TAU INF303 Software Engineering Projekt

Projektdokumentation

<2020.v0.0.3>

<e-debt book>

Product Owner: BMHM34

Autoren:

Name (alphab.)	Email
Alhamza Ibrahim	e170503110@stud.tau.edu.tr
Mahasin Elderviş	e170503105@stud.tau.edu.tr
MHD Belal Aljbawi	e170503113@stud.tau.edu.tr
(TL) Muhammed Kurnasan	e170503106@stud.tau.edu.tr
Volkan Aslan	e160501126@stud.tau.edu.tr

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
0.0.1	gültig	10/10/2020	Gruppe	-
0.0.2	gültig	11/10/2020	Gruppe	MS#2
0.0.3	gultig	28/12/2020	MHD Belal Aljbawi	Zwischenabgabe

**) Sofern im Projekt nicht anders vereinbart sind folgende Statusbezeichnungen zu verwenden (in obiger Tabelle und am Deckblatt):*
Entwurf / in Review / gültig / ungültig

Dokument wurde mit folgenden Tools erstellt:

Microsoft Office Word

Lucid.app

Commented [MK1]: Link for view: INF303-e-debtbook-Projektdokumentation 0.0.1.docx

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Dieses Kapitel dient dazu den Auswahlprozess und die Entscheidungsgründe zusammenzufassen, die zum vorliegenden Projekt geführt hat!

Not everywhere in the world people have access to proper banking services, not all stores and companies can afford the bank quotes for commissions-especially small and local companies, for using the POS machines, etc., and banks do not give everyone credit cards that easily! But fortunately, the internet service is spreading very exponentially, so instead of using the old-style debt books; a note book that contains details of the debts of a person to the store, manually, where a small flaw, a little dot could make a huge difference and even possibly open a way for mistrust and problems in local societies.

And from this approach came our inspiration; to provide a digitalized, easy-to-use platform "application" that helps both market owners to organize the debt within a digital platform in a modern way in which, all the records are kept and available at the same time for both the costumer and the market owners -from anywhere and at any time!- , ease the process, save time and resources. Simply organize and manage everything with transparency and efficiency, since the phone is a device that almost everyone has, and you do not have to have many books! It also allows costumers of a store "users" to access the details of their debts to each store, and choose a method to pay, notify them for upcoming paying dates, and also make them able to pay them via some defined easier and cheaper methods.

[REQ1] Motivation ist ausführlich dargestellt, ca. ¾ Seite.

1.2 Zweck des Dokuments

This documentation will have descriptions for the functionalities of the project, the methods that are used to implement the idea of the project, the technologies and tools, and the behavior of the application, as well as a guidance for the way in which the application functions.

1.3 Begriffsbestimmungen und Abkürzungen

DB	Database
UI	User interface

#to update

Auflistung von Begriffsdefinitionen und Abkürzungen, die für das Verständnis der Grobspezifikation wichtig sind (EDV-Fachausdrücke und Begriffe aus der Anwendungsdomäne) - unabhängig davon, ob diese in einem anderen Dokument (z.B. Pflichtenheft) bereits erklärt wurden.

Beispiel:

COTS	*Commercial Off-the-shelf Software (Kaufsoftware, Standardsoftware, OSS)*
IT	*Informationstechnologie*
DD	*Deployment Diagram*
AD	*Activity Diagram*
CD	*Class Diagram*
SD	*Sequence Diagram*
UD	*Use Case Diagram*

2 Projekt Management

First sprint:

- researching for a good debugging tool:
    - **Espresso** is a very advanced debugging tool and is specialized for **android debugging**.

- researching for the to be used database
    - **Google Firebase** is easy to use, reliable, safe, and affordable.

- Choosing the programing language and suitable IDE and development tools.
    - The team will be using **Android Studio**, and has chosen **Java**, to be the programming language for **android**.
- Searching for other tools that are helpful:
    - choosing a tool for dissecting Restful API's:
        - **Postman**: is a great tool for dissecting RESTful APIs or test ones. It has a good and an understandable UI and saves a lot of time.
- Searching for more helpful Api's I:
    - **Google maps Api** can provide good geolocation data.
    - **Currently under research**.

- search for data:
    - there will be a need of **photographic**, **sound**, and **video** materials for the application. Those will be stored and could be changed.

- Installing the necessary apps:
    - All previously mentioned tools are prepared for usage.

3 Produktumfang

In this article we will describe our product's architecture explicitly using UML diagrams and detailed use cases. Using the product requires the users, debtors and loaners, ultimately to have a sustainable internet connection, at least while using, and have the software product installed, so they can perform their wished tasks after registering for an account using their GSM number. Using the GSM number is necessary and is the safest way for providing one way to find all debts of a costumer, debtor, and all given loans for one loan provider, ex. Shop, without providing any further personal information, thus making it safe and easy to use. There are many scenarios for the usage of this application, they will be addressed respectively, and we'll provide the most couple of ease to understand and substantial usages of the application, provided the user-id, the steps for the scenario, and the necessary conditions for the usage too.

3.1 Technische Infrastruktur

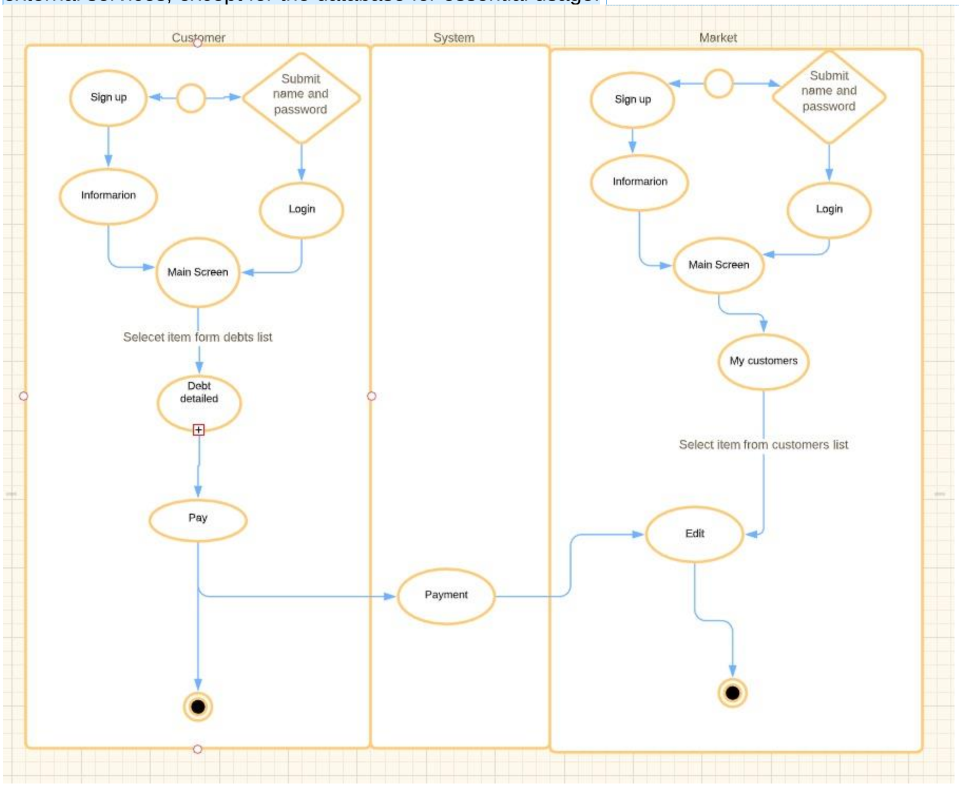
The development environment for our application is the most famous, easy to use, and practical IDE for Android, Android Studio, we'll use the most modern version of it, Android Studio 4.1 and the application will be connected to a Google Firebase database, that has a significant low latency time, and is very easy to use. The data there will be stored in the format of .json, all of this, regarding the database connection and state, makes programming the application also easy and faster too, as much it will provide an approximately real-time update of the database. There will be more discuss about the product's architecture and use cases in the following descriptive chapters.

3.2 Anforderungsanalyse / Use Cases

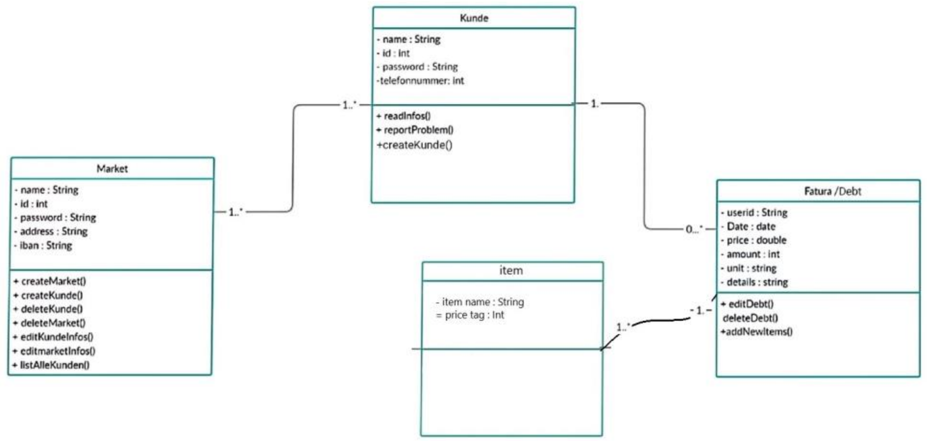
Was sind die wichtigsten Anwendungsfälle / Use Cases des < e-debt books>?

Our system will have two use cases; a user as a loaner and another as a debtor, that will be performing their tasks respectively. The loan provider, debtor, will be able to register, sign in, add debts, and approve paid debts etc. - will be addressed in a detail- and the loaner will be bale to register, sign in, see the loans and pay- will also be addressed in detail-. There will be no needs for external services, except for the database for essential usage.

Commented [MK2]: Could be changed later.



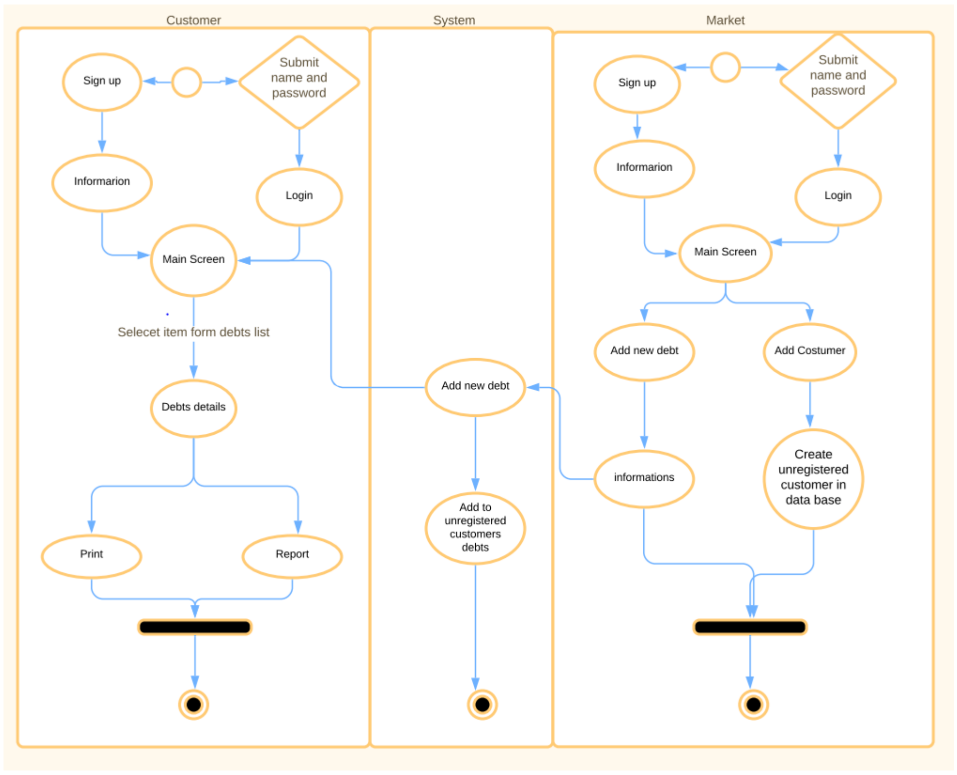
Was sind die wichtigsten Datenstrukturen / Use Cases des <Produkts>?



Our product will provide many use cases, here's a description of the two most essentials:

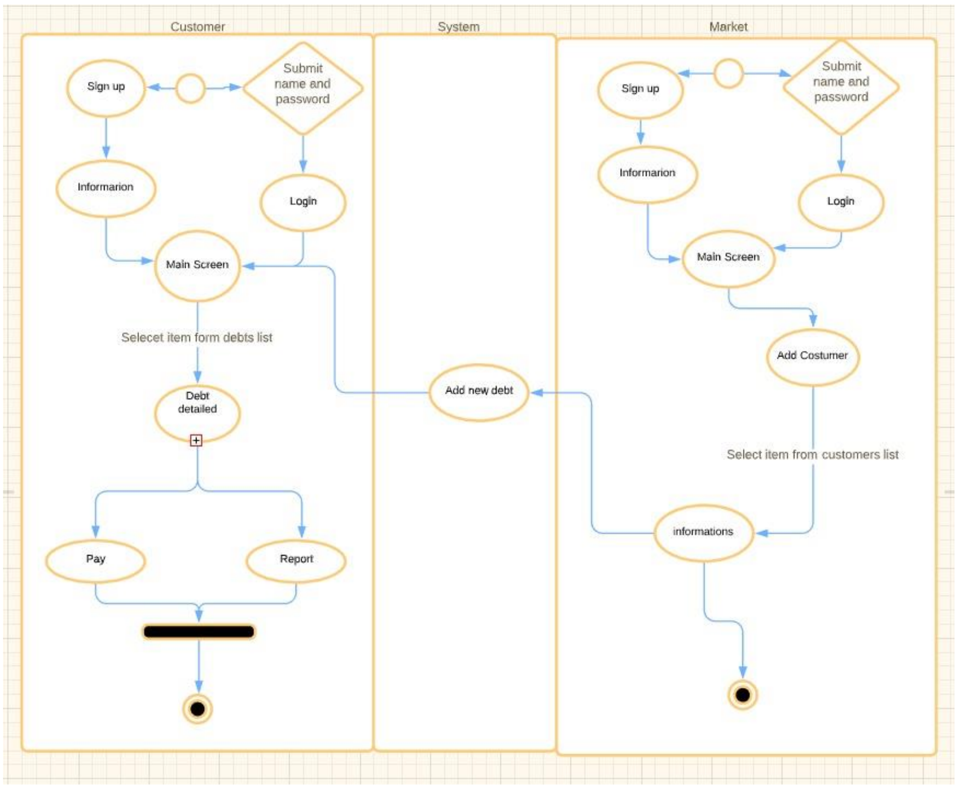
1. the debtor fulfils a debt; performs a payment:

<UseCaseID>	**<Name>**	**<Beschreibung>**
scenario	Performing a payment	
condition	Regestration, loan	• a debtor must have a record, signed up. • A debt must be added by loaner user.
Step 1	Sign in	The debtor must sign in from the main screen using his/her profile credentials.
Step 2	Go to the debt list menu	The debts list menu provides the details of any confirmed debt registered by a loaner
Step 3	Perform the payment of a selected debt/loan	The user-debtor- selects a specific debt, and performs a payment.



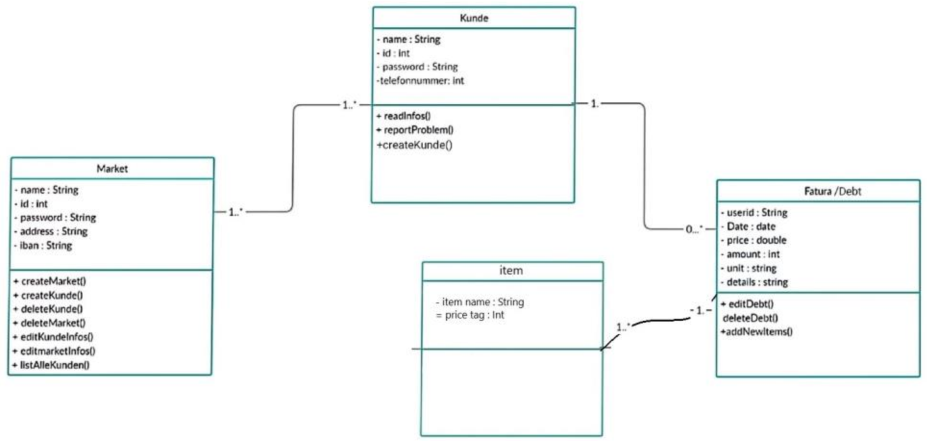
2. the loan provider wishes to list all debts/loans he has given:

<UseCaseID>	**<Name>**	**<Beschreibung>**
scenario	Listing debts	
condition	Regestration, loan a debtor	• a loaner must have a record, signed up. • A debt must be added by the loaner user.
Step 1	Sign in	The loaner must sign in from the main screen using his/her profile credentials.
Step 2	Add a customer menu	Create an account with the GSM number of the new customer/ debtor, that has no account yet.
Step 3	Add a debt for this account.	Register a debt for this user.
Step 4	List the debts	Lists all debts



3.3 Architektur

The following graphic describes the classes that will be used in the implementation of the project, from an object oriented perspective, also in the database, there will be three classes, one for the users labeled as loaners, the other for the users labeled as debtors, and finally a class for the bills as followed:



Use cases with tables

Main Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	User of the system	Market Login	➢ Direct the user to the Market Login Screen.
2	User of the system	Customer Login	➢ Direct the user to the customer login screen.

Market Login Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	Market	Login	➢ Check the market's email and password. Whether the entries are correct lead the user to the Market Main Screen. Whether the inputs are incorrect, return an error. ➢ If desired, keep the user signed in.
2	Market	Sign up	➢ Direct the user to the Market Sign Up Screen.

Market Sign Up Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	Market	Sign up	➢ Name, phone number, password, Iban number and address information of the user are requested. ➢ The system checks if there is another account that has already signed up with the same Iban number. ➢ If yes, the system returns an error. ➢ If no, the market's information will be saved to the database and the user will be directed to the Market Main Screen.

Market Main Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	Market	Add Customer	➢ Direct the user to the Add Customer Screen.
2	Market	Add Debt	➢ Search for customer using phone number ➢ Add debt details , product , price and amount to the bill ➢ Add bill to the linked data base ➢ Print bill as PDF
3	Market	Debt list	➢ List all the customers of the market. ➢ The market can edit, delete and print the information of any customer from the list.
4	Market	Settings	➢ Direct the user to Settings Screen. ➢ Edit Market Information ➢ Change language
5	user	About us	➢ Information about the project and the developers
6	User	Market Profile	➢ Edit the user's data on the database according to his old Phone number -if changed- and direct the user to the Main Market Screen

Add unregistered Customer Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	Market	Done	➢ Phone number and first debt amount information of the customer are requested. ➢ Save the debt info to database and send a notification to the customer.

Customer Login Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	Customer	Log in	➢ Check the customer’s phone number and password. Whether the entries are correct lead the user to the Customer Main Screen. Whether the inputs are incorrect, return an error. ➢ If desired, keep the user signed in.

Debt Screen Use Cases:

Srl No	Actors	User Case Name	Goals
1	Market	Add debt	➢ Search for customer using phone number ➢ Add debt details , product , price and amount to the bill ➢ Add bill to the linked data base
2	Market	Print	➢ Print bill as PDF ➢ Save PDF to device

Customer Sign Up Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	Customer	Sign Up Done	➢ Name, last name, password, phone number and email of the user are requested. ➢ The system checks if another customer has been already added with the same phone number. ➢ If yes, the system returns an error. ➢ If no, information of the customer will be saved into the database. ➢ Direct the user to Customer Main Screen.

Customer Main Screen Use Cases:

Srl No	Actors	User Case Name	Goals
1	Customer	(Click on any item of the debts list)	➢ Direct the user to Debt Screen and display the debt and the market info.

Debt Screen Use Cases:

Srl No	Actors	Use Case Name	Goals
1	Customer	print	➢ Download bill as PDF file .
2	Customer	Report	➢ Send a email to the market about the report.

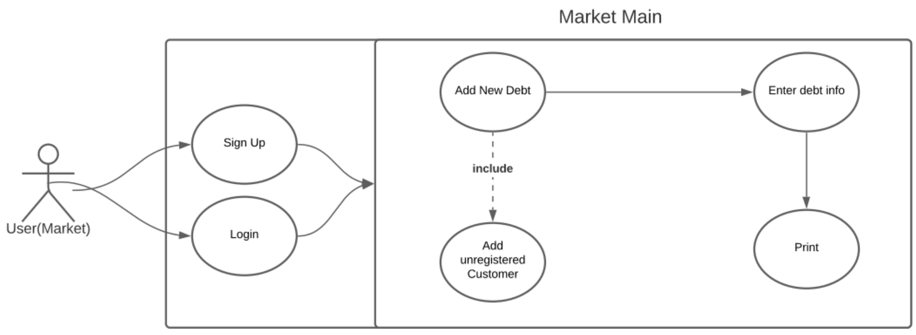
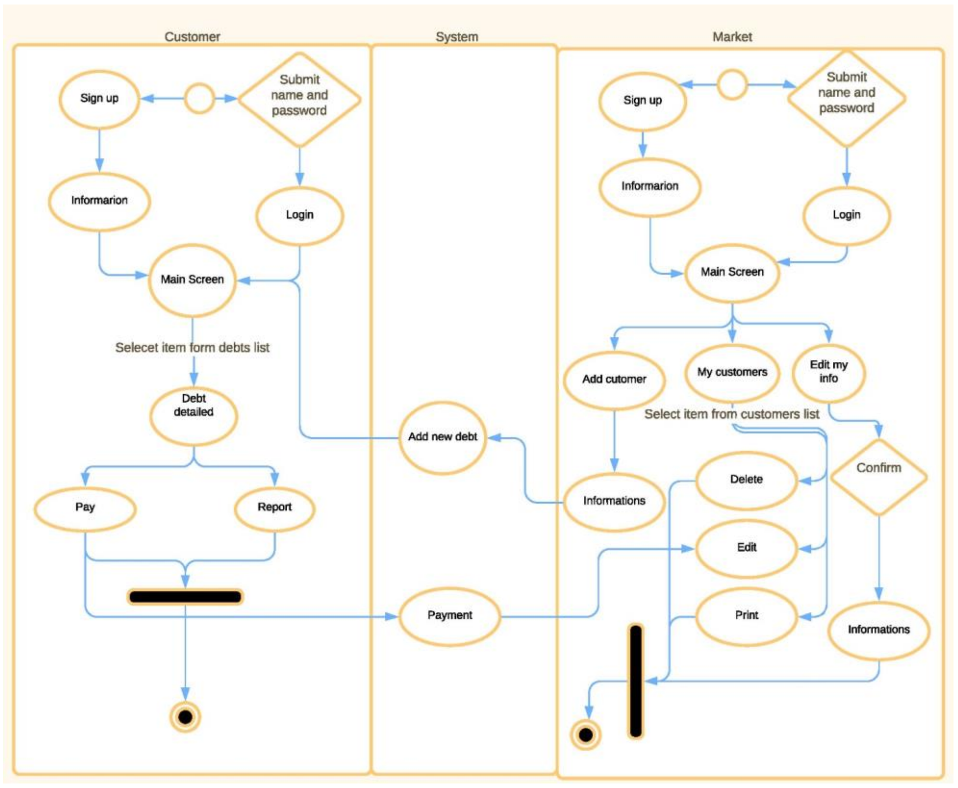
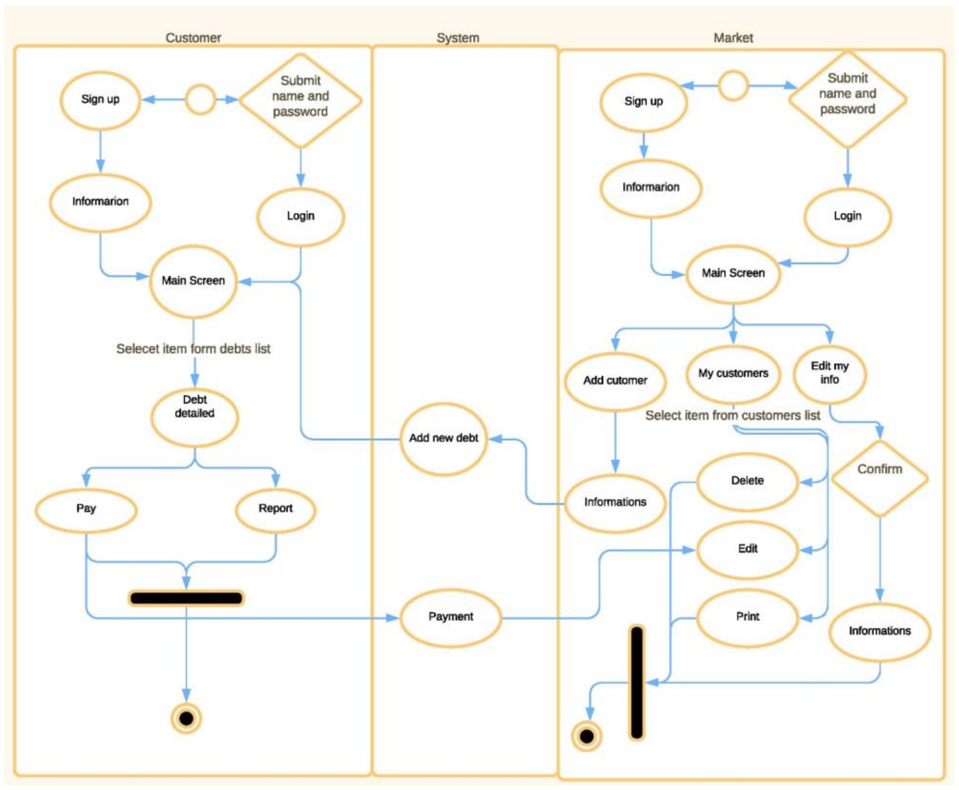


Tabelle 3-1: a UML table for activity



3.4 Deployment View

All the operations performed and labeled under system, market, and customer will be performed locally, and their actions will be saved in the data base, when necessary on the google firebase servers. As already mentioned, the users must maintain the intern



All the operations performed and labeled under system, market, and customer will be performed locally, and their actions will be saved in the data base, when necessary on the google firebase servers. As already mentioned, the users must maintain the internet connection that enables the program to connect to Google firebase for performing any action, other way the software will have no functionality other than the local database's services, if existed.

4 Teststrategie und Testplanung

Die oberste Priorität ist es, die definierten Use Cases des Produkts zu validieren. Hier soll es beschrieben werden wie das Produkt getestet wird.

[REQ4] Als Projektanforderung soll es für jeden Use Case mindestens einen Test Case definiert, implementiert und durchgeführt werden.

test_marketRegister () : launch main activity class > click on market butoon > check if market window is displayed > click on sign up button > check if market regester window is displayed > fill name, email,phone,passowrd, iban and adress fields > click on sign up button

test_customerRegister(): launch main activity class > click on customer butoon > check if customer window is displayed > click on sign up button > check if customer regester window is displayed > fill name, last name,phone,passowrd and email fields > click on sign up button > chick if verfication window is displayed > enter verification code > click verify button

test_MarketLogin () launch main activity class > click on Market butoon > check if Market window is displayed > fill email and passowrd fields > click on log in button > check if MarketHome window is displayed.

test_customerLogin () launch main activity class > click on customer butoon > check if customer window is displayed > fill phone and passowrd fields > click on log in button check if CustomerHome window is displayed.

test_addNewDebt() : launch main activity class > click on Market butoon > check if Market window is displayed > fill email and passowrd fields > click on log in button > check if MarketHome window is displayed. > click on add newdebt button > check if add debt window is displayed > enter the desired phone number > click on search button > check if the user exist or not > enter debt details , amount price and date > click on add debt button.

Test_printBill() launch main activity class > click on customer butoon > check if customer window is displayed > fill phone and passowrd fields > click on log in button > check if CustomerHome window is displayed > click on bill from bills list > check if bill is displayed > click on print button.

Test_change language() launch main activity class > click on customer butoon > check if customer window is displayed > fill phone and passowrd fields > click on log in button > check if CustomerHome window is displayed > click on langauge button from drawer > choose the desired language > click on done button.