MANİSA CELAL BAYAR UNIVERSITY

CSE 3105/ CSE 3137

OBJECT ORIENTED ANALYSIS AND DESIGN

FALL 2020

COURSE PROJECT: Media Browser Application Project

System Design Document

Group 1

Rümeysa Karagöz – 180315047
Merve İrmak- 180315038

Merve Uğurlu-180315039

Abdullah Cahid Alır-180315006

Baran Kalkan- 190315070

Deyan Bora Çetin- 180315016

16 January 2021

Table of Contents

1 Introduction

We originally planned to use model-view-controller architecture style. However, in order to add elements like online store, downloadable content and reduce the local storage usage we decided to use client/server style architecture.

1.1 Purpose of the System

Purpose of the system is to access and maintain local and online media.

1.2 Design goals

->Our program should be reliable in terms of daily usage.

->Our program should be modifiable for additional contents and features that can be added later.

->We put the usability first because our program has audience from a wide range of age.

->Our program should be cost efficient in terms of system usage and handling the operation at hand without causing too much complexity to the system. However, the program, due to information hiding, prevents each error from affecting the other parts of the system therefore makes it a robust program.

->We prioritized using newest options and functions to make our program more efficient and easier to use. However, this logic prevented us from implementing any backward compatibility.

2 Current Software Architecture

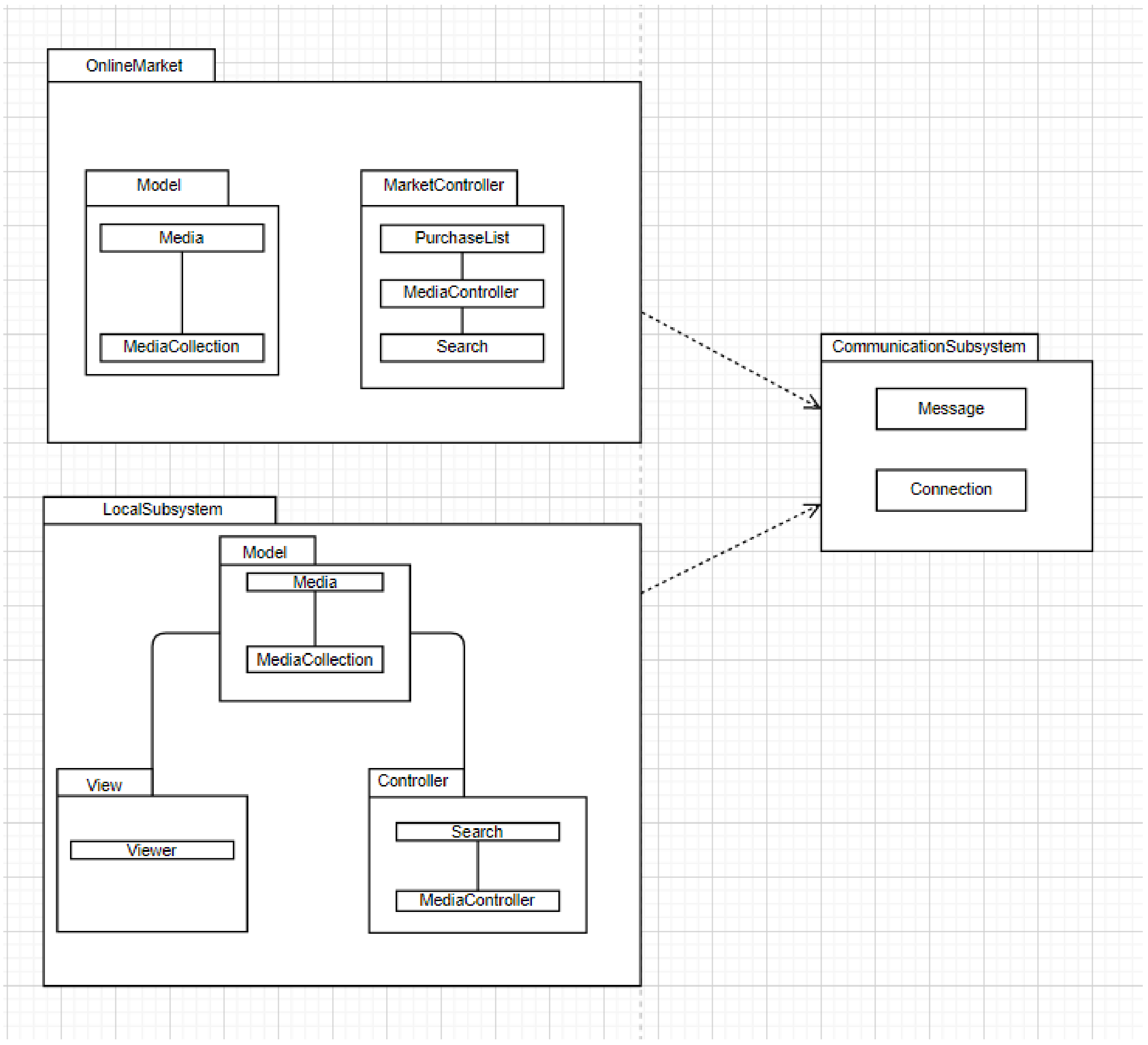
We took "Spotify" as an example of a current system. Spotify is a Swedish audio streaming and media service provider. Users can search for music based on artist, album or genre and can create, share, edit playlists. As for the system architecture, Spotify uses caching, peer-to-peer, client-server architectures by combining of them. This combination provides low latency service. Spotify has many advantages but at the same time there are disadvantages too. For example, Spotify is not quite enough about portability in some cases. PC version of Spotify has some advantages over mobile version of Spotify, like converting music to local drive. We can generate the songs to other formats, and we can save them to our local drive with PC version, but we cannot do this with mobile app.

3 Proposed Software Architecture

The third section, *Proposed Software Architecture*, documents the system design model of the new system. It is divided into five subsections:

3.1 Subsystem decomposition

We use client-server architecture because it provides us a platform online where we can store media, so it does not take up any space on our device. Furthermore by storing all the data online our device works more efficiently and using a server with online authentification provides more secure enviroment.



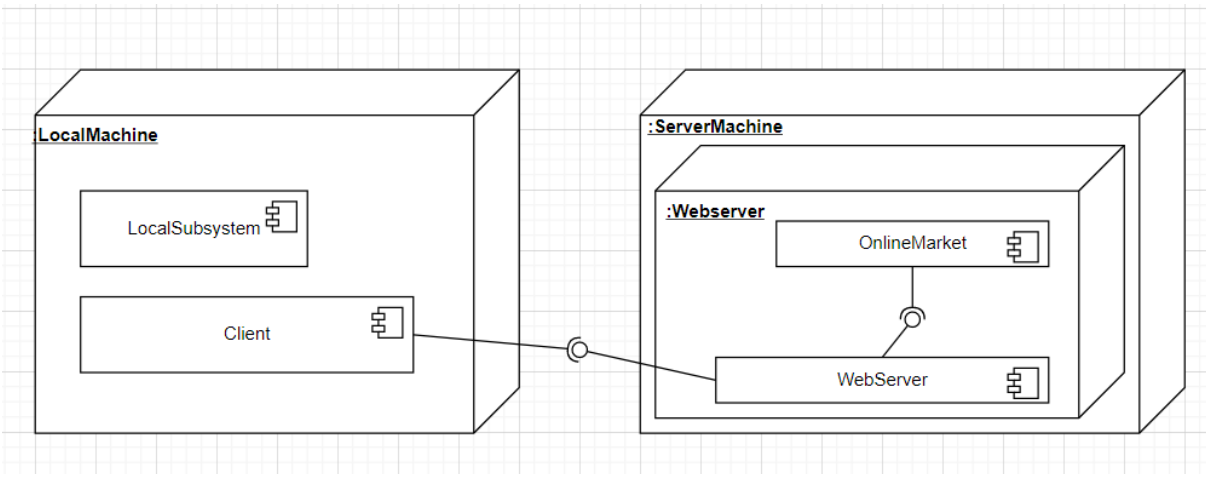
OnlineMarketSubsytem: The OnlineMarketSubsytem is responsible for purchasing and uploding Media files.

LocalSubsystem: The LocalSubsystem is responsible for performing all file operations registered in the local part.

CommunicationSubsytem: The CommunicationSubsystem is responsible for ensuring connection between LocalSubsytem and OnlineMarketSubsytem.

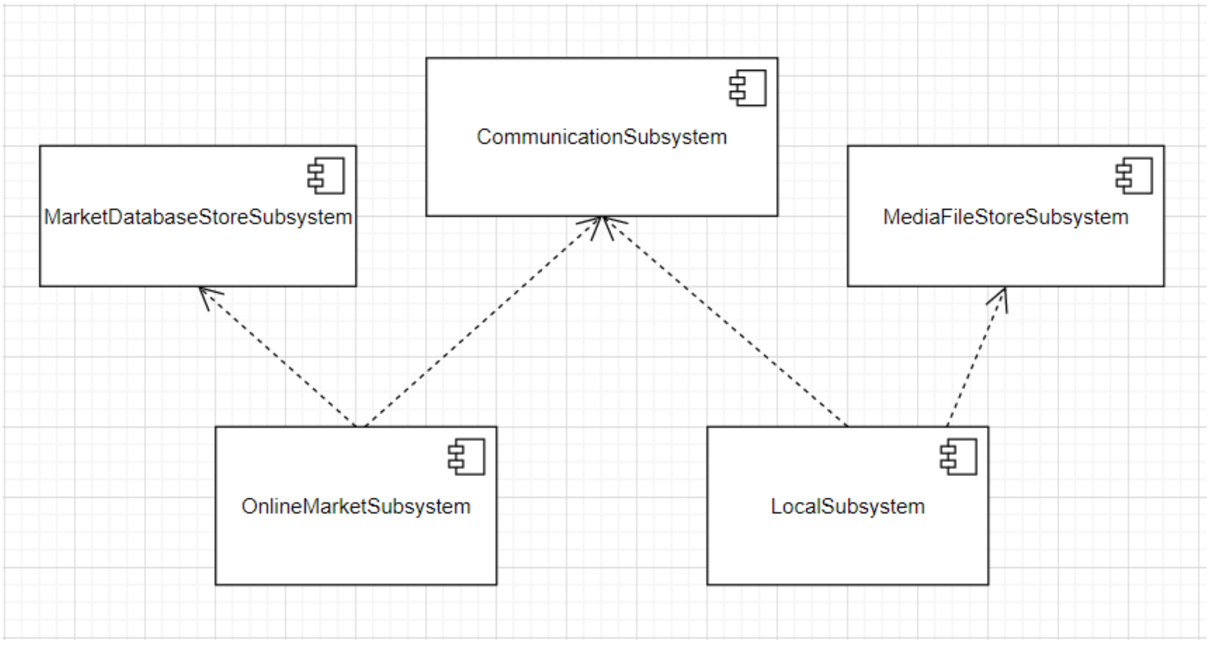
3.2 Hardware/software mapping

We must use a server machine to provide access to market and all the content in it to each user without loading up on individual users machines. Each user will only preserve the data that is related to them and access it with ease.



3.3 Persistent data management

As persistent objects, we chose PurchaseHistory and Media because we need these objects after the execution too. We can only see the informations of the medias, in the market.If we decide purchase them ,we can download these files. By downloading, we transfer them to the LocalSubsystem part, so these new files stored in filesystem.



MediaFileStoreSubsystem: The MediaFileStoreSubsytem is responsible for storing medias that already in there and medias that purchased then downloaded from the market. **MarketDatabaseStoreSubsystem**: The MarketDatabaseStoreSubsystem is responsible for storing Media in the market and PurchaseHistory.

3.4 Access control and security

	Media	**MediaCollection**	**Search**	**Market**	**Settings**

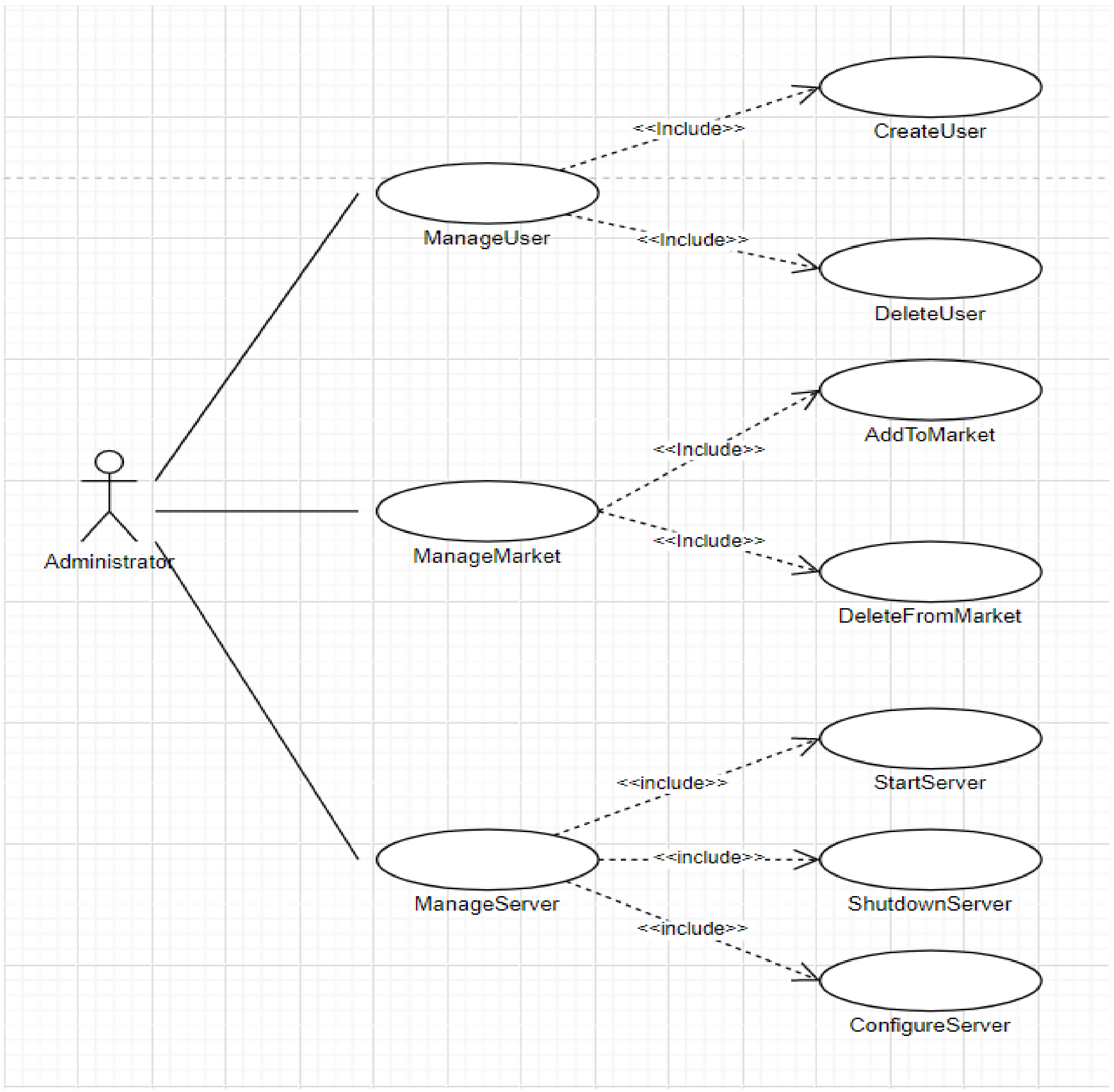
User	display()
	edit()
		display()
		edit()
			search()	view()
				purchase()
				refund()
				download()
					login()
					logout()
					signin()
					easyMode()
					offline/online()

Artist	create()
	display()
	edit()
		create()
		display()
		edit()
			search()	view()
				add()
				purchase()
				refund()
				download()
				upload()
					login()
					logout()
					signin()
					easyMode()
					offline/online()

Administrator	create()
	edit()
	display()
		create()
		edit()
		display()
			search()	view()
				edit()
				delete()
					login()
					logout()
					signin()
					easyMode()
					offline/online()

Some function names might be different from object modeling for clearance of understanding

3.5 Boundary conditions



->**ManageServer** is entitled to start, shutdown and configure the server.

->**ManageUser** is entitled to create user and delete user.

->**ShutdownServer** is entitled to stop the processing of the server.

->**StartServer** is entitled to start the processing of the server.

->**ConfigureServer** is invoked to modify settings of server.

->**CreateUser** is entitled to create a new user.

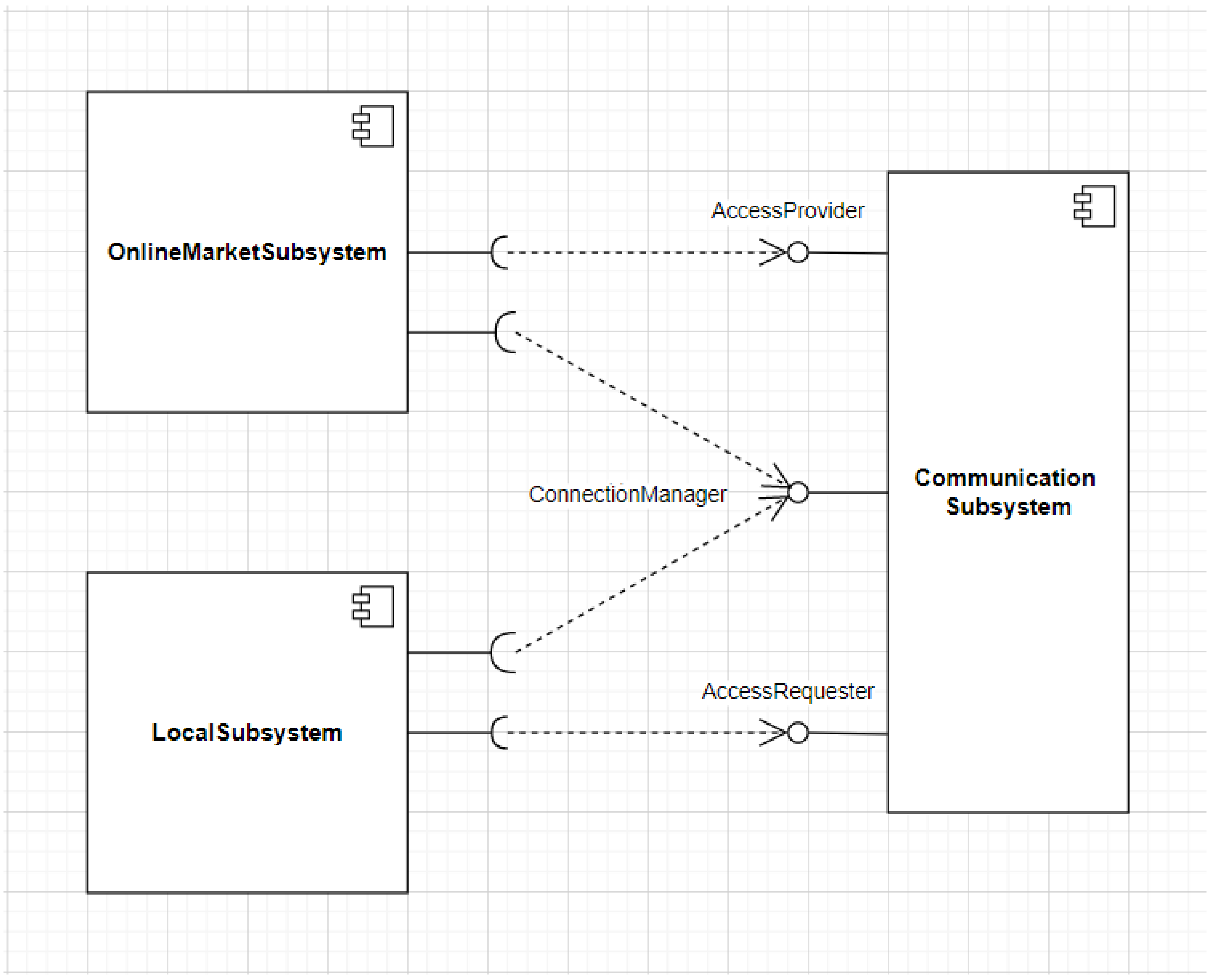
->**ManageMarket** is entitled to adding files to market and deleting from market.

->**DeleteUser** is entitled to delete a user that already exists.

->**DeleteFromMarket** is entitled to delete files when requested.

->**AddToMarket** is entitled to add files when requested

4 Subsystem Services



AccessRequester: Sends a request for uploading a content from the local storage or making a purchase from the online market.

AccessProvider: Confirms the request of the user.

5 Glossary

6 References

research.net

sidify.com

Wikipedia.com

Spotify