



slington college
(इस्लिङ्टन कलेज)

Module Code & Module Title
CS5002NI SOFTWARE ENGINEERING

Assessment Weightage & Type
35% Individual Coursework

Year and Semester
2021-22 Spring

Student Name: Mubson Karki

London Met ID: 20048923

College ID: np01cp4s210231

Assignment Due Date: 9th May 2022

Assignment Submission Date: 9th May 2022

Title (Where Required):

Word Count (Where Required):

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Contents

Table of figures	2
Table of tables	3
1. Introduction.....	4
2. Gantt Chart.....	5
3. Use Case Model	6
3.1. Use Case Diagram.....	7
3.2. High-level level Use Case Description	8
3.3. Expanded Use Case Description	11
4. Communication Diagram	14
4.1. Collaboration Diagram.....	14
4.2. Sequence Diagram.....	16
5. Class Diagram.....	18
6. Development Process	20
7. Prototype Development	24
8. Conclusion	32
9. References	33

Table of figures

Figure 1: Gantt chart.....	5
Figure 2: Use a case diagram	7
Figure 3: Collaboration Diagram.....	15
Figure 4: Sequence Diagram	17
Figure 5: Class Diagram	19
Figure 6: Prototype for start page of T-14	24
Figure 7: Prototype for sign-in in T-14	24
Figure 8: Prototype for registering in T-14	25
Figure 9: Prototype for the home page of T-14 after successful login	25
Figure 10: Prototype for left side navigation slider feature of T-14	26
Figure 11: Prototype for viewing the accessories' details before purchasing.....	26
Figure 12: Prototype for viewing expanded product details after selecting a kit ..	27
Figure 13: Prototype for displaying alert message with the price to be paid for the purchase.....	27
Figure 14: Prototype for choosing any one of the online payment options	28
Figure 15: Prototype for choosing any one of the exam types	28
Figure 16: Prototype for watching videos as an exam	29
Figure 17: Prototype for giving the mock test as an exam	29
Figure 18: Prototype for viewing the report of a user	30
Figure 19: Prototype for the requirement of staff authentication to create exam papers or videos.....	30
Figure 20: Prototype for creating exam questions and videos by staff	31
Figure 21: Prototype for viewing notices uploaded by admin	31

Table of tables

Table 1: Expanded use case description for purchase kits	11
Table 2: Expanded use case description for creating exams (manually typing in the system)	12
Table 3: Expanded use case description for creating exam (adding the saved files from desktop)	13
Table 4: Domain classes of purchase kits use case	14

1. Introduction

Object-oriented Analysis and Design, OOAD in short is a software engineering technique in which a system is represented as a group of interaction objects. Each object is distinguished by its class, state, and behavior, and represents some entity of interest in the system being represented. Using this methodology, a system is to be designed that meets the requirements of this coursework. This system should focus on the online management of T-14 academy's business along with the different football training programs.

T-14 Training Academy is a new project launched by a group of former national team football players that specializes in football, training programs, and the business of football accessories. Also, this academy has thought of taking exams in the form of watching videos and conducting mock exams for developing the football IQ among players. This training academy wants to provide training in two different levels – basic and intermediate. The intermediate training can only be taken by those who excel in all written and physical tests. Lastly, they want to take their business of football accessories into the online market and provide a special discount for the academy's members. So, for all these requirements, along with the online examination, a system is to be developed using the OOAD technique.

For the development of this system, Scrum, which is an Agile framework focusing on cross-functional teamwork, accountability, and iteration is used along with the Unified Modeling Process (UML). For effective time management and meeting the goal in the specified time, a Gantt chart presenting the period of completion is created using the Dynamic System Development Method (DSDM) principle. This chart is implemented for an entire month where a use case is developed in the initial phase for gathering the requirements of the development process.

The entire process of the development of a system is focused on an object-oriented. The collaboration diagram, sequence diagram, and class diagrams are created to suit the coursework's criteria. At the culmination of the coursework, a software prototype is also created to sketch the look of the T-14 Training Academy system.

2. Gantt Chart

A Gantt chart is one of the most popular and practical ways of displaying activities against the time. In software development, project schedules are tracked using this chart. Additional information on the various tasks or phases of the project, such as how the tasks relate to one another, how far each task has advanced, what resources are being used for each task, and so on, is useful for this. In our task for creating a system for the T-14 training academy, the DSDM methodology is used for separating different tasks over a time frame of 45 days as shown in the figure below:

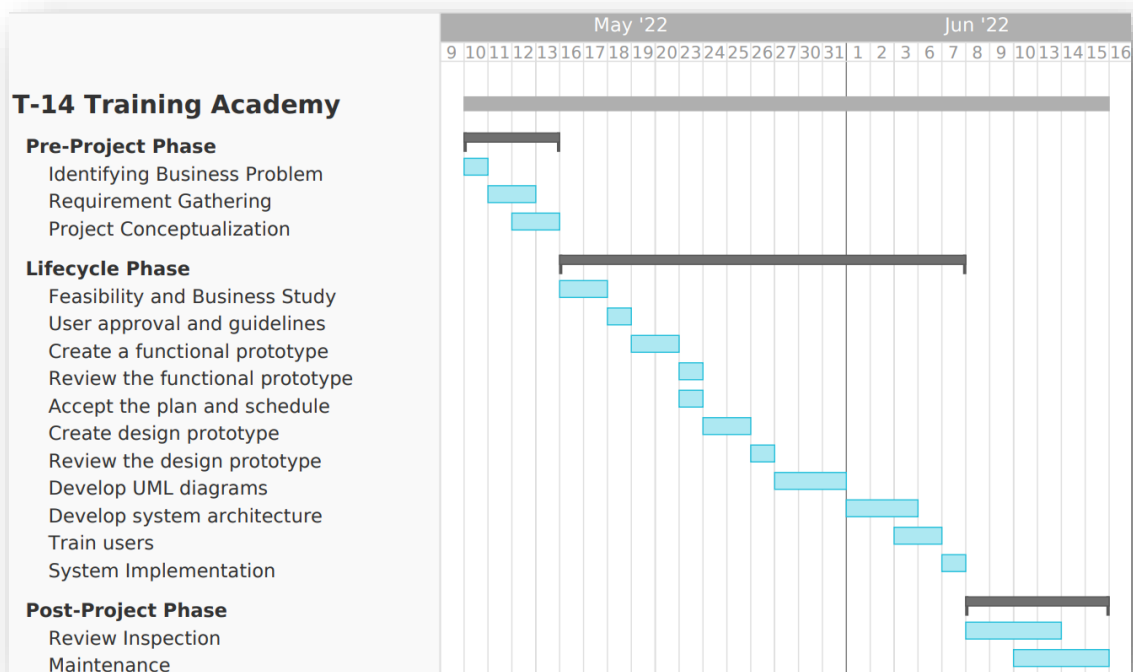


Figure 1: Gantt chart

Here, the above Gantt chart for the T-14 training academy is created from the “TeamGantt” website that excels at presenting the chart simplistically and understandably. A list of activities is on the left side of the chart, and a suitable time scale is along the top. Each action is represented by a bar, whose location and length indicate the activity’s start, duration, and end dates. From this chart, the system development life-

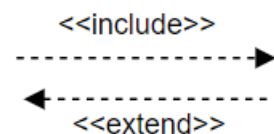
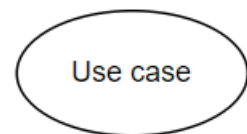
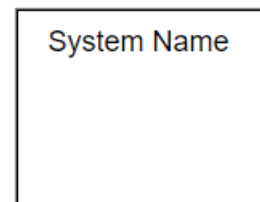
cycle becomes much easier by understanding how long each activity is scheduled to last, where activities overlap with others, efficient planning, and so on.

3. Use Case Model

A use-case model is a model that represents the interaction of roles (also known as “actors” in UML) and systems for solving a problem. Thus, it specifies the user’s goal, the system’s interactions with the user, and the system’s behavior required to achieve these goals.

When creating a use-case model, one should consider four components:

- System: A system can be considered as an application or software itself where interaction between the actors and the functionalities (use case) occurs. It is represented by a rectangular box as shown alongside:
- Use Case: The use case is like a function that describes how actors interact with a system. It is typically introduced by the user to fulfill the goals and objectives of the actions. It is represented by an oval shape as shown alongside:
- Actors: Actors are persons who are involved with the system and are defined by their roles. An actor can be a human or another external system. It is represented by a human symbol as shown below:
- Relations: Relations are the links connecting the model elements – use case and actors. It defines the structure and behavior between model elements, adding semantics to the model. It is represented by an arrow as shown below:



3.1. Use Case Diagram

Use case diagrams are used to collect a system's needs, covering both internal and external influences. The majority of these requirements are design-related. As a result, use cases are generated and actors are identified when a system is analyzed to gather its functionalities (Waykar, 2015).

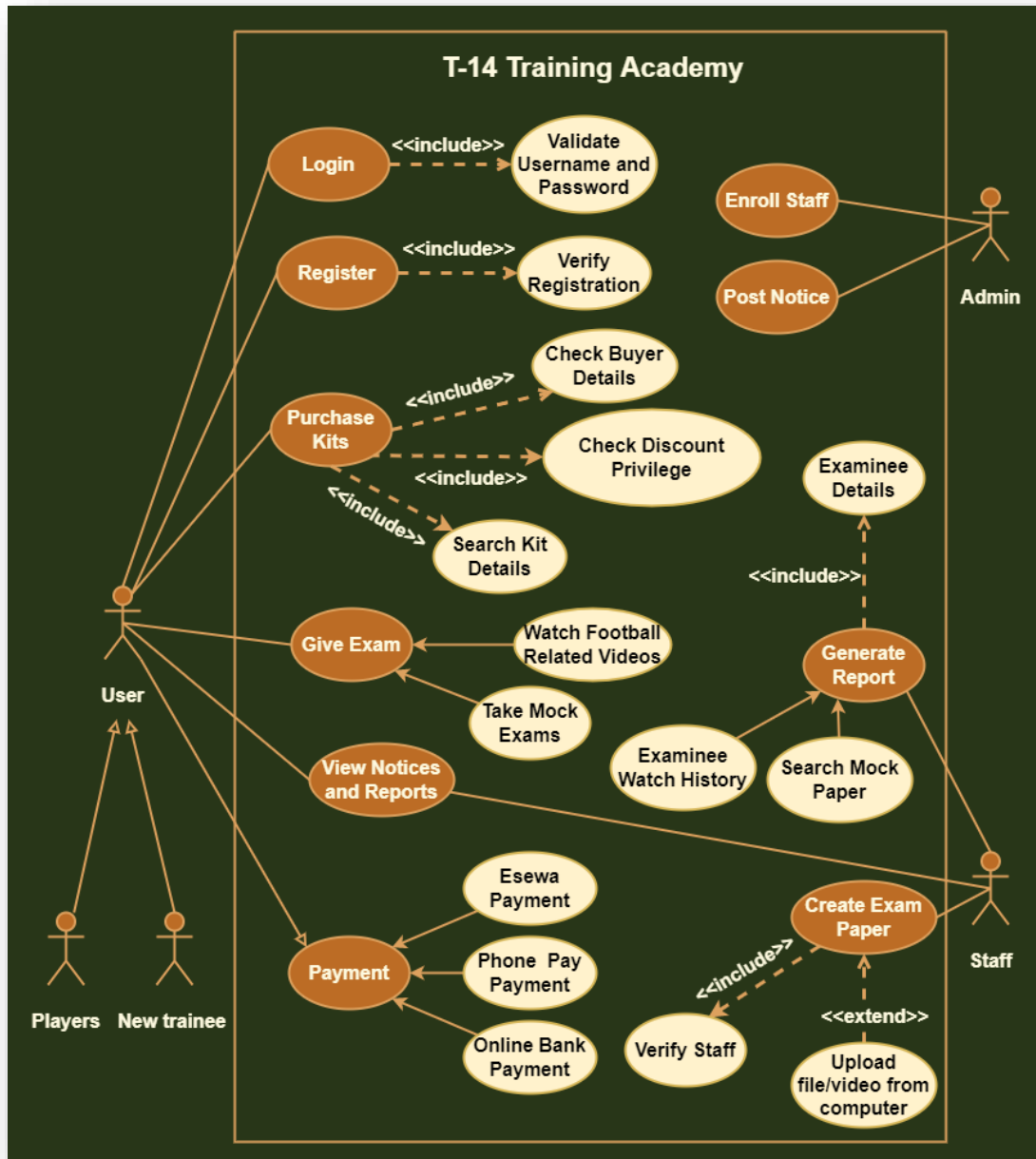


Figure 2: Use a case diagram

In this coursework, the use case model is used to visualize a system's functional requirements. It has helped in providing an overview of the roles of every component in the system along with defining the roles of user, staff, and admin. From the above diagram, one can easily understand that generalizations and associations are being used in different use cases like "Register" is completed once "Verify Registration" is executed.

3.2. High-level level Use Case Description

3.2.1.

Use Case: Login

Actor: User

Description: The user can get access to the features of the T-14 Training Academy's system by providing a valid username and password that is provided while registering.

3.2.2.

Use Case: Register

Actor: User

Description: Anyone who wants to access the feature needs to register for the membership by providing their first name, last name, phone number, photo, username, and password that is used while logging in to the system.

3.2.3.

Use Case: Purchase Kits

Actor: User

Description: After logging in, the user can choose to purchase the kits where they can view the list of kit details. The system itself verifies if the user is a player or not and then provides the user with the discounted price (discount for players). Lastly, the user has to confirm if they want to purchase the kits for the allocated price or not.

3.2.4.

Use Case: Give Exam

Actor: User

Description: A user can give the exam for advancing to the intermediate training. Exams can be given in two ways – by watching related videos and by giving mock exams.

3.2.5.

Use Case: View Notices and Reports

Actor: User, Staff

Description: Admin daily posts different notices which can be seen by staff as well as users. Also, the progress reports of a player can be seen by the user.

3.2.6.

Use Case: Payment

Actor: User

Description: For a successful purchase, payment is required. It can be done in any of the three ways – Esewa, Phone Pay, or Mobile Banking.

3.2.7.

Use Case: Create Exam Paper

Actor: Staff

Description: For users to give exams, the staff creates exam papers in the system itself. They can also choose to upload the exam paper or videos that are saved on their computer. Exam papers can be created only after completing staff authentication.

3.2.8.

Use Case: Generate Report

Actor: Staff

Description: Staff creates reports of the user based on videos watched, performance is a mock exam and physical exams.

3.2.9.

Use Case: Enroll Staff

Actor: Admin

Description: Admin is responsible for selecting the staff. Admin provides the staff with staff authentication identities (id) and passwords.

3.2.10.

Use Case: Post Notices

Actor: Admin

Description: Admin posts notices daily about different subject matters like training, new accessories, and others.

3.3. Expanded Use Case Description

3.3.1.

Use case: Purchase Kits

Actors: User

Description: The customer gets details of all the available accessories where they select their interest. After the system verifies the user type - player or new trainee, it then calculates the discount and provides the purchase price to the user. The system waits for user confirmation for the purchase for the given price.

Typical Course of Events:

User Action	System Response
1. User chooses to purchase kits.	
	2. System presents all the available kits in the database.
3. Users select the kits of their interest.	
	4. Checks if the purchaser (user) is a new trainee or player.
	5. Calculates the selling price of the kit after discount (for the player) and presents it to the user.
6. Receives a confirmation box with the “Yes”, and “No” options for purchase.	
	7. Redirects to payment.

Table 1: Expanded use case description for purchase kits

Alternative:

Line 6: If the user declines the confirmation of purchase for the generated price, the system redirects the user to the page containing the list of football kits.

3.3.2.

Use case: Create Exam Paper

Actors: Staff

Description: Staff can only create exam papers after they have entered the admin given id and password. Upon successful staff authorization, the system lets the staff create the question paper. If the user has the question/video on their computer, they can also insert the file directly from their computer.

Typical Course of Events (if staff wishes to create an exam paper):

Staff Action	System Response
1. Staff chooses to create paper.	
	2. Request authentication as staff.
3. Provides ID and password of the staff given by admin.	
	4. Validates the ID and password.
	5. Provides an in-built writing tool to create exam papers.
6. Writes down the exam paper and click on the "Upload" button.	
	7. Adds exam paper to the database.

Table 2: Expanded use case description for creating exams (manually typing in the system)

Alternative:

Line 4: Invalid ID and password can be given 3 consecutive times and then the system bans the staff for 24 hours to re-enter the correct ID and password.

Typical Course of Events (if staff wishes to upload a file/video from their computer):

Staff Action	System Response
1. Staff chooses to create paper.	
	2. Request authentication as staff.
3. Provides ID and password of the staff given by admin.	
	4. Validates the ID and password.
	5. Provides an in-built writing tool to create exam papers.
6. Clicks on the “+” button.	
	7. Desktop location browser is opened.
8. User selects a file from their desktop.	
	9. Adds a file/video to the database.

Table 3: Expanded use case description for creating exam (adding the saved files from desktop)

Alternative:

Line 4: Invalid ID and password can be given 3 consecutive times and then the system bans the staff for 24 hours to re-enter the correct ID and password.

Line 8: If the file type is not supported or the file size is too long, it displays an error and re-does line 5.

4. Communication Diagram

4.1. Collaboration Diagram

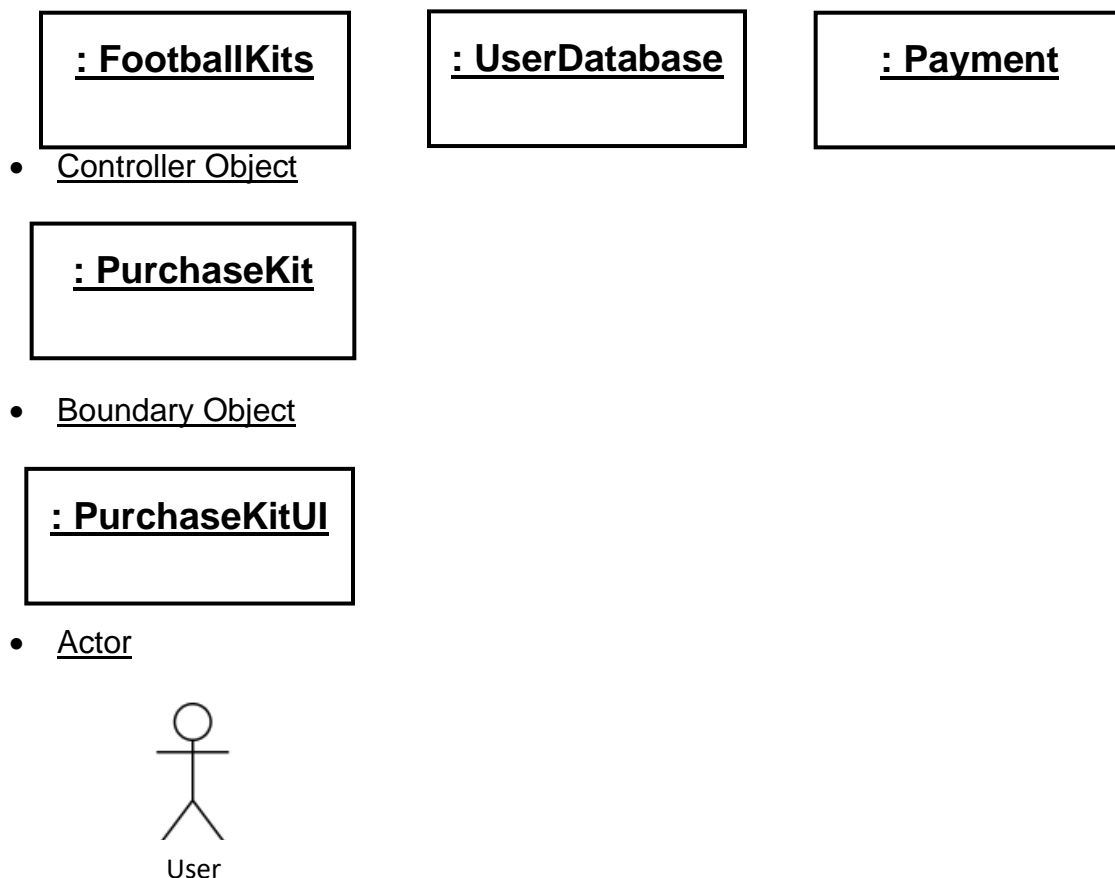
A collaboration diagram is a diagram that shows how objects interact to conduct a particular application behavior or part of a use case. So, a collaboration diagram helps in a precise understanding of the different aspects and abstraction levels of the object interaction (Heckel & Sauer, 2001).

For the “Purchase Kits” use case, collaboration is created that contains:

- Domain Class

Use case	Domain Classes
Purchase Kits	User, Football Kits, Payment

Table 4: Domain classes of purchase kits use case



These different domain classes, boundary object, controller object, and actors are linked together. And the flow of messages between these objects is shown by a directional arrow indicating the direction of the flow of messages along with a method name in each of the arrows as shown in the figure below:

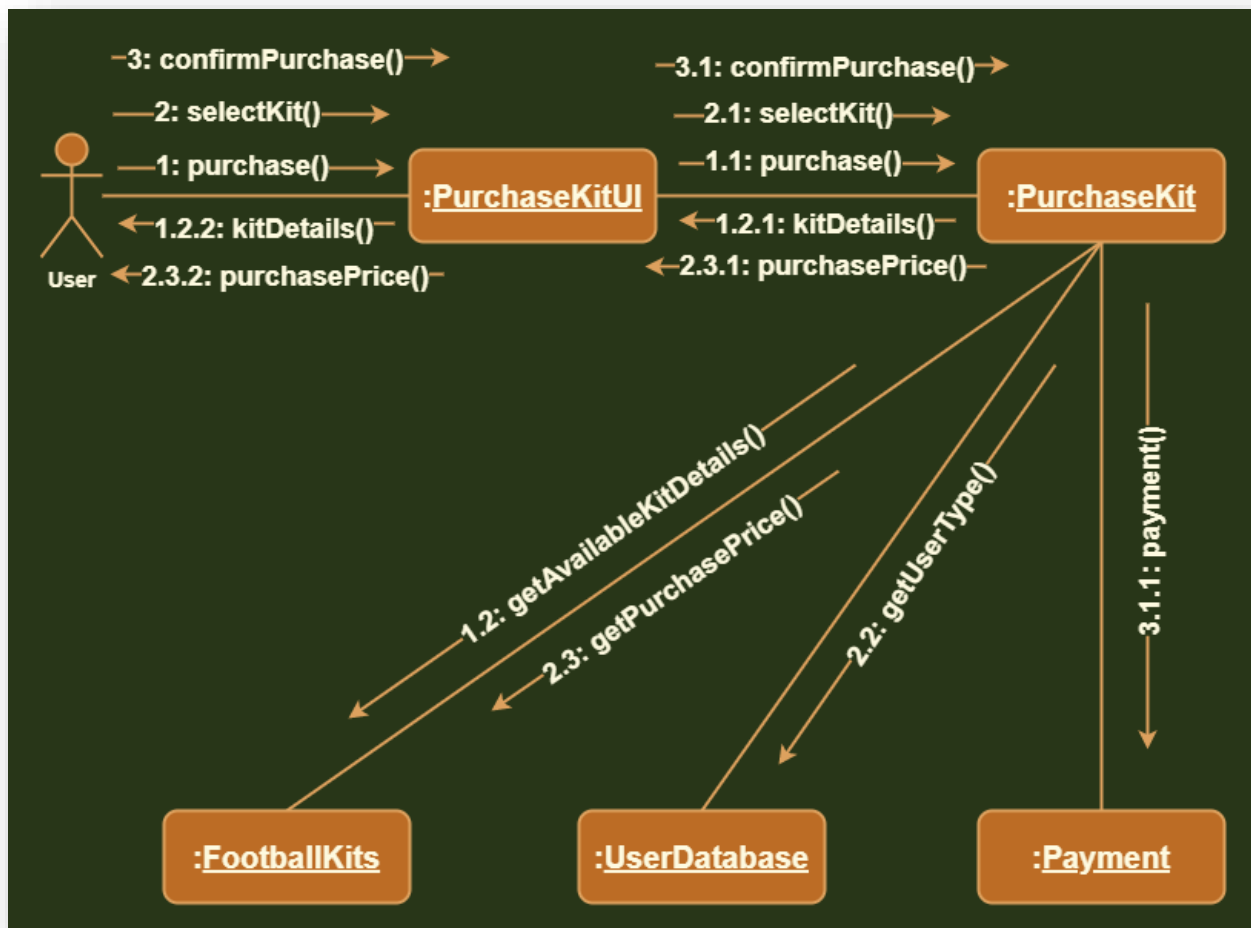


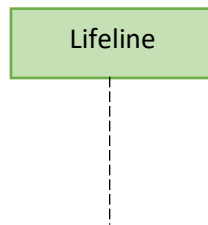
Figure 3: Collaboration Diagram

The above collaboration diagram is created by researching the structural elements (class, roles, objects, and subsystems) that are showing the functionality of collaborations. From the Use-case diagram and expanded Use-case description, the context of interactions like system, subsystem, use case, and operation are chosen for creating this diagram.

4.2. Sequence Diagram

A sequence diagram is a communication diagram that depicts the communication between any two lifelines as a time-ordered series of events, with these lifelines participating in the run time. It is also termed an event diagram which assists in visualizing a variety of dynamic settings. Some of the components of the sequence diagram are:

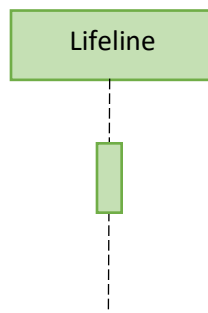
- Lifeline: It is the participant or an object in a sequence diagram.



- Actor: It is an outside entity that interacts with the subject. It represents the role of an entity.



- Activation: It represents the period during which an element operates.



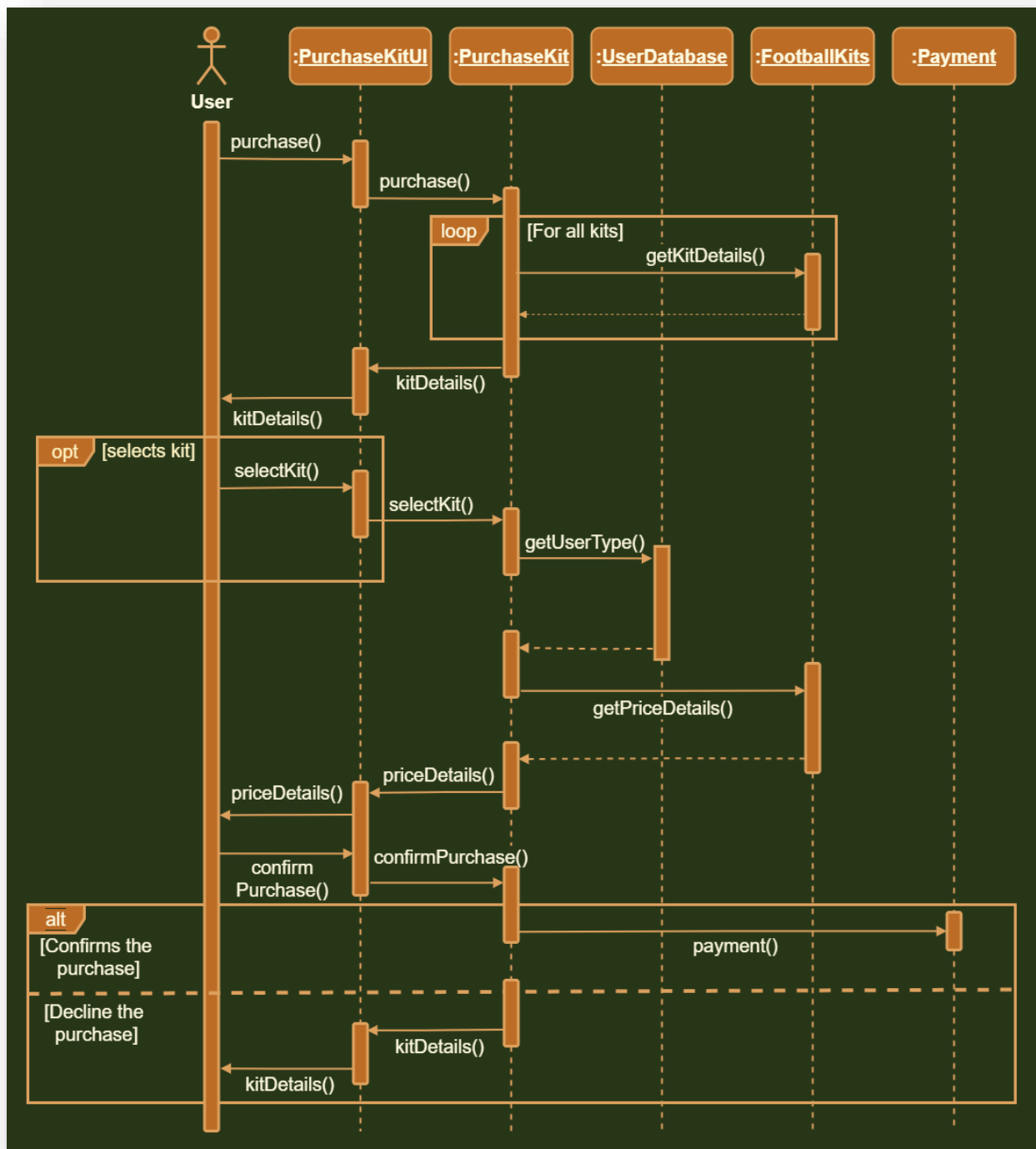


Figure 4: Sequence Diagram

By implementing a time frame in the collaboration diagram, the sequence diagram is created as above. Different fragments are used in this diagram that helps to present a

clear-cut idea about statements, loops, and actions. Thus, this diagram has helped in defining how and in what order the object in the system works.

5. Class Diagram

The static view of an application is depicted by the class diagram. Different types of objects in the system as well as their relationships are depicted in this class diagram. The class diagram includes different components like:

- **Dependency:** A semantic link between two or more classes in which changes in one class result in a change in another.



- **Generalization:** A relationship between a parent class and a child class where the child class inherits from the parent class.



- **Association:** A connection between two or more objects that is either static or physical. It shows the number of objects in the relationship.



- **Aggregation:** A subset of association is an aggregation that represents a relationship. The child class in this case can exist independently of its parent class. Represents a part of the relationship.



- **Composition:** Depicts the relationship between a parent and their child, implying that if one component is removed, the other is also removed. Represents a whole-part relation.



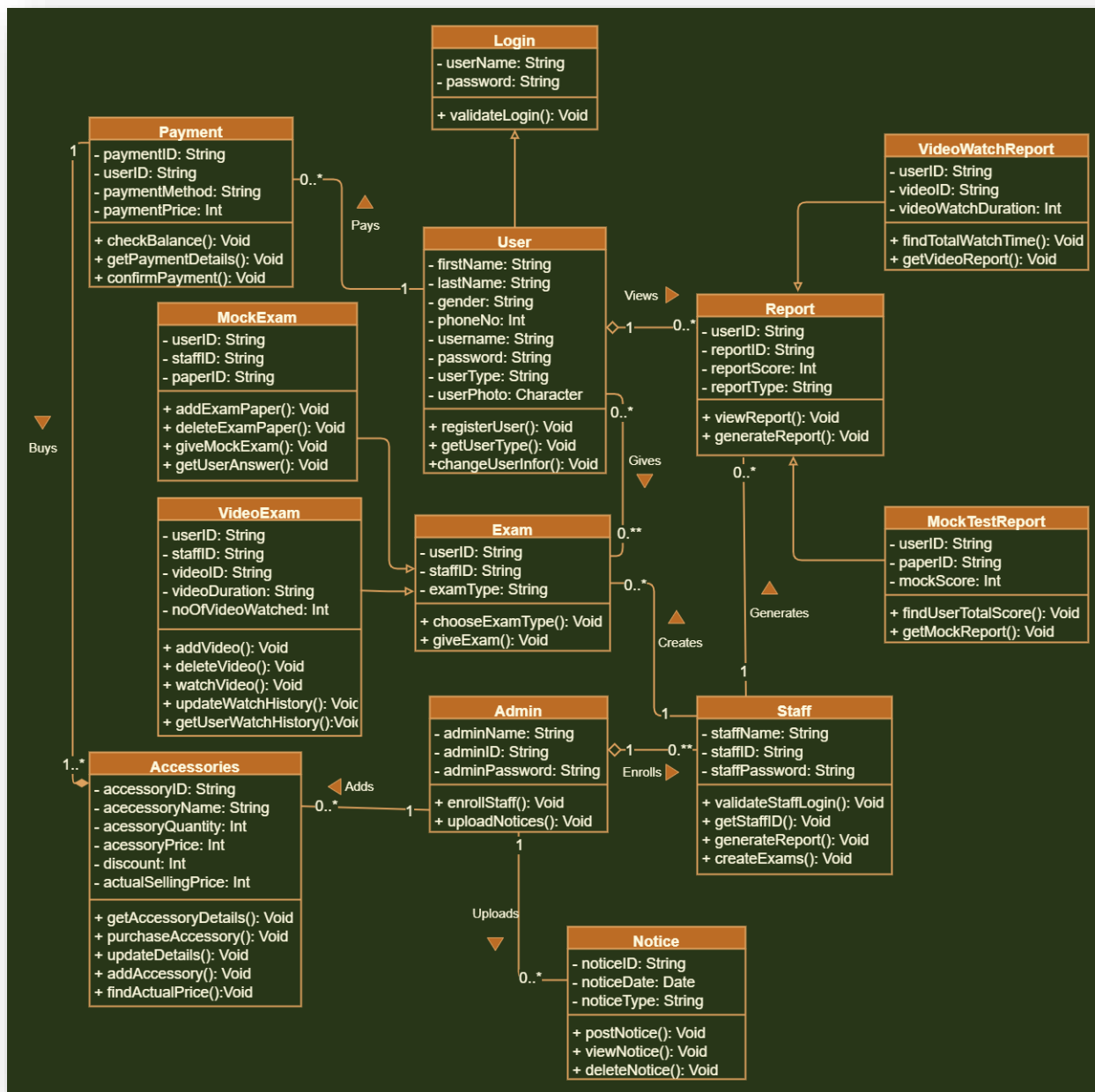


Figure 5: Class Diagram

Here, the different classes by studying high-level descriptions. These are the classes that would be created for developing the codes. It would help to represent, explain, and document numerous parts of a system, as well as to create executable software. By dividing class names, attributes, and methods into separate compartments, class diagrams help in software development.

6. Development Process

The prime requirement of the coursework is to use Object-Oriented Analysis and Design to develop a system for a football training academy. The DSDM principle along with UML has been used for completing this task. Unified Modeling Language (UML) is a modeling language that is used for software engineering. It has become an industry-standard technique for visualizing software using diagrams. So, UML allows teams to see how a project is or will work through the use of diagrams. Likewise, Dynamic Systems Development Method (DSDM) is one of the first agile project management approaches, and while it was designed with software development in mind, it can be used for a variety of projects too.

The DSDM technique, like other agile methodologies, is built on an incremental approach with frequent releases and testing (Gisclard-Biondi, 2021). Under this methodology, there are four main project phases:

- **Feasibility and Business Study:** During this phase, the project is analyzed in detail and some important decisions are made as to whether to take over the project or if the project is possible to complete. In this phase, application and infrastructure diagrams are finalized.
- **Functional Model Iteration:** The prototype models are studied for their quality and improvement possibilities. Activities like prototype presentation to the users, gaining user feedback, and others are done. At the end of this phase, a functional model with an analysis model and some significant functionality is to be developed.
- **Design and Build Iteration:** The different software components that were created during the previous phase are refined until they meet a satisfactory standard and are combined into one system in this step. This phase results in a tested system that is ready for implementation,
- **Implementation:** It involves transitioning from the development to the operational environment, which includes training users and delivering the system. Activities like a gathering of comments and reviews from end-users, and supply of proper solutions to end-users with all of their needs met are done.

➤ **Designing part of the development:**

Following the DSDM technique, different step-by-step procedures are done which will make the development process considerably more efficient and reliable. Now, we can begin the development of the overall system. The design stage of the SDLC is where the product is conceptualized. Designers are assigned the task to construct the project's entire structure and deliver the final prototype for usage in the software development process following the Creational Design Pattern. It is a design pattern that defines class instantiation. This could help in the use of inheritances and object-creation during the development of the system. Components of this pattern are:

- Abstract Factory: Creates an instance of various classes from different families.
- Builder: Distinguishes the construction of a thing from its representation.
- Factory Method: Instances of numerous derived classes are created.
- Object Pool: Helps save money and resources by recycling objects that are no longer in use.
- Prototype: A fully initialized copy or clonable instance.
- Singleton: There can only be one instance of a class.

➤ **Selecting an architectural pattern:**

After designing, developers get their coding parts started. Every programmer has his list of software development responsibilities for which he is accountable. This is the most time-consuming procedure. Then for the architectural pattern, Model View Controller (MVC) is used that helps in implementing user interfaces, data, and controlling logic. This MVC pattern would help in providing a better division of labor and improved maintenance. This MVC architecture is implemented through the use of AngularJS web frameworks. It is based on JavaScript and can be used in HTML pages with a <script> tag. This framework will use directives to expand HTML attributes and expressions to tie data to HTML in our system development process.

➤ **Testing in the development phase:**

As the system development follows the DSDM methodology whose one of the principles is to assure quality, testing is the most important in the development process. All types of functional testing, including integration testing, unit testing, system testing, acceptance testing, and non-functional testing are carried out. QA engineers inspect the code that developers have created. Different frameworks and test cases are employed to find out if the system has any bugs which are then reported to developers, who then solve them. The different testings that are carried out in the SDLC are:

- **Unit Testing:** The smallest aspect of software designs is tested. Single units or collections of interconnected components are tested.
- **Integration Testing:** The software modules are integrated logically and tested as a group. In this testing, a group of components is combined to produce output.
- **Regression Testing:** Throughout development, different modules are added to the program. So, when adding a new module, the whole component could be affected. This testing ensures that the complete program would not get hampered by adding new modules to the program.
- **Smoke Testing:** This test ensures that the program under test is ready or stable for further evaluation. It's termed a smoke test since it's used to see if it caught fire or produced smoke when turned on for the first time.

Black-box testing is carried out to check/test each of the modules about the requirement provided by the user and its context. In this testing approach, input values are divided into valid and invalid classes and a representative value from each class is selected as test data. The boundary value analysis is also done. Also, the deployment of this system to the user is considered to be done in the beta-testing mode for catching any other bugs.

➤ **Maintenance:**

After the system has been properly tested and handed to the user, it still needs maintenance. The SDLC method used in this system doesn't end when the system reaches the client. A system must be maintainable, meaning it should meet all of the qualities of modifiability, extensibility, reliability, and robustness (FlexBase, 2019). The following mentioned maintenances would be done after the system reaches the client:

- **Adaptive Maintenance:** The system must be designed in such a way that it should be able to adapt to certain technological changes over time.
- **Corrective Maintenance:** Any system failures or bug issues are fixed by performing corrective maintenance. But, during this maintenance, changes to one part should not lead to failure of the other part. So, design pattern, loose coupling, and quality of code are mainly focused on.
- **Perfective Maintenance:** To maintain the quality of the system over time, this maintenance is done. It deals with the concept of continual improvement. Based on the contract with the T-14 training academy, this maintenance is done for the given time and expenses.
- **Preventive Maintenance:** This maintenance helps to avoid the occurrence of errors or malfunctions in the system. This maintenance is done to prevent the deterioration of the system over a long time.

7. Prototype Development

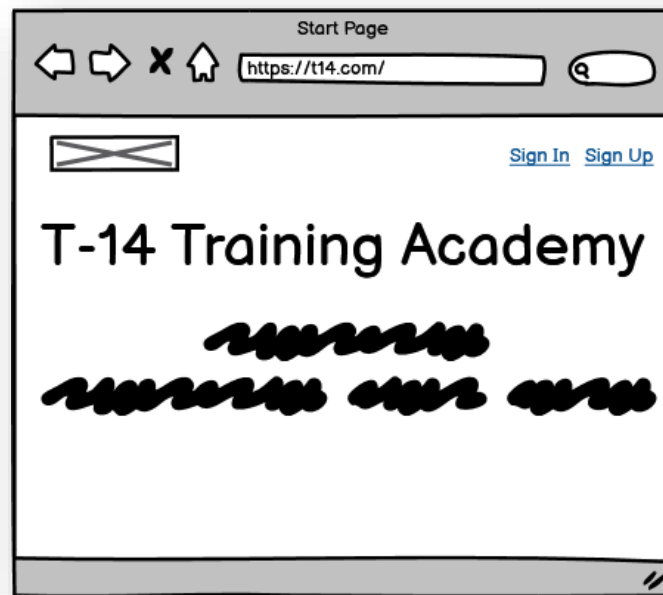
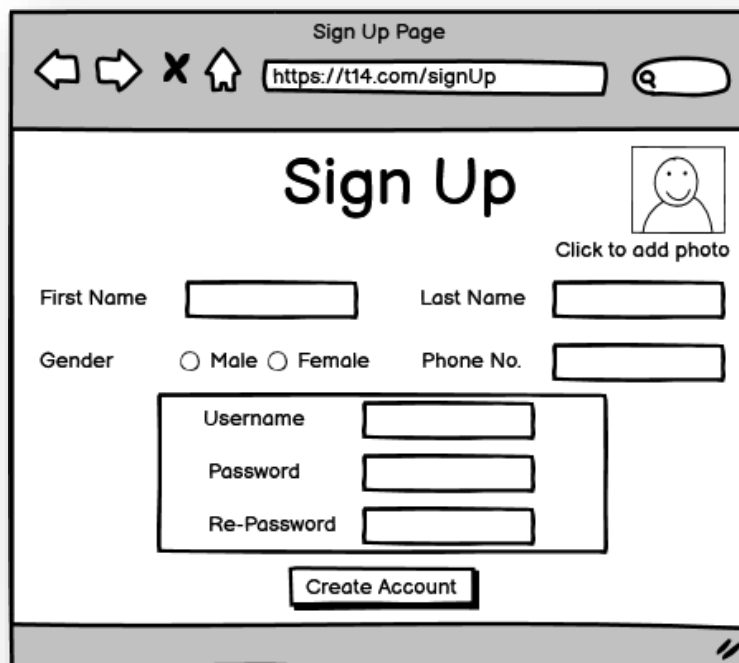


Figure 6: Prototype for start page of T-14

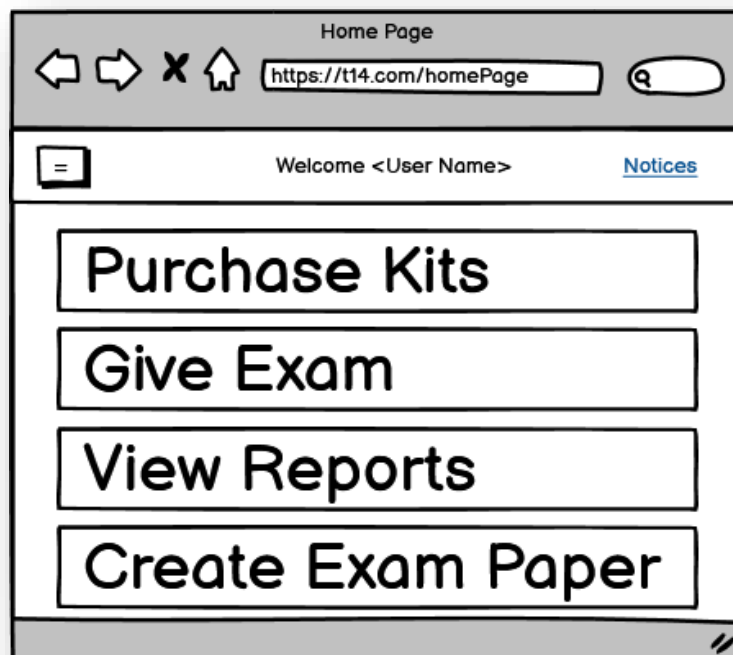


Figure 7: Prototype for sign-in in T-14



A browser window titled "Sign Up Page" with the URL "https://t14.com/signUp". The page features a "Sign Up" heading and a placeholder for a profile picture with the text "Click to add photo". The form includes fields for "First Name", "Last Name", "Gender" (with radio buttons for "Male" and "Female"), "Phone No.", "Username", "Password", and "Re-Password". A "Create Account" button is at the bottom.

Figure 8: Prototype for registering in T-14



A browser window titled "Home Page" with the URL "https://t14.com/homePage". The page shows a "Welcome <User Name>" message and a "Notices" link. Below this is a list of four buttons: "Purchase Kits", "Give Exam", "View Reports", and "Create Exam Paper".

Figure 9: Prototype for the home page of T-14 after successful login

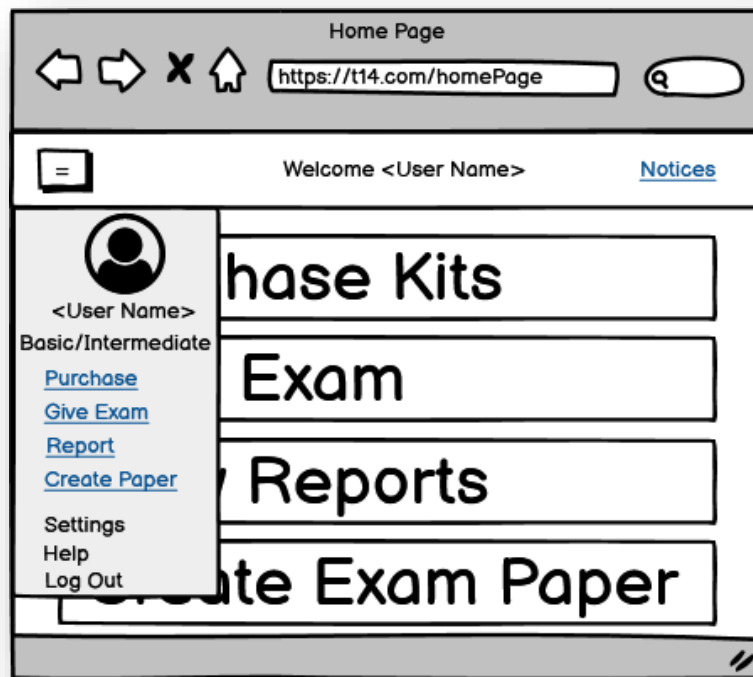


Figure 10: Prototype for left side navigation slider feature of T-14

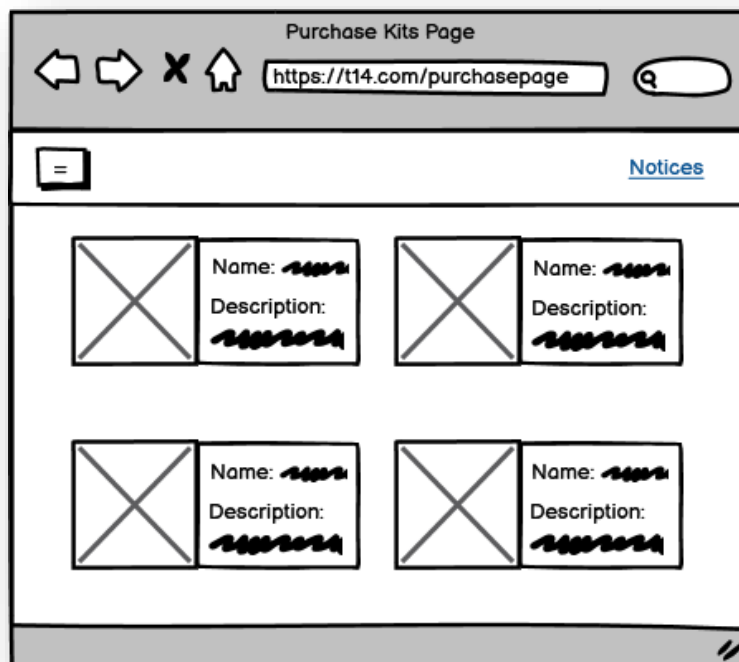


Figure 11: Prototype for viewing the accessories' details before purchasing

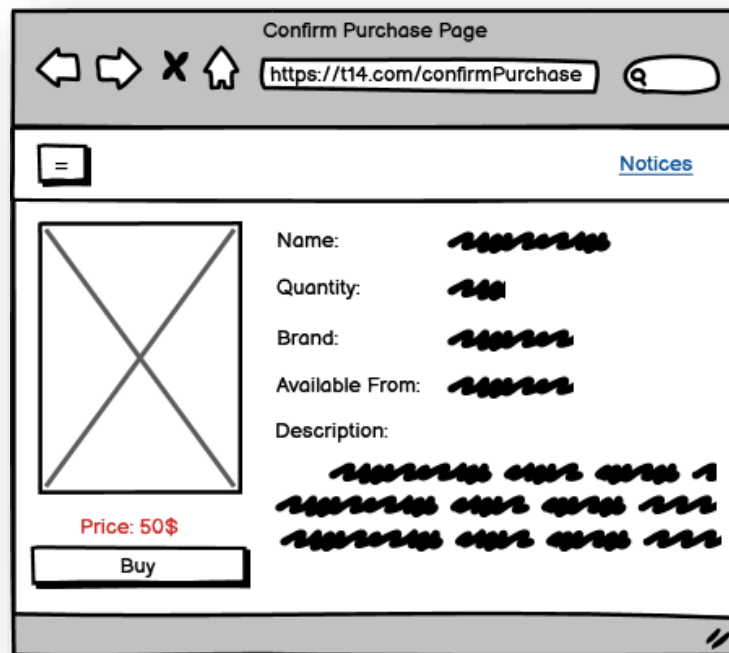


Figure 12: Prototype for viewing expanded product details after selecting a kit

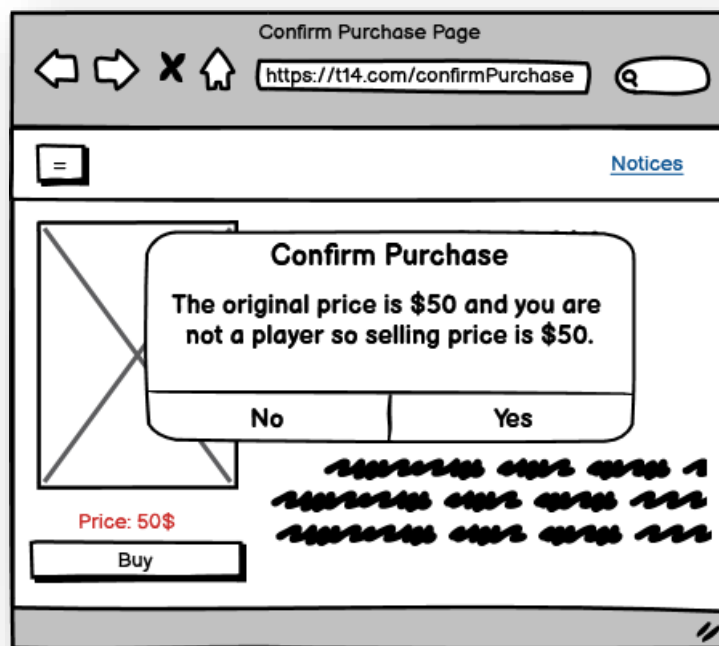


Figure 13: Prototype for displaying alert message with the price to be paid for the purchase

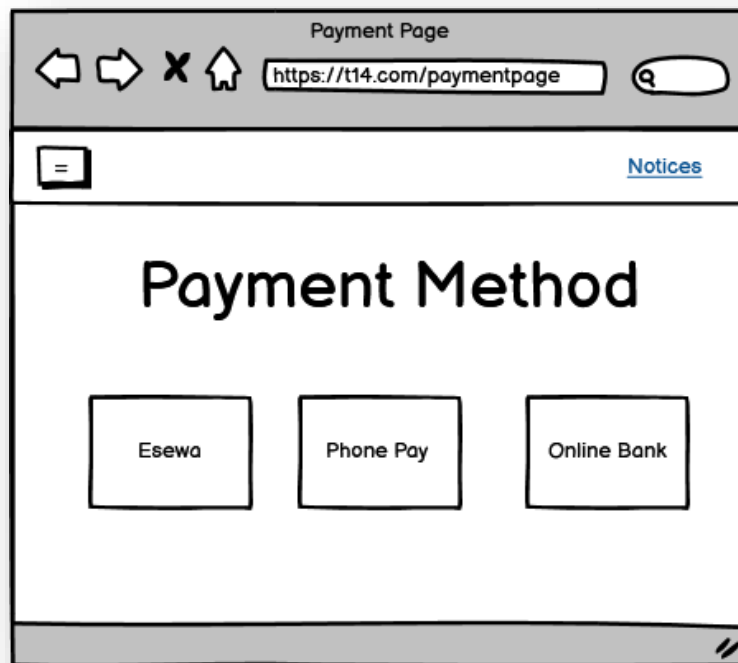


Figure 14: Prototype for choosing any one of the online payment options

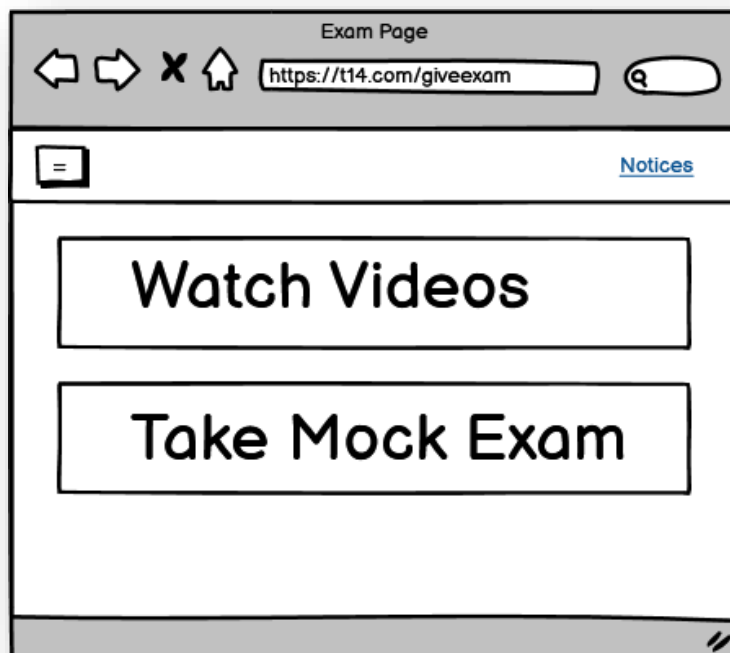


Figure 15: Prototype for choosing any one of the exam types

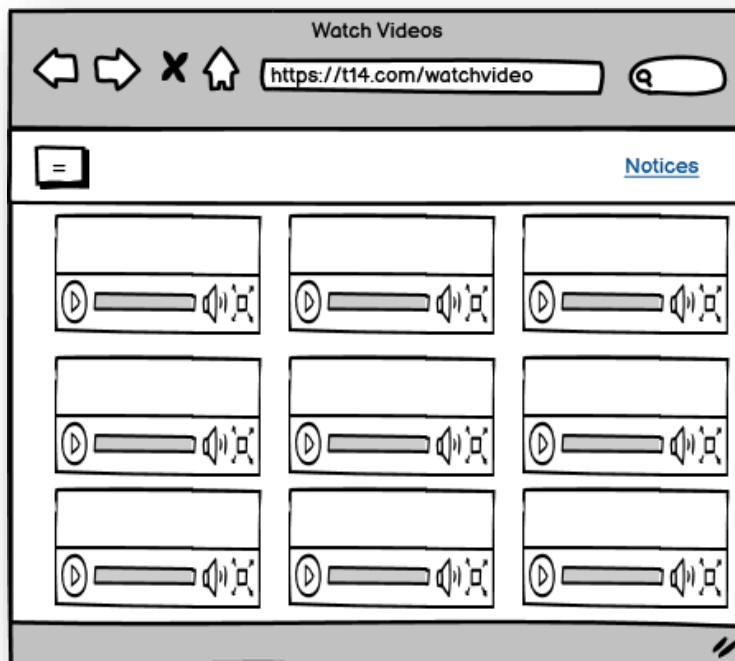


Figure 16: Prototype for watching videos as an exam

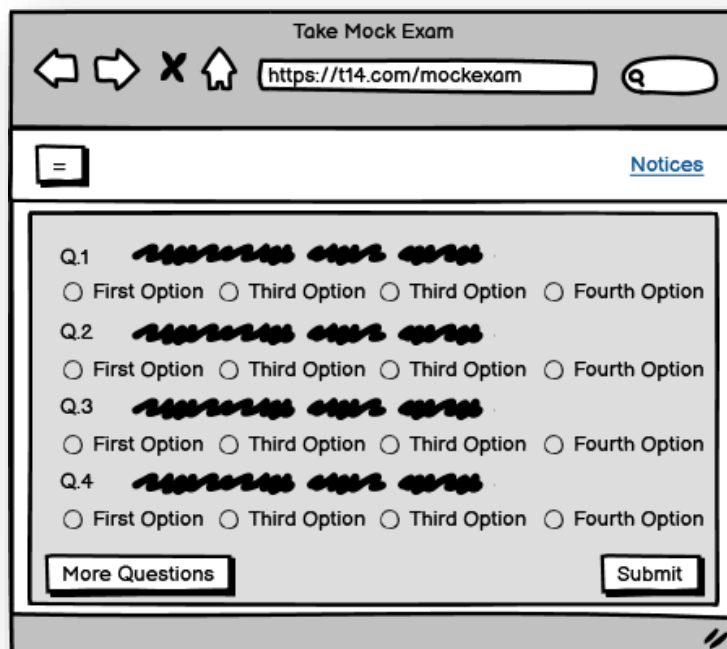


Figure 17: Prototype for giving the mock test as an exam

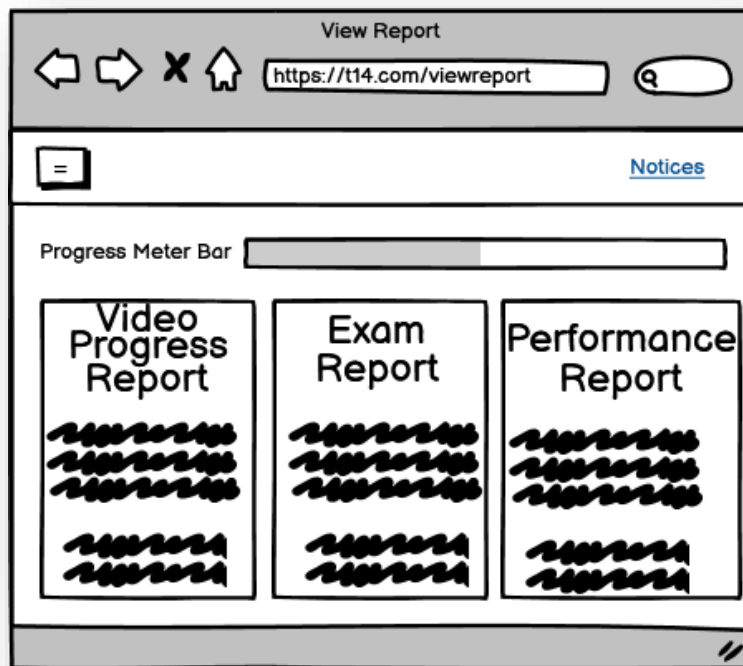


Figure 18: Prototype for viewing the report of a user

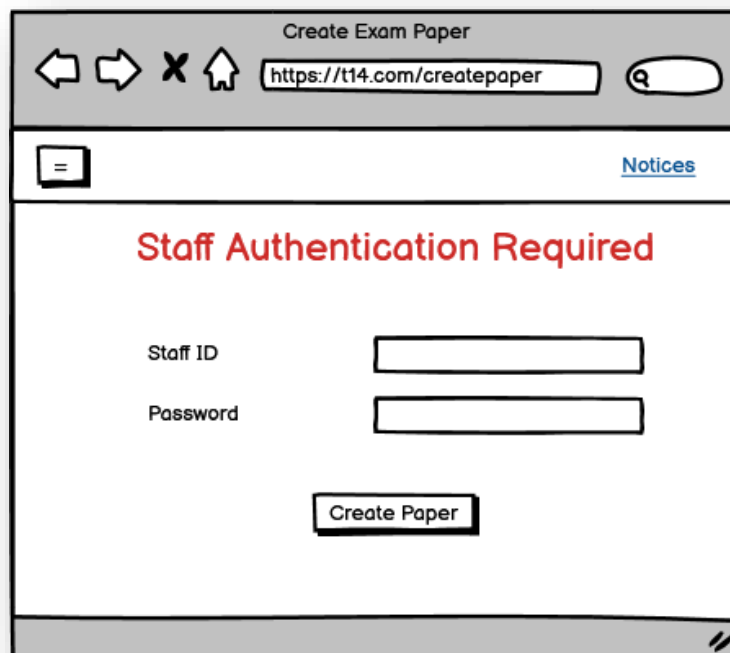


Figure 19: Prototype for the requirement of staff authentication to create exam papers or videos

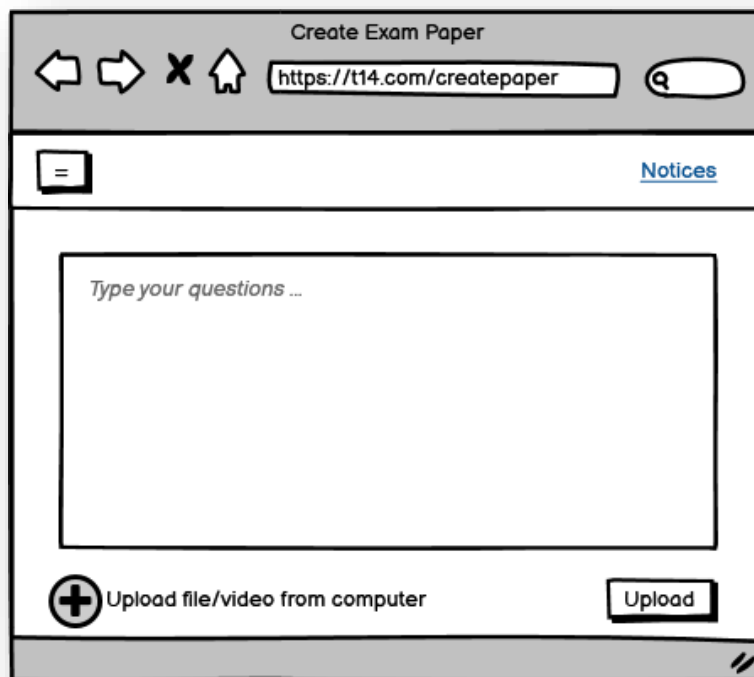


Figure 20: Prototype for creating exam questions and videos by staff

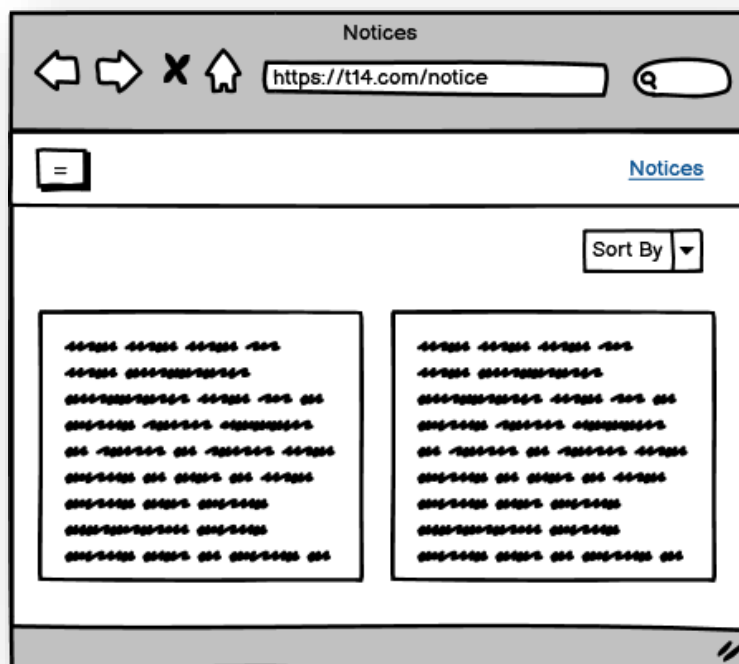


Figure 21: Prototype for viewing notices uploaded by admin

8. Conclusion

From the completion of this coursework, an understanding of software engineering and the use of UML diagrams in modeling the system was gained. With the main focus on object-oriented programming, software was created for the T-14 training academy. This coursework is the continuation of the previous one where the requirements – functional and nonfunctional, DTD, SRS, and others – are already done. Now the remaining part of SDLC, which involves a Gantt chart, communication diagram, class diagram, expanded and high-level use case descriptions prototype development are created.

In this task, the system is created following the DSDM principles which is an incremental approach with frequent releases and testings. Likewise, the system is created by defining the class instantiation. Thus, a creational design pattern is used. The MVC architectural pattern is used followed by various testing and maintenance. By implementing these different methods of the software development process, the task's object to develop a small system of software for the T-14 Training Academy is completed. This coursework helped in being familiar with how the system's architecture is put together to produce a user-friendly home application. But, this coursework was not easy at all. This coursework was completely based on theory and lots of research was to be done. It was very challenging. Constant net surfing, and questionnaires to module teachers, and seniors, were done. Articles and books were read for a proper understanding of different concepts. By the end, the task was completed by letting us learn a lot about software development.

Overall, the task made us learn a whole bunch of things about software development. After finishing this task, it felt like the whole world of the development process still exists that is yet to be learned. This has motivated me to want to learn more about software architecture and the development process. Also, we were allowed to see the work of software engineers in action which has enhanced our learning experience. It has also aided in the adoption of an analytical approach to problem-solving and made us recognize that system development is a long-term process.

9. References

- FlexBase, 2019. *medium.com*. [Online]
Available at: <https://medium.com/@flexbasenet/topic-software-maintainability-checklist-for-software-architects-44527ae5f2af#:~:text=So%2C%20what%20is%20maintainability%3F,of%20a%20failure%20or%20downtime.>
[Accessed 8 May 2022].
- Gisclard-Biondi, H., 2021. *appvizer*. [Online]
Available at: <https://www.appvizer.com/magazine/operations/project-management/dsdm>
[Accessed 8 May 2022].
- Heckel, R. & Sauer, S., 2001. *Strengthening UML Collaboration Diagrams by State Transformations*. s.l.:s.n.
- Waykar, Y., 2015. Role of Use Case Diagram in Software Development. *International Journal of Management and Economics*.