Semestral Work

Design of Information System

Faculty of Mechanical Engineering

Czech Technical University in Prague

Department of Instrumentation and Control Engineering

ČVUT

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Omar Alif Allam

Project

UML for a library management system

Contents

Table of figures

Introduction

The Library Management System Activity Diagram is one of the UML behavioural diagrams. It shows the system's behaviour by presenting the flow of activities from one to another. The possible flow of activities can be in order, split, or continuous.

The table below lists the names and details of the UML diagrams for the library management system. It contains detailed information on project components as well as diagramming tools.

Name:	**Library Management System UML Diagrams**
Abstract:	Library Management System UML Diagrams are used to represent the library management system as well as its primary users, roles, activities, and classes.
Users:	**Member, Library, Author and Librarian.**
Tools Used:	Diagram tools that provide UML diagram symbols (Umbrello). Natural Language Processing with Python (NLP) using NLTK library.

Scope

This purpose of this document is to firstly introduces the main content of the Unified Modelling Language (UML) and proves that it can transmit information among the users, the developers, the designers and the managers efficiently, which improves their collaboration capabilities and greatly increases the degree of industrialization in software development projects.

Secondly, a library management system development and design are carried out, based on UML modelling mechanism to analyse a simple library management system.

Thirdly, our approach is to first use lexical analysis to process our normal text from the software requirements specifications by means of Natural Language Processing with Python (NLP) using NLTK library to identify our main classes and attributes. A demand analysis mode of the management system is built with the help of a case diagram and an analysis state machine diagram after analysing our simple library management system, using lexical analysis and UML modelling mechanism. Then a borrowed/check out book management scenario has been designed by means of state machine diagram and a sequence diagram for one of the classes. The design process indicates that as a modelling language of software engineering, UML has a very good application prospect to summarize our software requirement specifications.

Software Requirement Specification (SRS)

In most cases, requirement analysis is performed to determine the system function and user interface. It enables users to comprehend system functionality and developers to understand the system requirements. The requirements analysis' principal task is to identify system use cases diagram and create a system requirements model. The use of case diagram, state machine diagram and sequence diagram are the major outcomes.

Requirements analysis phase

First, we are analysing and refine the requirements of the system. In addition, Listing the involved operators in the system and identify all the use cases and roles to describe. And then to analyse the relationship between the roles and the cases and use UML modelling tool to draw the case diagram.

System analysis and design phase

Identify all the needs of the system. Abstract the classes from the actual needs and describe the relationship between these classes. Extracting our attributes, actors, and classes from analysing our text in the Natural Language Processing with Python (NLP) using NLTK library.

Phase of system implementation

The model in the first two phases is actually created in a system logic aspect. This phase is to accomplish the physical realization of the system, such as main classes, operators or associations and attributes.

The library information management system is an electronic archives management system that processes a large number of book information using a computer. The needs of three sorts of users must be met by this system: member, librarians, author and library (to check for another branch if found). Many individuals can be categorized as user or a librarian based on their status, and only one person serves as the system administrator. The readers' behaviour is to query the personal information, to query the books, to book, borrow and return them. In case of the due date of borrowing a book has passed, the member will be obliged to pay a fine.

A library database must store information about its members, librarian, and the rack number of the book where it can be found. Each book item's status, descriptive attributes, and cost for losses and late returns must be tracked by the library/librarian. The ISBN of books will be used to identify them. Each book will have a unique ISBN in order to allow multiple copies of the same book. For the users to be a member they need to apply for a library card, where details about the member can identified (card number, barcode, issue date and active status).

Furthermore, when applying for a library card, members must create an account, then they will be given, unique ID, their names, and password (that can be reset later after account activation by librarian). In addition, a library card will be required for checkout operations. The library card will have its own fines, but any outstanding fines on a member's cards will prevent them from using the library's services.

The system's primary users are librarians. They are in charge of day-to-day management and service operations such as book ordering, new book verification, book registration, borrow and return books from members and so on. The librarians are also in charge of all issues concerning books, including book status (borrowed, not found and etc), library management (to check if the book can be found in another branch). The key point among them is system maintenance, which includes the maintenance of member rights and account details, the creation and activation/deactivation of members' account, fine payment, and so on.

For further elaboration, check our Library Management System (Librarian) state machine diagram – This diagram depicts the activities and scenarios that occur when a member checks out a book. The actions and decisions were all highlighted in the state machine diagram see figure 2.

While designing our Library Management System, we will concentrate on the following requirements:

- ➢ Books should be searchable by title, author, subject category, and publication date for any library member.
- ➢ Each book will have a unique identification number as well as other details such as a rack number that will aid in physically locating the book.
- ➢ A book might have multiple copies, and library users should be able to borrow and reserve any of them. Each copy of a book will be referred to as a book item.
- ➢ The system should be able to retrieve information such as who took a specific book or what books are checked out by a specific library member.
- ➢ There should be a maximum limit of 5 books that a member can check out.
- ➢ A member should be allowed to keep a book for a maximum of 10 days.
- ➢ For books returned after the due date, the user/customer must pay a fine.
- ➢ Members should be able to reserve books that are currently unavailable.
- ➢ The system should be able to send notifications when reserved books become available, as well as when the book is not returned by the due date.
- ➢ Each book and member will have their own barcode. The system will be able to read barcodes from books as well as library cards from members.

In our system, there are three main actors:

- ✓ **Librarians** are primarily in charge of adding and modifying books, book items, and users. In addition, the Librarian can issue, reserve, and return book items.
- ✓ **Member**: All members have access to the catalogue and can check out, reserve, renew, and return books.
- ✓ **System**: Is primarily in charge of sending notifications for overdue books, cancelled reservations, and so on.

Our Library Management System is divided into the following classes:

- **Library**: The heart of the organization for which this software was created. It has attributes such as 'Name' to differentiate it from other libraries and 'Address' to describe its location, the class should be able to identify in which branch is the book located

- **Book**: The system's fundamental building block. Every book will have an ISBN, a title, and subject.

- **Book Item:** Any book can have multiple copies, and each copy is treated as a book item in our system. Each book will have its own barcode. In addition, giving information if the book is borrowed and when it should be returned by the members' user. Information also about the publication date of the book and date of purchase if the member bought it. As an operation process; librarian should be able to check out the book

- **Account**: There will be two types of accounts in the system, one for general members and one for librarians, where every class will have its own id, password and username. Member or librarian should be able to reset their passwords.

- **Library Card**: A library card will be issued to each library user, which will be used to identify users when issuing or returning books. Where there will be a unidirectional association with the *Account* details. Additionally, it should hold information like card number, barcode, issue date and account status.

- **Book Reservation**: In charge of managing reservations for book items based on the creation date and status. If the account is active book reservation can be confirmed

- **Book Lending**: Manage the loaning of books. Due date & return date should be known.

- **Catalogue**: A list of books sorted according to certain criteria. Our system will allow you to search four catalogues by title, author, subject, and publish date.

- **Fines**: in case the lend book is not returned by the due date specified in the *Book Lending class*; This class will be in charge of calculating and collecting fines from library patrons.

- **Bill**: this class will be referred from the transaction to be paid by the member. And librarian will be able to create, issue and update bill.

- **Author**: This class will focus on the author of a book, storing name of the book and a brief description.

- **Rack**: Books will be arranged on shelves. Each rack will be identified by a rack number and a location identifier that describes the rack's physical location in the library.

- **Notification:** This class will handle sending notifications to library users to their email addresses.

Lexical analysis for SRS by means of NLP (NLTK in python)

The process of creating Unified Modelling Language (UML) Diagrams from Natural Language (NL) requirements is generally viewed as complex and challenging. Software requirements specifications are frequently written in NL format, which can lead to issues. For this reason we are using NLP to analyse our software requirement specifications so we can create our class diagram and the attributes needed for each class.

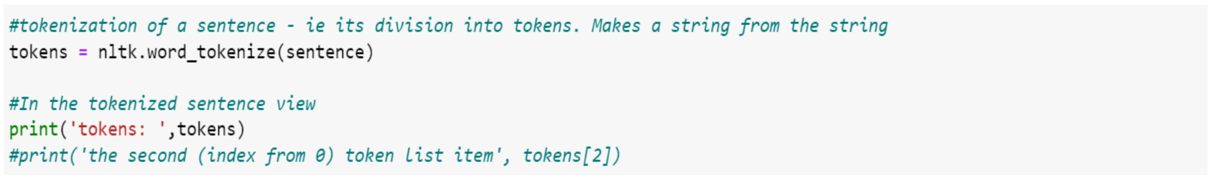
Analysts for requirements manually analyse and process natural language requirements to extract UML elements. Manual analysis takes a significant amount of time and effort, highlighting the need for automated assistance. Using natural language processing (NLP) techniques, this section proposes a method for facilitating the NL requirements analysis process and the extraction of UML diagrams from NL textual requirements specified in the discerption of our system in the software requirement specification.

Methodology

By processing our SRS in NLTK in python. Firstly, we added our software requirement specification described in our previous section and assigned the whole text as a **sentence.** Then we tokenized our sentence and tagging the sentence to divide it by part (chunks) & tokens (entities); the process in which our large quantity of text from SRS is divided into smaller parts.

<u>Word Tokenization</u>

Tokenization is the process of dividing raw text into small chunks. Tokenization divides raw text into words and sentences known as tokens. These tokens aid in understanding the context or developing the NLP model. Tokenization aids in interpreting the meaning of the text by analysing the word sequence.



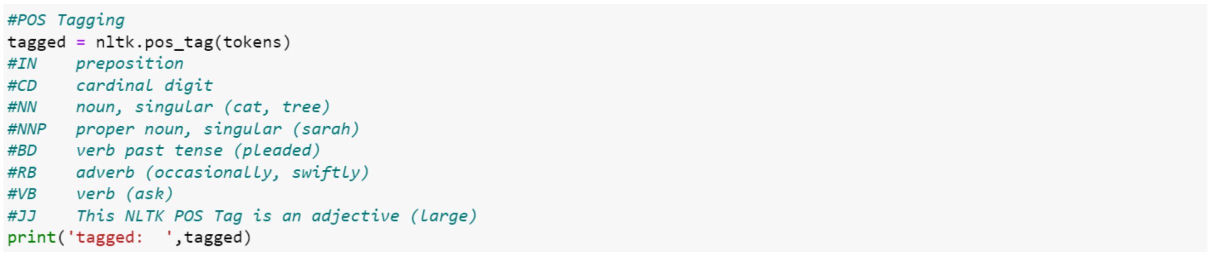
```
#tokenization of a sentence - ie its division into tokens. Makes a string from the string
tokens = nltk.word_tokenize(sentence)

#In the tokenized sentence view
print('tokens: ',tokens)
#print('the second (index from 0) token list item', tokens[2])
```

Figure 1:NLTK Analysis - Tokenization

<u>POS Tagging</u>

Part-of-speech (POS) Tagging is a popular Natural Language Processing process that involves categorizing words in a text (corpus) in accordance with a specific part of speech, based on the definition of the word and its context.

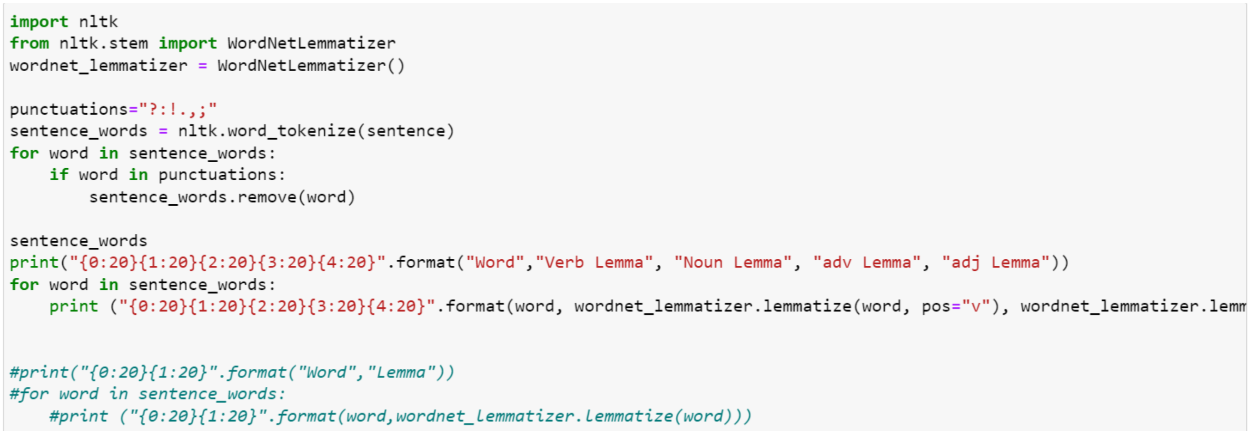


```
#POS Tagging
tagged = nltk.pos_tag(tokens)
#IN    preposition
#CD    cardinal digit
#NN    noun, singular (cat, tree)
#NNP   proper noun, singular (sarah)
#BD    verb past tense (pleaded)
#RB    adverb (occasionally, swiftly)
#VB    verb (ask)
#JJ    This NLTK POS Tag is an adjective (large)
print('tagged:  ',tagged)
```

Figure 2: NLTK Analysis - POS Tagging

Lemmatization

Lemmatization usually refers to doing things correctly by using a vocabulary and morphological analysis of words, with the goal of removing only inflectional endings and returning the base or dictionary form of a word, which is known as the lemma. Below is the process on how we developed our text to lemmatize the words and categorized them to verbs, adverbs and adjectives to return the base form.



```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

punctuations="?:!.,;"
sentence_words = nltk.word_tokenize(sentence)
for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

sentence_words
print("{0:20}{1:20}{2:20}{3:20}{4:20}".format("Word","Verb Lemma", "Noun Lemma", "adv Lemma", "adj Lemma"))
for word in sentence_words:
    print ("{0:20}{1:20}{2:20}{3:20}{4:20}".format(word, wordnet_lemmatizer.lemmatize(word, pos="v"), wordnet_lemmatizer.lemm


#print("{0:20}{1:20}".format("Word","Lemma"))
#for word in sentence_words:
    #print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word)))
```

Figure 3: NLTK Analysis - Lemmatization

Stemming

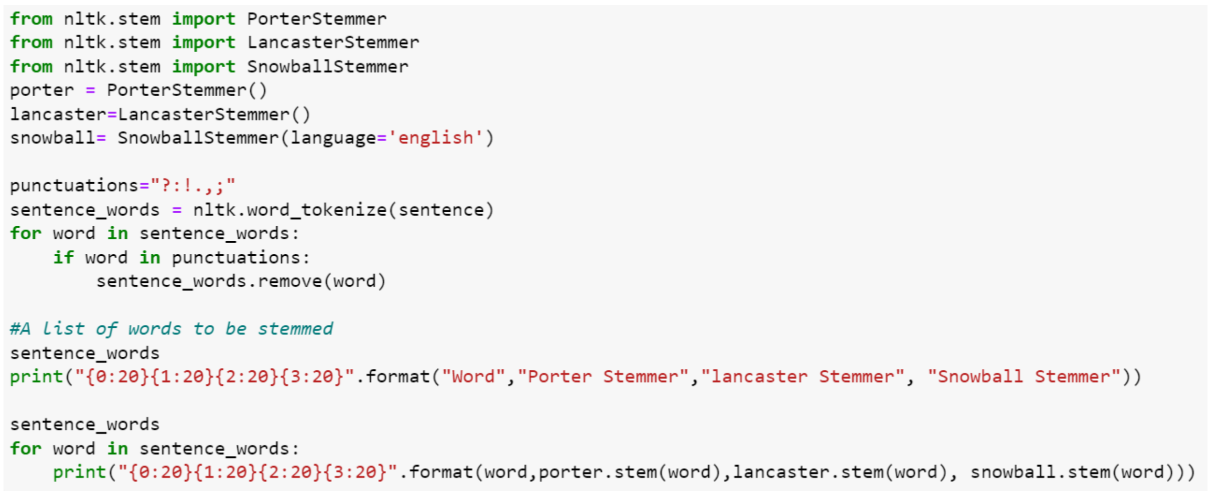
Stemming is a word normalization method used in Natural Language Processing. It is a technique that converts a set of words in a sentence into a sequence in order to shorten the lookup time. This method normalizes words that have the same meaning but differ slightly depending on the context or sentence. The nltk package includes both English and non-English stemmers. A stemming program, stemming algorithm, or stemmer is a computer program or subroutine that stems words. we can choose between Porter-Stemmer, Lancaster-Stemmer or Snowball-Stemmer for the English language.

Porter-Stemmer: uses Suffix Stripping to produce stems. It is well-known for its simplicity and speed. It is frequently used in Information Retrieval Environments (IR Environments) for quick recall and retrieval of search queries. Environment documents are typically represented as vectors of words or terms in an IR. Words with the same stem will have the same meaning.

Lancaster-Stemmer: is an iterative algorithm with externally saved rules. One table with approximately 120 rules indexed by the last letter of a suffix. It attempts to find an applicable rule by the last character of the word on each iteration. Each rule specifies whether an ending should be deleted or replaced. It ends if there is no such rule. It also terminates if a word begins with a vowel and only two letters remain, or if a word begins with a consonant and only three characters remain. Otherwise, the rule is applied, and the cycle begins again.

Snowball-Stemmer: Python NLTK included Snowball-Stemmers as a language to create a specific language stemmer (English, German, Czech…etc). In our case we programmed our snowball-Stemmer to English.

Below is the code on how we processed our text to stem it using three methods for stemming the word is stemmed correctly.



```python
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import SnowballStemmer
porter = PorterStemmer()
lancaster=LancasterStemmer()
snowball= SnowballStemmer(language='english')

punctuations="?:!.,;"
sentence_words = nltk.word_tokenize(sentence)
for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

#A list of words to be stemmed
sentence_words
print("{0:20}{1:20}{2:20}{3:20}".format("Word","Porter Stemmer","lancaster Stemmer", "Snowball Stemmer"))

sentence_words
for word in sentence_words:
    print("{0:20}{1:20}{2:20}{3:20}".format(word,porter.stem(word),lancaster.stem(word), snowball.stem(word)))
```

Figure 4: NLTK Analysis - Stemming

Filtration/Preparation

Three lists (Python dictionaries of nouns, adjectives, and verbs) are filtered according to predefined rules to eliminate words that are irrelevant to the problem being modelled:

- W - wide (eg. system, means, parameters, etc.)
- V - vague, indefinite (e.g. the first, case, equipment, etc.)
- S - synonymous (and also all tokens of one linguistic type)
- D Class name determinant (specifies the name of the class, e.g. screw is determinant of the class conveyor)
- S - Irrelevant (not important for the given model)
- P - property (property of any class is shifted to the given class as a property)
- A - association (is shifted to the association's list)
- - operation (of any class is shifted to the given class as an operation)

Next, the class names (D) are defined, operation (O) and Attributes (P) are connected to the classes (C). Associations are put into the diagram in the next phase.

Then, we were able to identify our class names (D), operators/associations (O/A), and attributes (P) to build our UML diagrams. Check the table below for classifying the words according to its category, in which we extracted the words we are interested in to build our UML diagrams

After Analysing our text and obtaining the class names, attributes, associations/operators; we can proceed further to build our system in Umbrello, where we:

- Synthetize the UML Class diagram and use classes & attributes extracted from our SRS by means of NLTK in python.
- create a UML State machine diagram for one important class.
- create a UML Sequence diagram for a few scenarios.

Word	POS_Tagging	Porter Stemmer	lancaster Stemmer	Snowball Stemmer	Verb Lemma	Noun Lemma	adv Lemma	adj Lemma	Type	Filtration
account	NN	account	account	account	account	account	account	account	Attributes	P
account	NN	account	account	account	account	account	account	account	Class Name	D
account	VBP	account	account	account	account	account	account	account	Asosciations/Operators	A/O
active	JJ	activ	act	activ	active	active	active	active	Asosciations/Operators	A/O
activities	NNS	activ	act	activ	activities	activity	activities	activities	Attributes	P
activities	NNS	activ	act	activ	activities	activity	activities	activities	Asosciations/Operators	A/O
Address	NN	address	'address	address	Address	Address	Address	Address	Attributes	P
addresses	NNS	address	address	address	address	address	addresses	addresses	Attributes	P
amount	NN	amount	amount	amount	amount	amount	amount	amount	Attributes	P
author	NN	author	auth	author	author	author	author	author	Attributes	P
author	NN	author	auth	author	author	author	author	author	Class Name	D
bank	NN	bank	bank	bank	bank	bank	bank	bank	Attributes	P
barcode	NN	barcod	barcod	barcod	barcode	barcode	barcode	barcode	Class Name	D
barcode	JJ	barcod	barcod	barcod	barcode	barcode	barcode	barcode	Attributes	P
Bill	VBP	bill	bil	bill	Bill	Bill	Bill	Bill	Asosciations/Operators	A/O
bill	NN	bill	bil	bill	bill	bill	bill	bill	Class Name	D
block	NN	block	block	block	block	block	block	block	Attributes	P
book	NN	book	book	book	book	book	book	book	Class Name	D
book	NN	book	book	book	book	book	book	book	Attributes	P
borrowed	VBN	borrow	borrow	borrow	borrow	borrowed	borrowed	borrowed	Attributes	P
borrowing	VBG	borrow	borrow	borrow	borrow	borrowing	borrowing	borrowing	Asosciations/Operators	A/O
card	NN	card	card	card	card	card	card	card	Attributes	P
cards	NNS	card	card	card	card	card	cards	cards	Class Name	D
cash	VB	cash	cash	cash	cash	cash	cash	cash	Class Name	D
cash	VB	cash	cash	cash	cash	cash	cash	cash	Attributes	P
catalogue	NN	catalogu	catalog	catalogu	catalogue	catalogue	catalogue	catalogue	Class Name	D
charge	NN	charg	charg	charg	charge	charge	charge	charge	Asosciations/Operators	A/O
check	VB	check	check	check	check	check	check	check	Asosciations/Operators	A/O
check	VB	check	check	check	check	check	check	check	Class Name	D
checkout	NN	checkout	checkout	checkout	checkout	checkout	checkout	checkout	Attributes/Associations/Operators	P/O/A
create	VB	creat	cre	creat	create	create	create	create	Asosciations/Operators	A/O
creation	NN	creation	cre	creation	creation	creation	creation	creation	Attributes	P
date	NN	date	dat	date	date	date	date	date	Attributes	P
date	NN	date	dat	date	date	date	date	date	Class Name	D
description	NN	descript	describ	descript	description	description	description	description	Attributes	P
Due	NNP	due	due	due	Due	Due	Due	Due	Class Name	D
due	JJ	due	due	due	due	due	due	due	Attributes	P
fines	NNS	fine	fin	fine	fin	fine	fines	fines	Class Name	D

get	VB	get	get	get	get	get	get	get	Asosciations/Operators	A/O
ID	NNP	id	id	id	ID	ID	ID	ID	Attributes	P
id	NN	id	id	id	id	id	id	id	Attributes	P
ISBN	NNP	isbn	isbn	isbn	ISBN	ISBN	ISBN	ISBN	Attributes	P
issue	NN	issu	issu	issu	issue	issue	issue	issue	Attributes	P
issued	VBN	issu	issu	issu	issue	issued	issued	issued	Asosciations/Operators	A/O
issues	NNS	issu	issu	issu	issue	issue	issues	issues	Class Name	D
item	NN	item	item	item	item	item	item	item	Attributes	P
items	NNS	item	item	item	items	item	items	items	Class Name	D
Lending	NNP	lend	lend	lend	Lending	Lending	Lending	Lending	Class Name	D
library	NN	librari	libr	librari	library	library	library	library	Class Name	D
library/librarian	NNP	library/librarian	library/librarian	library/librarian	library/librarian	library/librarian	library/librarian	library/librarian	Class Name	D
located	VBN	locat	loc	locat	locate	located	located	located	Asosciations/Operators	A/O
location	NN	locat	loc	locat	location	location	location	location	Class Name	D
location	NN	locat	loc	locat	location	location	location	location	Attributes	P
member	NN	member	memb	member	member	member	member	member	Class Name	D
name	NNS	name	nam	name	name	name	names	names	Attributes	P
Notification	NN	notif	not	notif	Notification	Notification	Notification	Notification	Class Name	D
notifications	NNS	notif	not	notif	notifications	notification	notifications	notifications	Attributes	P
number	NN	number	numb	number	number	number	number	number	Attributes	P
number	NN	number	numb	number	number	number	number	number	Class Name	D
overdue	JJ	overdu	overdu	overdu	overdue	overdue	overdue	overdue	Attributes	P
password	NN	password	password	password	password	password	password	password	Asosciations/Operators	A/O
passwords	NNS	password	password	password	passwords	password	passwords	passwords	Attributes	P
payment	NN	payment	pay	payment	payment	payment	payment	payment	Class Name	D
person	NN	person	person	person	person	person	person	person	Attributes	P
personal	JJ	person	person	person	personal	personal	personal	personal	Attributes	P
price	NN	price	price	price	price	price	price	price	Attributes	P
publication	NN	public	publ	public	publication	publication	publication	publication	Attributes	P
publication	NN	public	publ	public	publication	publication	publication	publication	Class Name	D
publish	JJ	publish	publ	publish	publish	publish	publish	publish	Attributes	P
purchase	NN	purchas	purchas	purchas	purchase	purchase	purchase	purchase	Attributes	P
rack	NN	rack	rack	rack	rack	rack	rack	rack	Class Name	D
rack	NN	rack	rack	rack	rack	rack	rack	rack	Attributes	P
readers	NNS	reader	read	reader	readers	reader	readers	readers	Class Name	D
Reservation	NN	reserv	reserv	reserv	Reservation	Reservation	Reservation	Reservation	Asosciations/Operators	A/O
reservation	NN	reserv	reserv	reserv	reservation	reservation	reservation	reservation	Attributes	P
reservations	NNS	reserv	reserv	reserv	reservations	reservation	reservations	reservations	Class Name	D
reserve	NN	reserv	reserv	reserv	reserve	reserve	reserve	reserve	Class Name	D

reset	VBN	reset	reset	reset	reset	reset	reset	reset	Asosciations/Operators	A/O
return	VB	return	return	return	return	return	return	return	Attributes	P
returned	VBN	return	return	return	return	returned	returned	returned	Asosciations/Operators	A/O
sending	VBG	send	send	send	send	sending	sending	sending	Asosciations/Operators	A/O
status	NN	statu	stat	status	status	status	status	status	Class Name	D
status	NN	statu	stat	status	status	status	status	status	Attributes	P
subject	JJ	subject	subject	subject	subject	subject	subject	subject	Attributes	P
subject	NN	subject	subject	subject	subject	subject	subject	subject	Class Name	D
title	NN	titl	titl	titl	title	title	title	title	Attributes	P
title	NN	titl	titl	titl	title	title	title	title	Class Name	D
total	NN	total	total	total	total	total	total	total	Attributes	P
tracked	VBN	track	track	track	track	tracked	tracked	tracked	Asosciations/Operators	A/O
transaction	NN	transact	transact	transact	transaction	transaction	transaction	transaction	Asosciations/Operators	A/O
transaction	NN	transact	transact	transact	transaction	transaction	transaction	transaction	Class Name	D
update	JJ	updat	upd	updat	update	update	update	update	Asosciations/Operators	A/O
username	JJ	usernam	usernam	usernam	username	username	username	username	Attributes	P

UML Class diagram

A Library Management System Class Diagram is a type of structural (UML) diagram that shows a system's structure. This is created by displaying the system's classes, attributes, methods, and class relationships.

It is important to understand our associations between our classes. For example, the figure below illustrated our dependencies and associations.

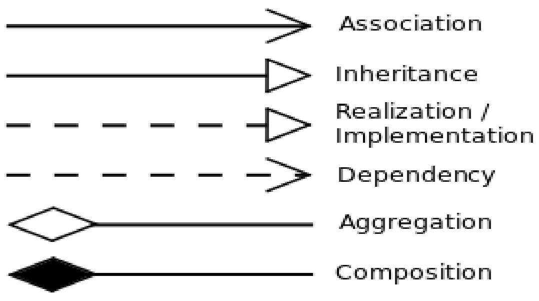


Figure 5: Relations

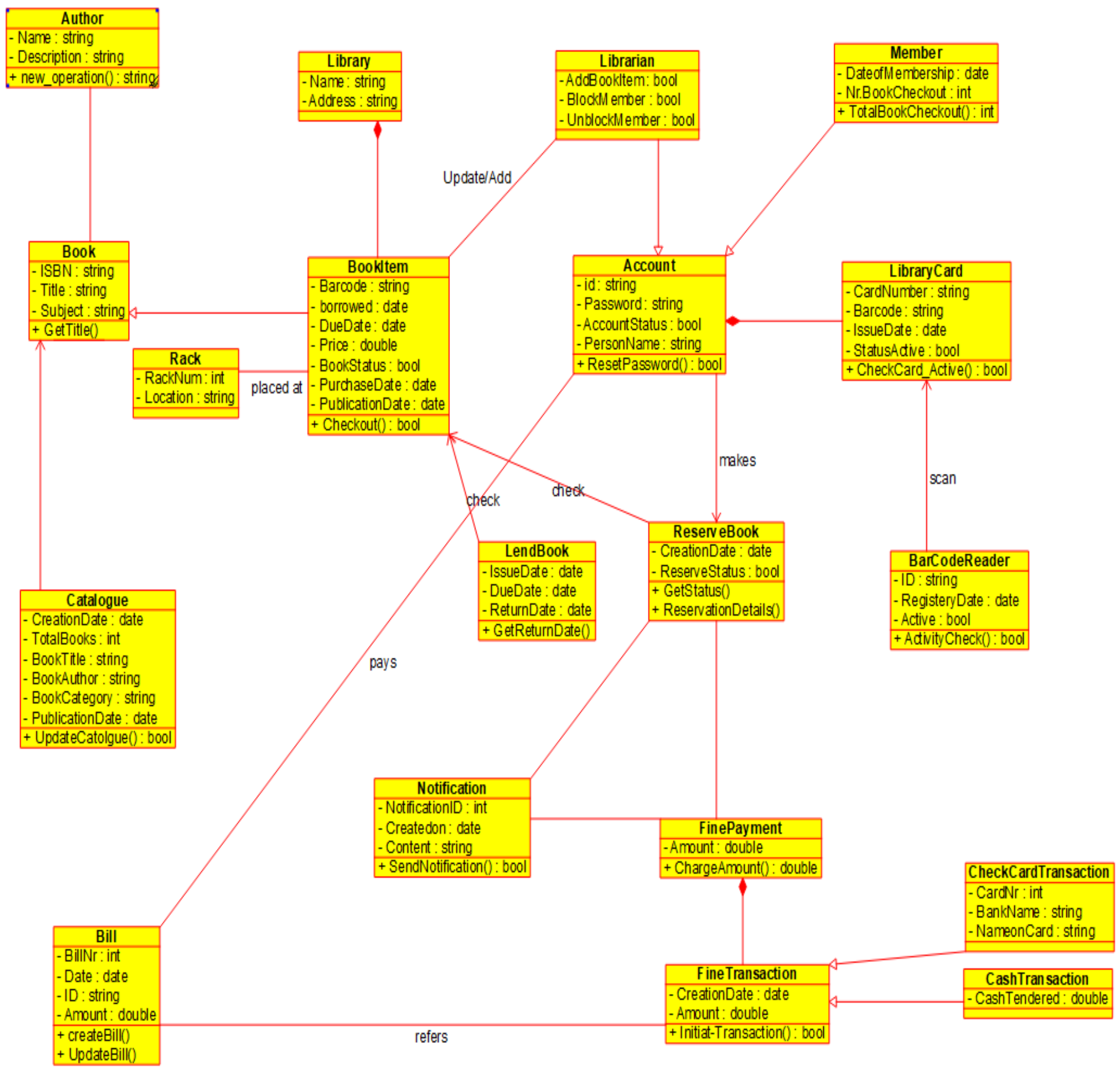
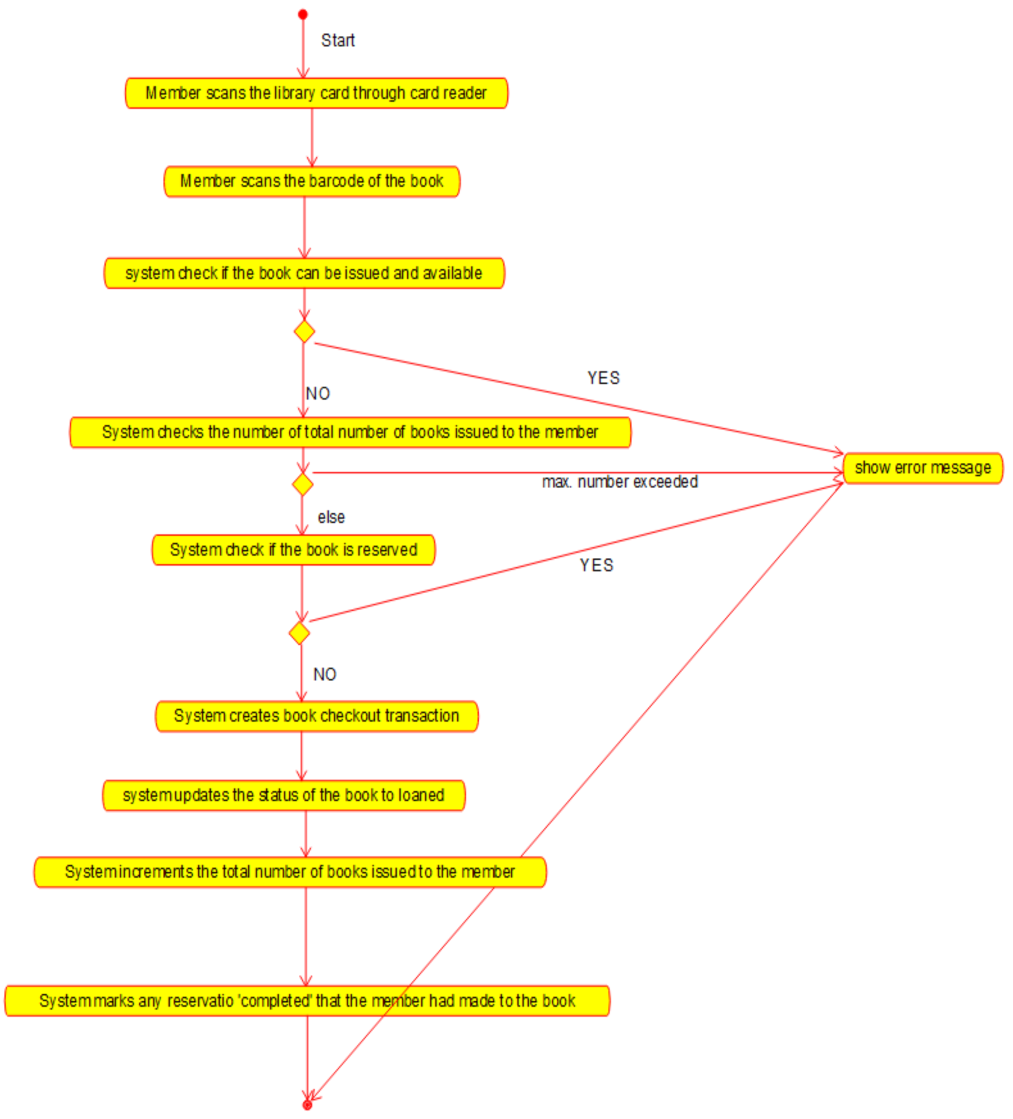


Figure 6: Class Diagram

State Machine Diagram

Check-out a book: Any library member or librarian can perform this activity. Here are the set of steps to check-out a book:



Start
Member scans the library card through card reader
Member scans the barcode of the book
system check if the book can be issued and available
YES
NO
System checks the number of total number of books issued to the member
show error message
max. number exceeded
else
System check if the book is reserved
YES
NO
System creates book checkout transaction
system updates the status of the book to loaned
System increments the total number of books issued to the member
System marks any reservatio 'completed' that the member had made to the book

Figure 7: State Machine Diagram

Sequence Diagram

The Sequence Diagram for Library Management System represents the scenario and the messages that must be passed between objects. This is done for the scenario's functionality to be realized. It's an interaction diagram that shows how activities are carried out, including when and how messages are sent between librarian, member and how transactions and bills are issued on the book.

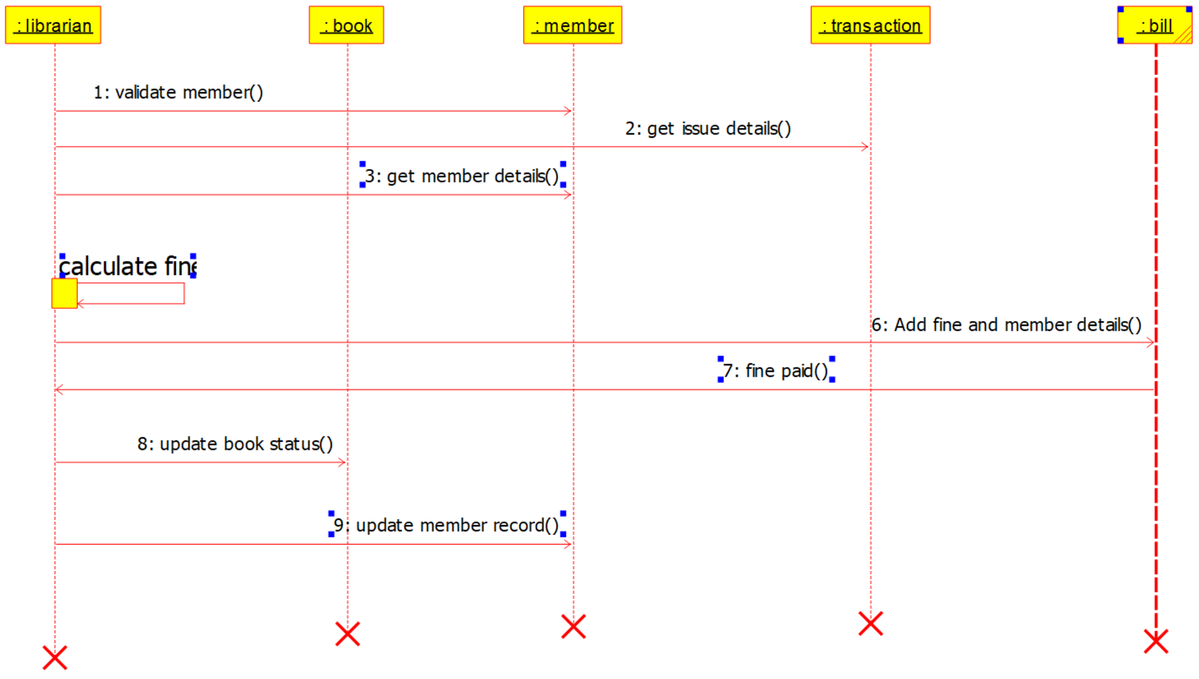


Figure 8: Sequence Diagram

Conclusion

To conclude, the Library Management System UML Diagrams collaborate to achieve their most important functions. All of these were created to instruct programmers and beginners on how the Library Management System should behave and be structured. Moreover, project development would be much easier and more feasible with the help all UML Diagrams. These UML diagrams were provided in order to teach and guide developers through project development. The UML Diagram ideas were all based on Library Management requirements.