[CS350] 2021-Fall, Team Project

Project #3. Software Design Description (SDD)

Open: 2021-11-02 (Nov 2nd, 2021)
Due: 2021-11-21 (Nov 21st, 2021)

In the idea proposal, your team proposed an idea of an IoT system that incorporates at least one sensor, and you also devised functional and non-functional requirements from the description. In SRS, you have made the analysis of use cases, domain, and specified several acceptance criteria. In this project, engineers (in particular, developers) of your system want to see how the basic functionalities of the system should be implemented and how they should work. In order to meet the requirements, your team will refine the requirements and tasks, and conduct the design process.

Software Design Description (SDD) usually includes the design of system architecture, data, components, and user interface. However, we simplify the tasks and the SDD document will include (i) Title Page, (ii) System Overview, (iii) System Design (Architecture, Class, UI design). (no more than **30 pages**, use appendix if needed)

1. Title Page

Include the name of the document, team name, team members, date, and table of contents (including page numbers and so on) of the document.

2. System Overview

Write a quick description of the system. Refine the requirements that you defined in Project 2, and describe some assumptions and dependencies of your system. You also need to specify the tasks to accomplish the requirements, and map them to requirements.

A. Requirements

Sort out the requirements to achieve main goals of your system. You need to classify and number the requirements with tables like the following.

Functional requirements		
R. ID	**Requirement Description**	**Dependencies/Assumptions**
R.F.1	Customer can make a reservation	-
Non-functional requirements		
R. ID	**Requirement Description**	
R.N.1	The response should be delivered within 1 second.	

*The IDs of requirements can be freely specified, and you can change the format of the requirements table if needed.

B. Tasks

List the tasks which should be done to develop your system. Make a task model in the hierarchical form. Identify which requirements are achieved by which task, and map all the requirements to the task model. The following example could be your task model.

Task Model		
Task ID	**Task description**	**Related Req(s).**
T.1	Build the development environment	-
T.2	Implement ‘make a reservation’ functionality	R.F.1
T.2.1	Implement ‘Retrieve the available room list’	R.N.1

*A task can be related to more than one requirement.

3. System Design

A. System Architecture

Provide a blueprint of your system with the structure and the relationships between system components. Any architectural format, such as, hierarchical, layered, agent-based style, can be used.

B. Class Diagram

Design classes for the implementation of your system, and draw an UML class diagram. Brief description for each class is required. We recommend that you refer to various design patterns of class diagrams.

(1) List of classes with descriptions and their classifications

(2) Class diagram

User Interface Design (Mockup design)

Design 3~5 main views of your system to provide a good user interface.

D. (Refined) Use Case Diagram & Description

Refine your use case diagram (1 diagram) and description (at least 6 descriptions) that you defined in Project 2. Highlight the refined use cases in the diagram in red.

E. (Refined) Sequence Diagram

Draw refined sequence diagrams (at least 6 diagrams). Design the interactions based on your team’s design/development patterns and styles. In your refined sequence diagram, participants should be specified consistently using (a) actors defined in Project 2 and (b) classes defined in 3-*B*.

4. Acknowledgement

Include authors of this document, by section, and editor of document.

[Scoring criteria]

- **Readability**: readability of the descriptions and diagrams
- **Understandability**: understandability of the descriptions and diagrams
- **Completeness**: completeness of the descriptions and diagrams
- **Consistency**: consistency between requirements, use-case diagram and description, class diagram, and sequence diagram
- **Unambiguity:** unambiguity of the UML diagrams
- **Understanding of basic skills for UML modeling**: syntax and semantic understanding for UML

[Submission Deadline]

Submission Deadline: **Nov 21st on KLMS**.

** Submit your report to KLMS (not by e-mail)*

** TAs will answer the questions about this project only if you asked before **Nov 19th**.*

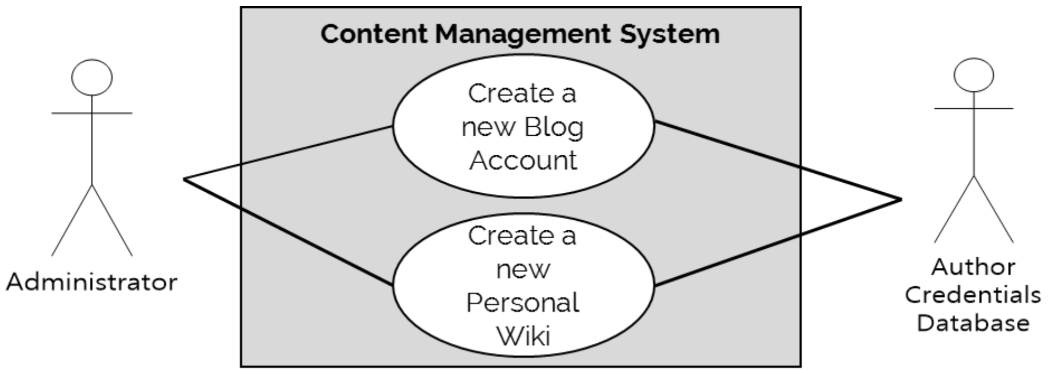
Appendix (Guideline)

Appendix A. Use Case Description Template

Use case name	**Name of a use case to be described**
Related Requirements	Some indication as to which requirements this use case partially or completely fulfills.
Goal in Context	The use case's place within the system and why this use case is important.
Preconditions	What needs to happen before the use case can be executed.
Successful End Condition	What the system's condition should be if the use case executes successfully.
Failed End Condition	What the system's condition should be if the use case fails to execute successfully.
Primary Actors	The main actors that participate in the use case. Often includes the actors that trigger or directly receive information from a use case's execution.
Secondary Actors	Actors that participate but are not the main players in a use case's execution.
Trigger	The event triggered by an actor that causes the use case to execute.
Main Flow	The place to describe each of the important steps in a use case's normal execution.
Extensions	A description of any alternative steps from the ones described in the Main Flow.

Appendix B. Use Case Description Example

(<Create a new blog account> use case of Content Management System (CMS))

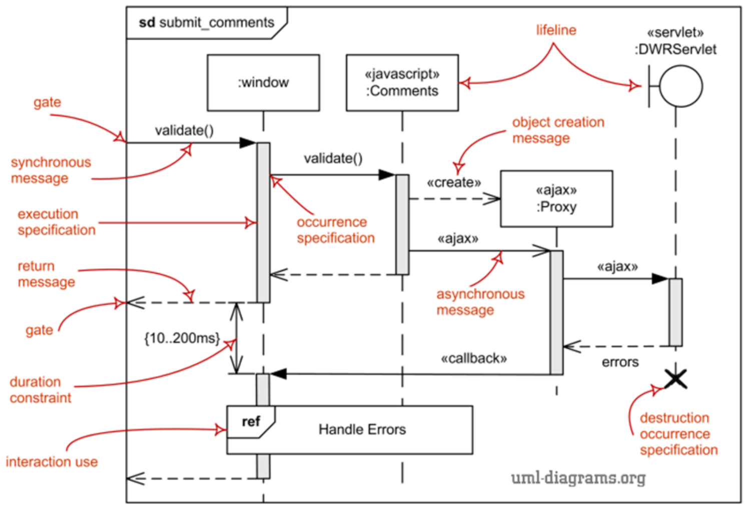

Content Management System
Create a new Blog Account
Create a new Personal Wiki
Administrator
Author Credentials Database

Use case diagram

Use case name	**Create a new blog account**	
Related Requirements	Requirement A.1.	
Goal in Context	A new or existing author requests a new blog account from the Administrator.	
Preconditions	The system is limited to recognized authors and so the author needs to have appropriate proof of identity.	
Successful End Condition	A new blog account is created for the author.	
Failed End Condition	The application for a new blog account is rejected.	
Primary Actors	Administrator.	
Secondary Actors	Author Credentials Database.	
Trigger	The Administrator asks the CMS to create a new blog account.	
Main Flow	**Step**	**Action**
	1	The Administrator asks the system to create a new blog account.
	2	The Administrator selects an account type.
	3	The Administrator enters the author's details.
	4	The author's details are verified using the Author Credentials Database.
	5	The new blog account is created.
	6	A summary of the new blog account's details are emailed to the author.
Extensions	**Step**	**Branching Action**
	4.1	The Author Credentials Database does not verify the author's details.
	4.2	The author's new blog account applications is rejected.

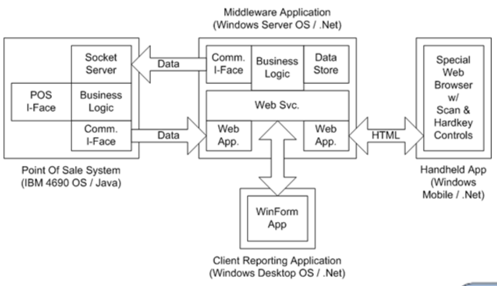
Appendix C. Sequence Diagram

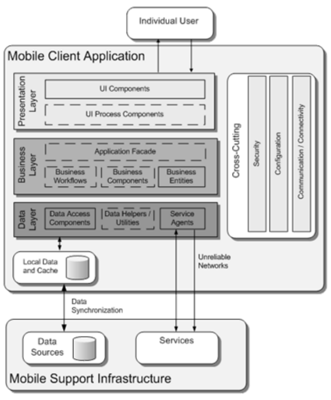
- **(Refined) Sequence Diagram**
  - To draw refined sequence diagrams.
    - At this sprint, it is important to consider your team's design/development patterns & methods & styles that you will actually use.
    - If there were ambiguous participants or interactions (i.e., messages/ordering), your team should describe them as clearly (i.e., unambiguously) as possible by the refinement.

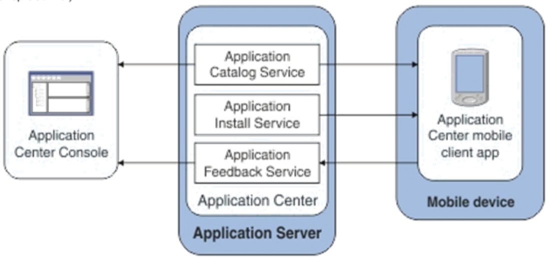


Appendix D. System Architecture

- **System Architecture**
  - System architecture should be able to provide a blue print of your system by describing the structure and some relationships between the components.
    - "Imagine an architectural drawings for the building construction"
  - A format for describing a system architecture is not fixed.

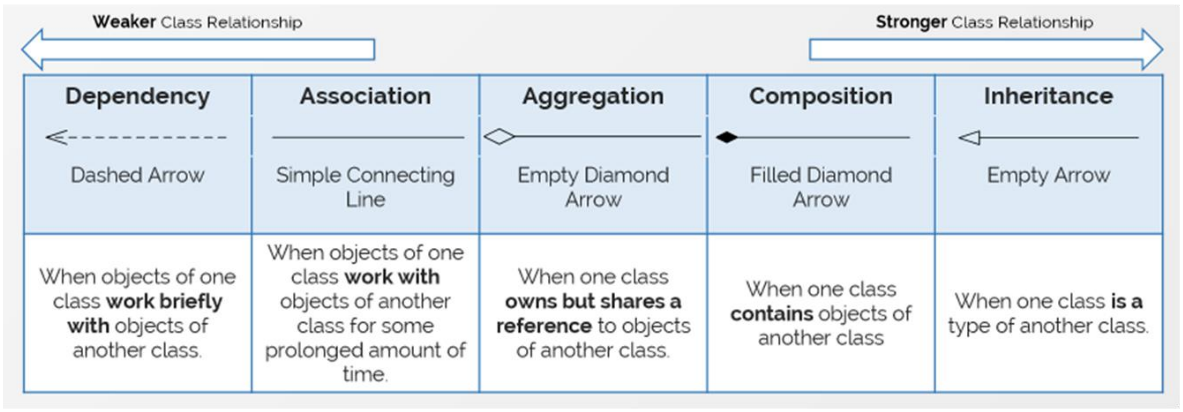

Middleware Application (Windows Server OS / .Net)
Socket Server
POS I-Face
Business Logic
Comm. I-Face
Data
Data
Comm. I-Face
Business Logic
Data Store
Web Svc.
Web App.
Web App.
HTML
Special Web Browser w/ Scan & Hardkey Controls
Point Of Sale System (IBM 4690 OS / Java)
Handheld App (Windows Mobile / .Net)
WinForm App
Client Reporting Application (Windows Desktop OS / .Net)




Application Center Console
Application Catalog Service
Application Install Service
Application Feedback Service
Application Center
Application Server
Application Center mobile client app
Mobile device

Appendix E. Class Diagram

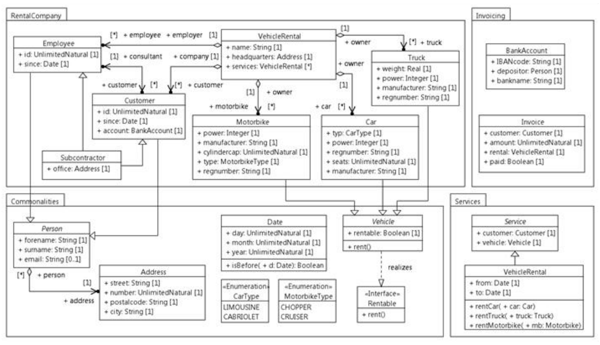
- **Class Diagram**
  - To identify classes for the implementation
    - What class will you implement in your system? (e.g., in Java/Python/HTML)
  - To design classes for the implementation by specifying identified classes and relationships between the classes.
    - Relationships should be properly used to design classes of your system

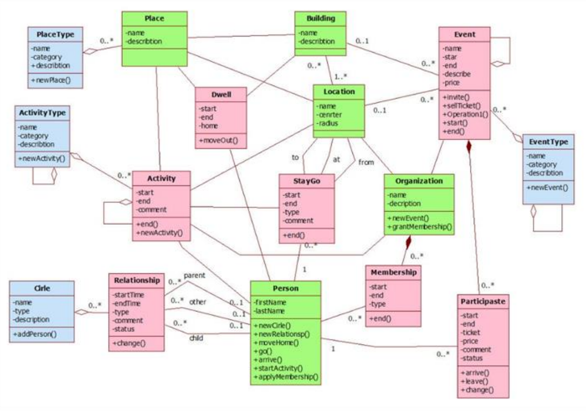


Weaker Class Relationship ⟵ ⟶ **Stronger** Class Relationship

Dependency	**Association**	**Aggregation**	**Composition**	**Inheritance**
<- - - - - - -	———	◇———	◆———	◁———
Dashed Arrow	Simple Connecting Line	Empty Diamond Arrow	Filled Diamond Arrow	Empty Arrow
When objects of one class **work briefly with** objects of another class.	When objects of one class **work with** objects of another class for some prolonged amount of time.	When one class **owns but shares a reference** to objects of another class.	When one class **contains** objects of another class	When one class **is a** type of another class.

- **Class Diagram**
  - To identify classes for the implementation
  - To design classes for the implementation by specifying identified classes and relationships between the classes.
    - Relationships should be properly used to design classes of your system


RentalCompany
Employee
+ id: UnlimitedNatural [1]
+ since: Date [1]
[*] + employee + employer [1]
[1] + consultant + company [1]
VehicleRental
+ name: String [1]
+ headquarters: Address [1]
+ services: VehicleRental [*]
[1] + owner [*] + truck
+ owner [1]
Truck
+ weight: Real [1]
+ power: Integer [1]
+ manufacturer: String [1]
+ regnumber: String [1]
+ customer [*] [*] + customer
[1] + owner
+ motorbike [*]
+ car [*]
Customer
+ id: UnlimitedNatural [1]
+ since: Date [1]
+ account: BankAccount [1]
Subcontractor
+ office: Address [1]
Motorbike
+ power: Integer [1]
+ manufacturer: String [1]
+ cylindercap: UnlimitedNatural [1]
+ type: MotorbikeType [1]
+ regnumber: String [1]
Car
+ typ: CarType [1]
+ power: Integer [1]
+ regnumber: String [1]
+ seats: UnlimitedNatural [1]
+ manufacturer: String [1]
Invoicing
BankAccount
+ IBANcode: String [1]
+ depositor: Person [1]
+ bankname: String [1]
Invoice
+ customer: Customer [1]
+ amount: UnlimitedNatural [1]
+ rental: VehicleRental [1]
+ paid: Boolean [1]
Commonalities
Person
+ forename: String [1]
+ surname: String [1]
+ email: String [0..1]
[*] + person
[1] + address
Address
+ street: String [1]
+ number: UnlimitedNatural [1]
+ postalcode: String [1]
+ city: String [1]
Date
+ day: UnlimitedNatural [1]
+ month: UnlimitedNatural [1]
+ year: UnlimitedNatural [1]
+ isBefore(+ d: Date): Boolean
«Enumeration» CarType
LIMOUSINE
CABRIOLET
«Enumeration» MotorbikeType
CHOPPER
CRUISER
Vehicle
+ rentable: Boolean [1]
+ rent()
realizes
«Interface» Rentable
+ rent()
Services
Service
+ customer: Customer [1]
+ vehicle: Vehicle [1]
VehicleRental
+ from: Date [1]
+ to: Date [1]
+ rentCar(+ car: Car)
+ rentTruck(+ truck: Truck)
+ rentMotorbike(+ mb: Motorbike)


PlaceType
-name
-category
+describtion
+newPlace()
Place
-name
-describtion
Building
-name
-describtion
Event
-name
-star
-end
-describe
-price
+invite()
+sellTicket()
+Operation1()
+start()
+end()
ActivityType
-name
-category
-describtion
+newActivity()
Dwell
-start
-end
-home
+moveOut()
Location
-name
-cenrter
-radius
EventType
-name
-category
-describtion
+newEvent()
Activity
-start
-end
-comment
+end()
+newActivity()
StayGo
-start
-end
-type
-comment
+end()
Organization
-name
-decription
+newEvent()
+grantMembership()
to
at
from
Membership
-start
-end
-type
+end()
Cirle
-name
-type
-description
+addPerson()
Relationship
-startTime
-endTime
-type
-comment
-status
+change()
parent
other
child
Person
-firstName
-lastName
+newCirle()
+newRelationsp()
+moveHome()
+go()
+arrive()
+startActivity()
+applyMembership()
Participaste
-start
-end
-ticket
-price
-comment
-status
+arrive()
+leave()
+change()
0..*
0..1
1..*
1