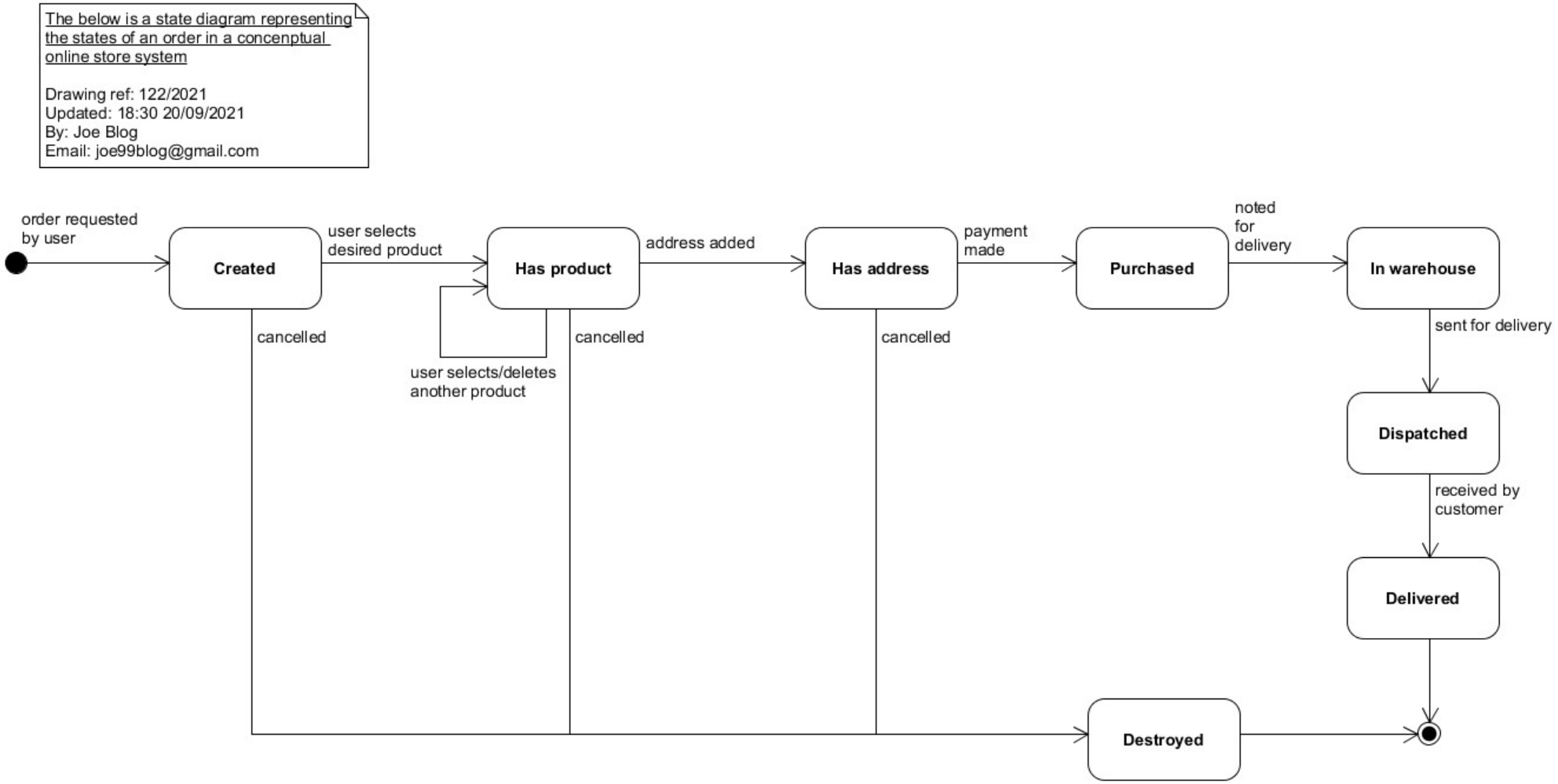

Figure 3 – State Diagram

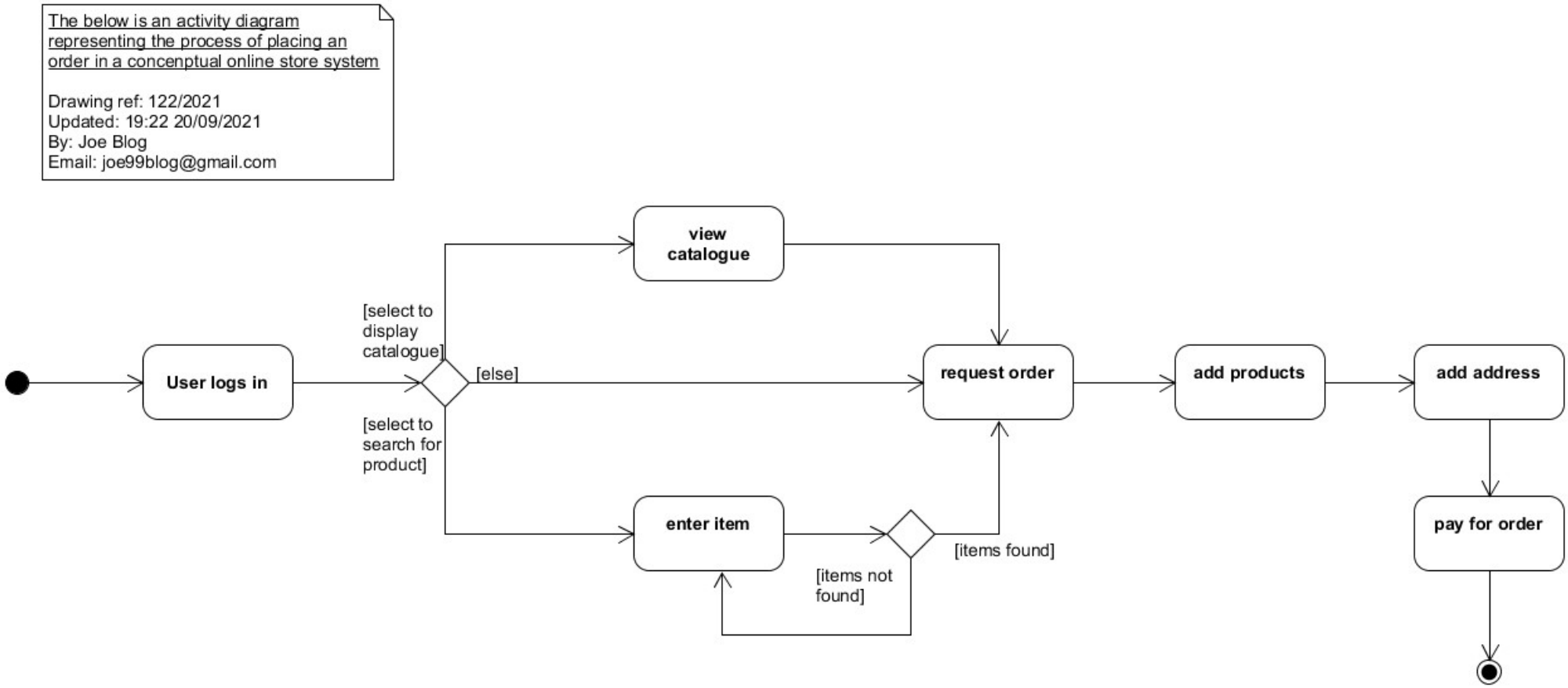

Figure 4 - Activity Diagram

References:

Ambler, S. (2003) *The Elements of UML Style*. Cambridge: Cambridge University Press.

Bourgeois, D. (2014) *Information Systems for Business and Beyond*. Washington: The Saylor Academy.

Brookshear, J., Brylow, D. (2018) *Computer Science: An Overview*. 13th ed. London: Pearson.

Fowler, M. (N.D*) UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed. Massachusetts: Booch Jacobson Rumbaugh.

Jasminder. (2014) Concept of Dependency, Generalization, Association, Aggregation, Composition in Object Oriented programming. Available from: https://dotnetfreakblog.wordpress.com/2014/01/11/concept-of-dependency-generalization-association-aggregation-composition-in-object-oriented-programming/ [Accessed 19 September 2021].

Nieles, M., Dempsey, Kelly., Pillitteri, V. (2017) *An Introduction to Information Security*. Revision 1. United States of America: National Institute of Standards and Technology. Available from: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-12r1.pdf [Accessed 18 September 2021].

Phillips, D. (2018) *Python 3 Object-Oriented Programming*. 3rd Edition. Birmingham: Packt Publishing.

Prototype Project. (2014) UML Tutorial: Association, Aggregation, Composition, Dependency, Generalization, and realization. Available from: https://www.youtube.com/watch?v=6cQs1JkUrY0 [Accessed 18 September 2021].

Sommerville, I. (2016) *Software Engineering*. 10th ed. Essex: Pearson Education Limited.

University of Essex Online. (2021a) *Introduction to Information Systems* [Lecturecast]. OOIS_PCOM7E AUGUST 2021 Object-oriented Information Systems. University of Essex Online.

University of Essex Online. (2021b) *Understanding UML* [Lecturecast]. OOIS_PCOM7E AUGUST 2021 Object-oriented Information Systems. University of Essex Online.

University of Essex Online (2021c) *Linux Command Line Interface: Introduction* [Codio Lessons]. LCS_PCOM7E MAY 2021. University of Essex Online

University of Essex Online. (2021d) *Fundamentals of Object-Oriented Design* [Codio Lessons]. OOIS_PCOM7E AUGUST 2021 Object-oriented Information Systems. University of Essex Online.

University of Essex Online. (2021e) *Classes and Objects* [Codio Lessons]. OOIS_PCOM7E AUGUST 2021 Object-oriented Information Systems. University of Essex Online.

University of Essex Online. (2021f) *Class Functions and Methods* [Codio Lessons]. OOIS_PCOM7E AUGUST 2021 Object-oriented Information Systems. University of Essex Online.

University of Essex Online. (2021g) *Advanced Topics in Object Oriented Programming* [Codio Lessons]. OOIS_PCOM7E AUGUST 2021 Object-oriented Information Systems. University of Essex Online.

University of Essex Online. (2021h) *Inheritance* [Codio Lessons]. OOIS_PCOM7E AUGUST 2021 Object-oriented Information Systems. University of Essex Online.