CASHIERY PRJECT DOCUMENT

TEAM MEMBERS

ALAA ADEL ABDEL HAFEZ	20140065
ESRAA GAMAL RAGAB	20140052
ESRAA RGAB ABDEL SATTAR	20140053
AYA FAWKY ABDEL ATTY	20140086

Supervised by:
Dr. Osama Ismaiel
TA. Ahmed Ramzy

June 2018

CASHIERY
Just scan it

TABLE OF CONTENTS

CONTENTS PAGE

CASHIERY
Just scan it

CASHIERY
Just scan it

1-System Proposal

1-1 Background

We have to make best of customer's data, we will get thousands or millions of customer's data that is not used in its optimal exploitation during physical transactions.

Our application 'Cashiery' first will exploit the customer's data to make profile for each customer which will be useful to recommend for each customer or for a group of customers the suitable products and offers that can be bought during shopping, second the application will provide online payment method, third the ability to control the budget through knowing the total cost of the shopping cart instead of being shocked at the point of sale.

1-2 Motivation

As a customer to any hypermarket will be interested to use 'Cashiery' application because it provides online payment, self-scan property, virtual online shopping cart with its total cost, show recommendations, and easy access to the hypermarket offers. This occur only when customer make an account on the application. In this project we will develop the application which help the clerk of hypermarkets in calculations/accounting, make mining on the customers' transactions to help them with providing recommendations.

1-3 Problem Statement

Thousands or millions of customer's data is not exploited as an optimal exploitation during physical transactions. Especially in hypermarkets which be exploited by one side who be hypermarket whereas another side (every single customer) not have an optimal benefit from this data. Hypermarkets don't store favorite items or products for each customer from its daily transaction when comes to hypermarket.

1-4 Scope

The domain of our application is focus on profiling data for every customer in hypermarket, providing recommendations to other items related to them, also provide online payment and show the offers for selected hypermarket.

1-5 Project Objective

The main goal of our application is to optimize the usage of data by making a profile for every user and recommend offers based on his needs, facilities the purchasing order in hypermarkets, increase the customer's loyalty and satisfaction increasing the firm's profit by increasing the sales.

1-6 Methodology

We using *agile approach*, which is quick and iterative, the tasks are divided into short sprints/scrums of work, frequent assessment and adaptation to plans,

The reasons to choose agile approach/development:

(1) Agile methodologies have proved efficient and helpful to mobile app development environment. It fits these characteristics appropriately as it is more flexible, while traditional methods are costly and there is very less scope of change.

(2) Testing can be performed at each stage.

(3) Agile development makes the app more stable with fewer errors, thus increasing the quality.

1-7 Solution Statement

Our application will exploit customers' data which presented in each transaction for every customer by making personal profile to each customer, and providing recommendations for each single user based on his needs, to help them to buy items by recommending other items related to them in hypermarkets in a small amount of time and it's also good for world of speed as the customer help himself, also provide online payment and can show offers for selected hypermarket.

1-8 Techniques and Tools

This project gives us the opportunity to explore and learn android, development the mobile applications, using spring boot as a framework.

Our knowledge of this field at the beginning of the project was limited, but our base knowledge was in java language only and limited knowledge with spring technology band not treat with android, but we find out that android provides a rich application framework that allow to us to build innovative apps and games for mobile devices in a Java language environment.

We learning that (1) android apps are built as a combination of distinct components that can be invoked individually. For instance, an individual *activity* provides a single screen for a user interface, and a *service* independently performs work in the background, (2) Android provides an adaptive app framework that allows you to provide unique resources for different device configurations. For example, you can create different XML layout files for different screen sizes and the system determines which layout to apply based on the current device's screen size (3) Spring boot allow us to convert the backend code into services that we can call in the android we perform the function we need .

1-9 Related work

Shopping Cart	- This application works as a smart calculator, you simply put in what you're going to buy and its price, and the app will does the rest for you! When you're done with your shop, you can either delete your list or keep it there for some future consultation. All you need is to sign in using your google account.
Shopping list – buy me pie	- This application allow more than one person to share the list and if one add an item to the shopping list it will be updated automatically in another list, and this is the smartest way to write a shopping list.

Bring! Grocery shopping list	- here one person create a shopping list and share it that he is shopping now and there's number of persons share this together any of them can add items to this list and the first person (the owner of the original list) can see the list updates.
Shopping list	- This is useful and simple way of making out a shopping list on your phone. You can input items from keyboard, add from database, filled by you, by barcode scanning and even using your voice! Also, you can mark out important purchases in the list. You just have to tap an item to mark it as "bought".
Out of milk – grocery shopping list	- With Out of Milk, your Shopping List stays with you everywhere you go that you can scan the things that you need to buy anytime and you'll have it on-hand once you're ready to go grocery shopping. The Pantry List allows you to keep track of your pantry items (spices, essentials, etc...) so that you always know what you have at home. The To-Do list helps you keep track of any other items on your daily list.
Grocery shopping list – Listonic	- With this application one person can create a shopping list and share it with another person and add items too if needed.
Shopping list voice input	- This application allows create the list using voice records.

Our groceries shopping list	- This application keep track of a family shopping list in shared with all family members' phones even it's an IPhone, it enables share receipts, images for the items, add items by scanning the barcodes and add items using amazon.

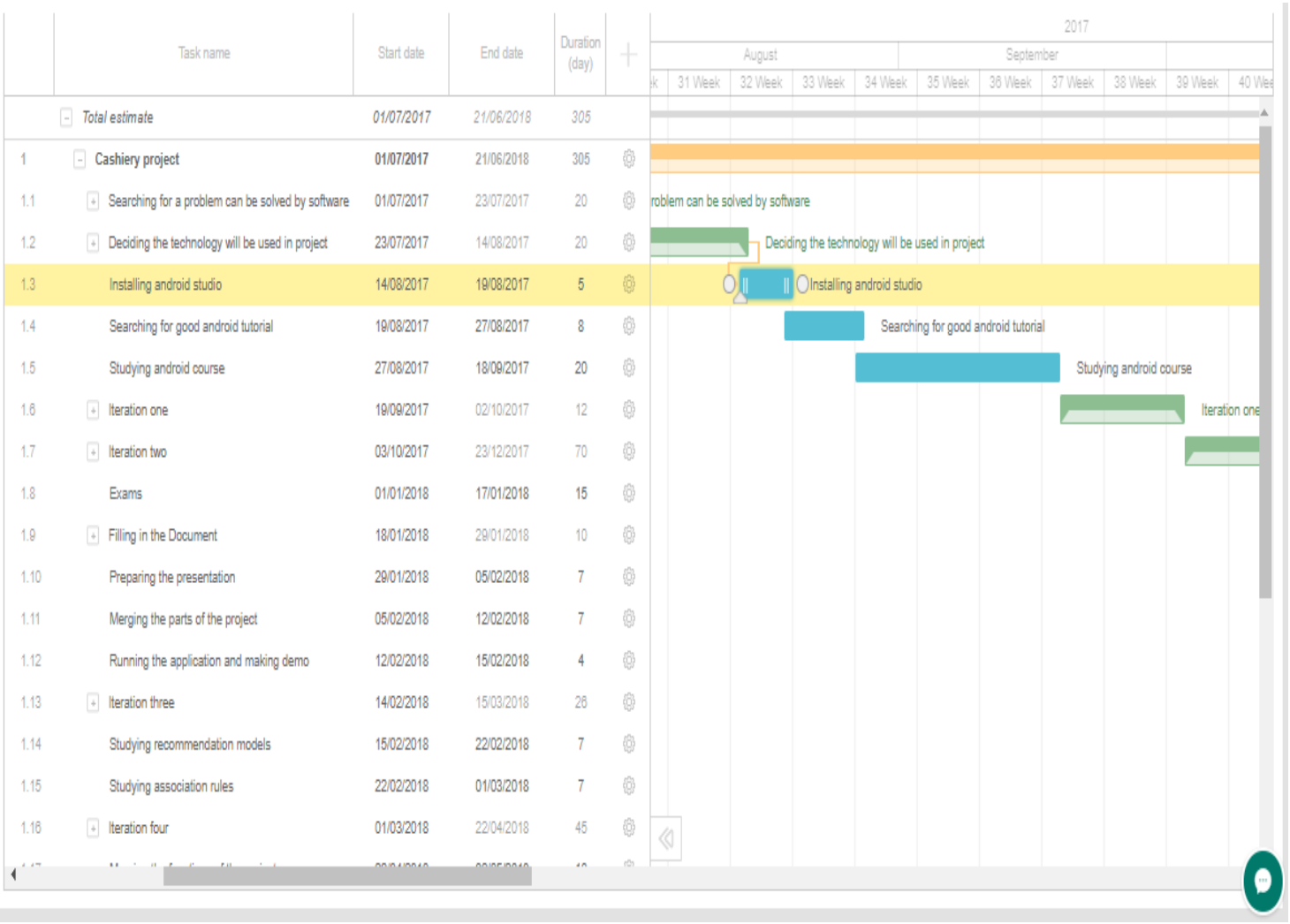
1-10 Project Management

1-10-1 SWOT Analysis

Strength • Offering recommendation for each specific customer. • Control budge of shopping cart. • Optimize the usage of customers' data. • Available for android mobile phones are used commonly among different clients. • Offering online payment.	Weakness • Support only one platform "android platform".
Opportunities • Smart/android mobile phones are used by different clients. • Support online payments for hypermarkets.	Threats • May be connect to server of hypermarkets (to access database to retrieve products details for selected hypermarket) fail, this a big threat For us with this application.

1-10-2 Plan and Gantt chart

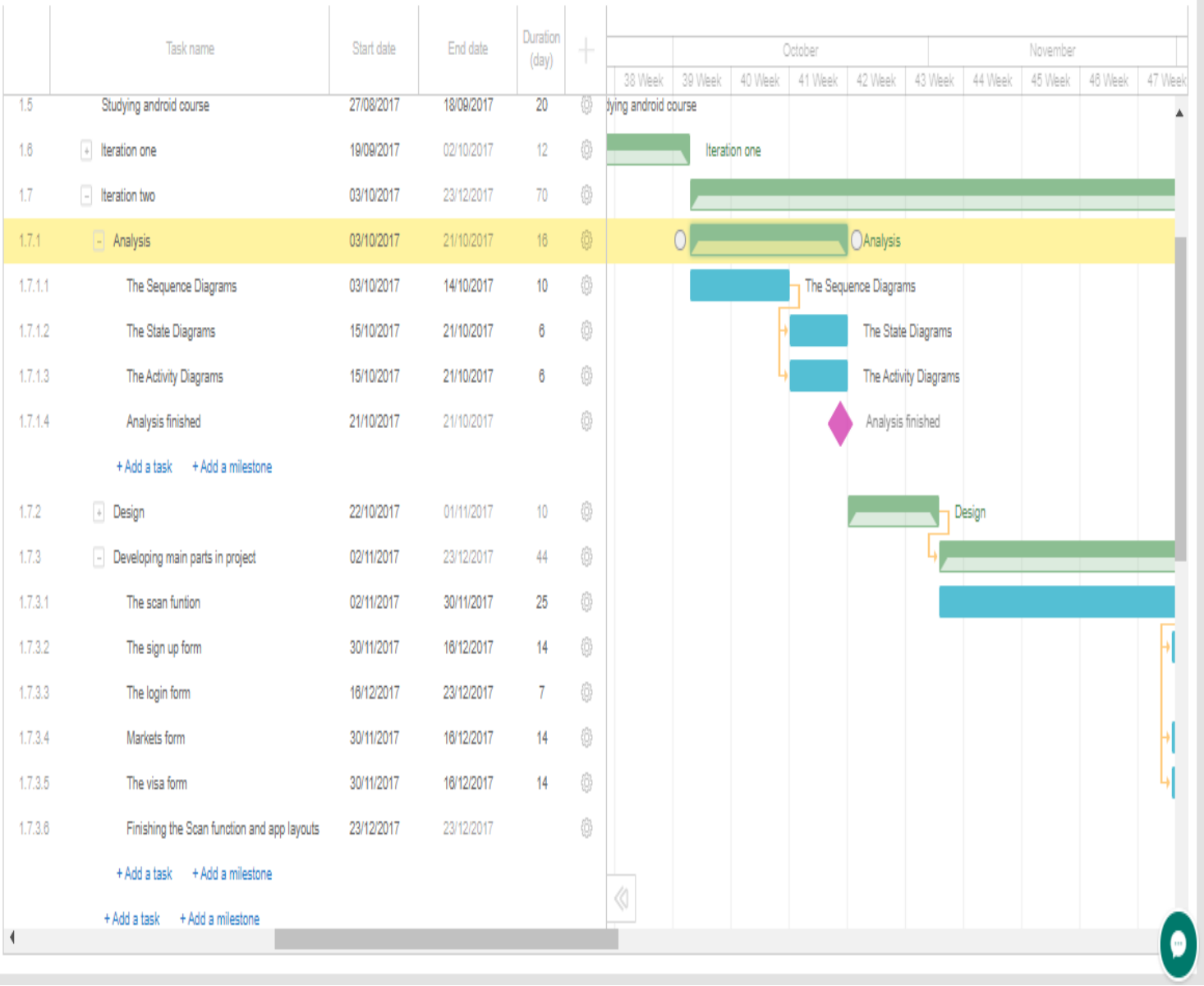
	Task name	Start date	End date	Duration (day)
	Total estimate	*01/07/2017*	*21/06/2018*	*305*
1	Cashiery project	01/07/2017	21/06/2018	305
1.1	Searching for a problem can be solved by software	01/07/2017	23/07/2017	20
1.1.1	Searching in medical field	01/07/2017	23/07/2017	20
1.1.2	Searching in commercial field	01/07/2017	23/07/2017	20
1.1.3	Searching in fashion field	01/07/2017	23/07/2017	20
1.1.4	Searching in entertainment field	01/07/2017	23/07/2017	20
1.1.5	Choose the idea of the project	20/07/2017	20/07/2017	
	+ Add a task + Add a milestone			
1.2	Deciding the technology will be used in project	23/07/2017	14/08/2017	20
1.2.1	Reading about .net technology	23/07/2017	14/08/2017	20
1.2.2	Reading about IOS technology	23/07/2017	14/08/2017	20
1.2.3	Reading about Android technology	23/07/2017	14/08/2017	20
1.2.4	Settle on a technology	14/08/2017	14/08/2017	
	+ Add a task + Add a milestone			
1.3	Installing android studio	14/08/2017	19/08/2017	5
1.4	Searching for good android tutorial	19/08/2017	27/08/2017	8
1.5	Studying android course	27/08/2017	18/09/2017	20



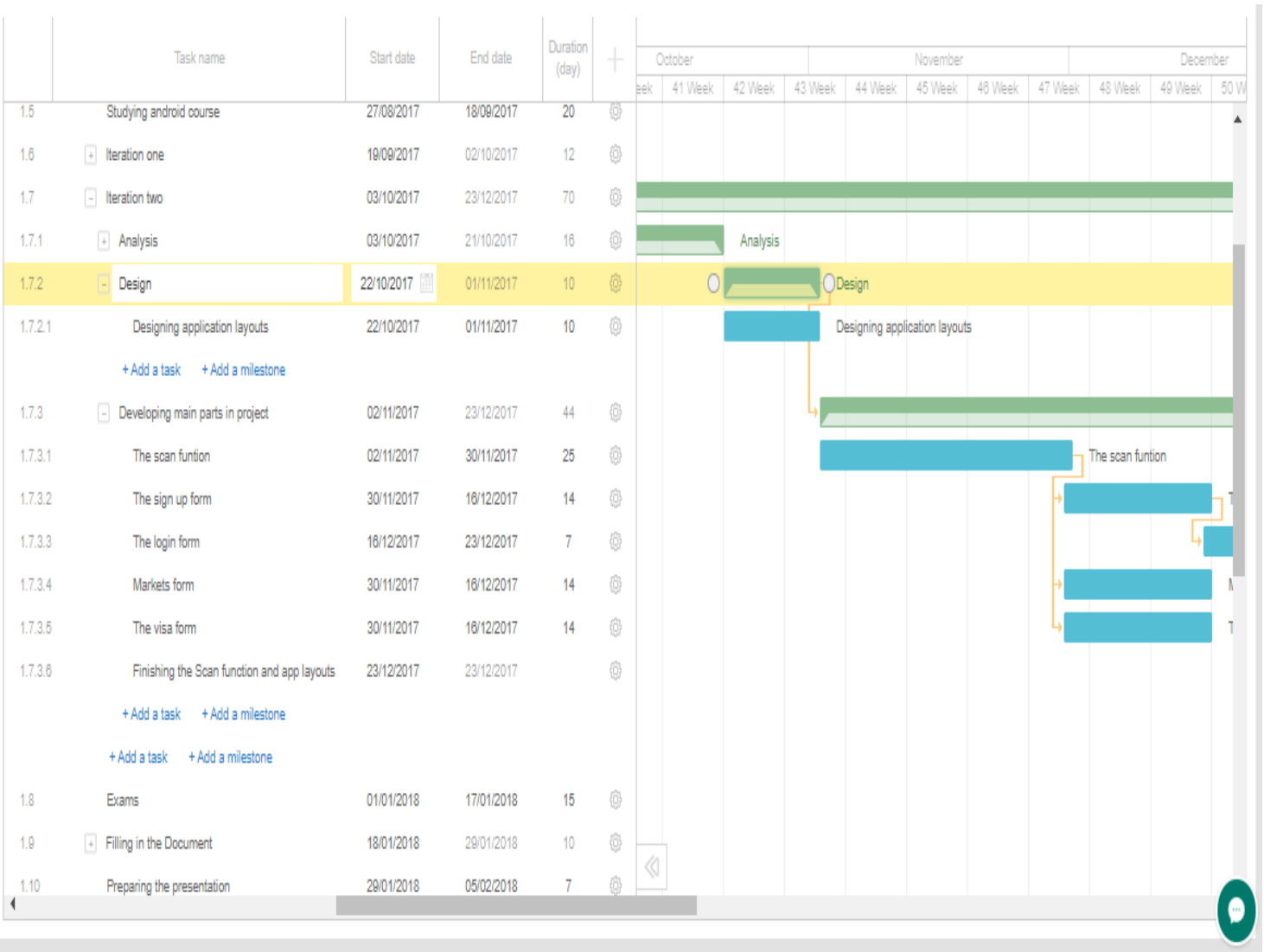
	Task name	Start date	End date	Duration (day)
	Total estimate	*01/07/2017*	*21/06/2018*	*305*
1	Cashiery project	01/07/2017	21/06/2018	305
1.1	Searching for a problem can be solved by software	01/07/2017	23/07/2017	20
1.2	Deciding the technology will be used in project	23/07/2017	14/08/2017	20
1.3	Installing android studio	14/08/2017	19/08/2017	5
1.4	Searching for good android tutorial	19/08/2017	27/08/2017	8
1.5	Studying android course	27/08/2017	18/09/2017	20
1.6	Iteration one	19/09/2017	02/10/2017	12
1.7	Iteration two	03/10/2017	23/12/2017	70
1.8	Exams	01/01/2018	17/01/2018	15
1.9	Filling in the Document	18/01/2018	29/01/2018	10
1.10	Preparing the presentation	29/01/2018	05/02/2018	7
1.11	Merging the parts of the project	05/02/2018	12/02/2018	7
1.12	Running the application and making demo	12/02/2018	15/02/2018	4
1.13	Iteration three	14/02/2018	15/03/2018	26
1.14	Studying recommendation models	15/02/2018	22/02/2018	7
1.15	Studying association rules	22/02/2018	01/03/2018	7
1.16	Iteration four	01/03/2018	22/04/2018	45

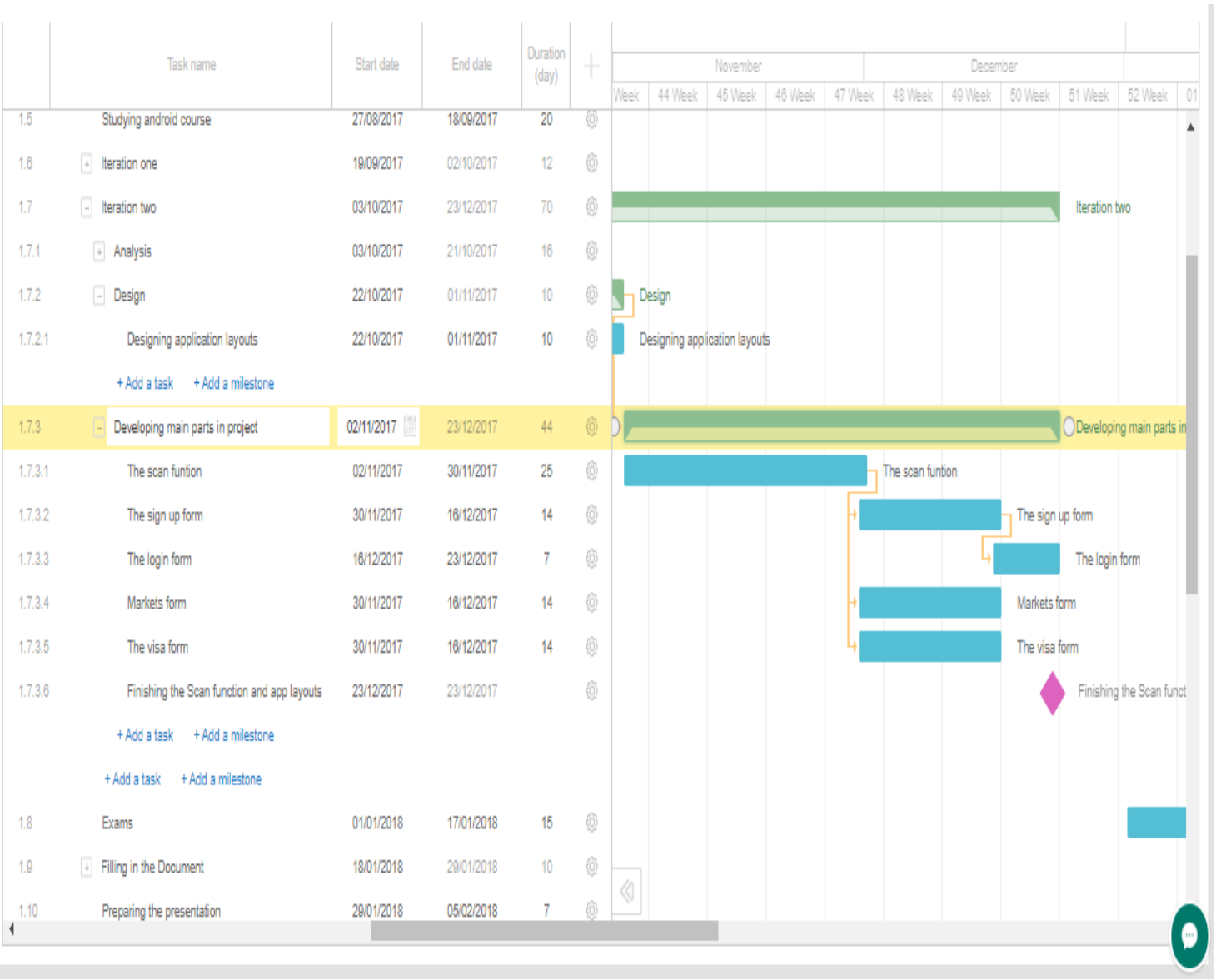
CASHIERY
Just scan it

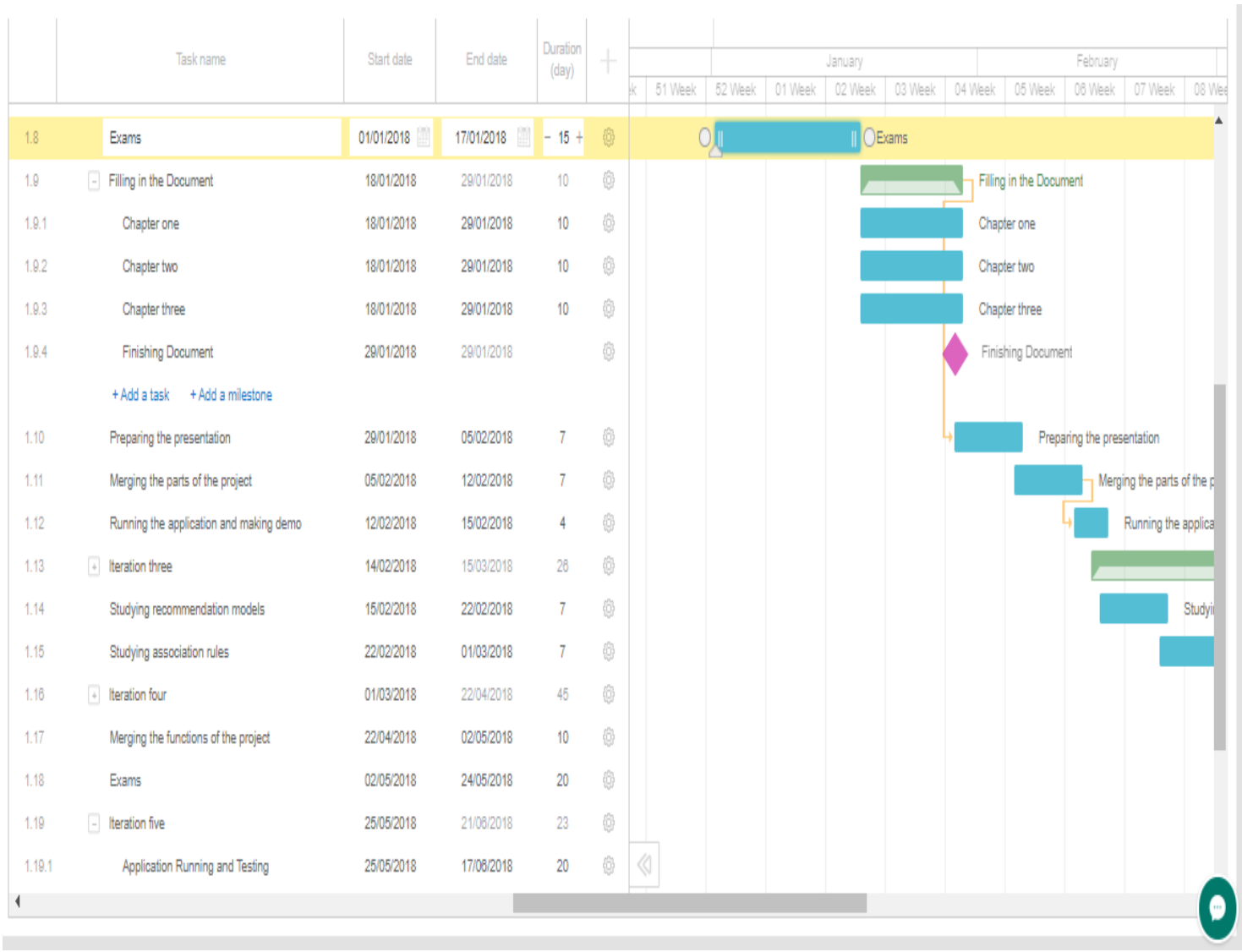
	Task name	Start date	End date	Duration (day)
	Total estimate	*01/07/2017*	*21/06/2018*	*305*
1	Cashiery project	01/07/2017	21/06/2018	305
1.1	Searching for a problem can be solved by software	01/07/2017	23/07/2017	20
1.2	Deciding the technology will be used in project	23/07/2017	14/08/2017	20
1.3	Installing android studio	14/08/2017	19/08/2017	5
1.4	Searching for good android tutorial	19/08/2017	27/08/2017	8
1.5	Studying android course	27/08/2017	18/09/2017	20
1.6	Iteration one	19/09/2017	02/10/2017	12
1.6.1	Analysis	19/09/2017	02/10/2017	12
1.6.1.1	The ER Diagram	19/09/2017	25/09/2017	6
1.6.1.2	The Class Diagram	26/09/2017	02/10/2017	6
	+ Add a task + Add a milestone			
1.6.2	Design	19/09/2017	01/10/2017	10
1.6.2.1	Designing application logo	19/09/2017	01/10/2017	10
1.6.2.2	Designing application slogan	19/09/2017	01/10/2017	10
	+ Add a task + Add a milestone			
1.6.3	Iteration one finished	02/10/2017	02/10/2017	
	+ Add a task + Add a milestone			



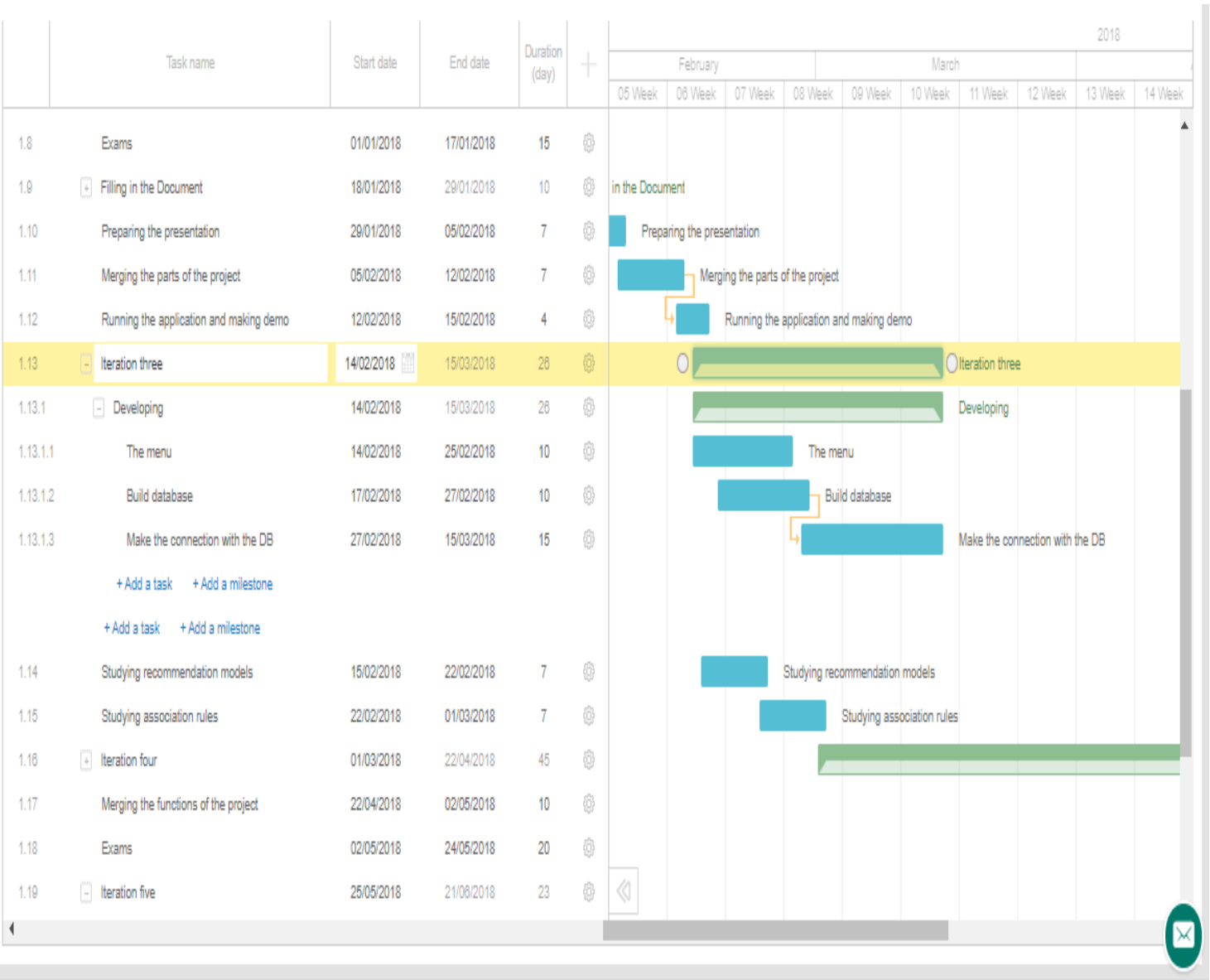
	Task name	Start date	End date	Duration (day)
1.5	Studying android course	27/08/2017	18/09/2017	20
1.6	Iteration one	19/09/2017	02/10/2017	12
1.7	Iteration two	03/10/2017	23/12/2017	70
1.7.1	Analysis	03/10/2017	21/10/2017	16
1.7.1.1	The Sequence Diagrams	03/10/2017	14/10/2017	10
1.7.1.2	The State Diagrams	15/10/2017	21/10/2017	6
1.7.1.3	The Activity Diagrams	15/10/2017	21/10/2017	6
1.7.1.4	Analysis finished	21/10/2017	21/10/2017	
	+ Add a task + Add a milestone			
1.7.2	Design	22/10/2017	01/11/2017	10
1.7.3	Developing main parts in project	02/11/2017	23/12/2017	44
1.7.3.1	The scan funtion	02/11/2017	30/11/2017	25
1.7.3.2	The sign up form	30/11/2017	16/12/2017	14
1.7.3.3	The login form	16/12/2017	23/12/2017	7
1.7.3.4	Markets form	30/11/2017	16/12/2017	14
1.7.3.5	The visa form	30/11/2017	16/12/2017	14
1.7.3.6	Finishing the Scan function and app layouts	23/12/2017	23/12/2017	
	+ Add a task + Add a milestone			
	+ Add a task + Add a milestone			


Task name
Start date
End date
Duration (day)
October
November
December
41 Week
42 Week
43 Week
44 Week
45 Week
46 Week
47 Week
48 Week
49 Week
1.5 Studying android course 27/08/2017 18/09/2017 20
1.6 Iteration one 19/09/2017 02/10/2017 12
1.7 Iteration two 03/10/2017 23/12/2017 70
1.7.1 Analysis 03/10/2017 21/10/2017 16
1.7.2 Design 22/10/2017 01/11/2017 10
1.7.2.1 Designing application layouts 22/10/2017 01/11/2017 10
+ Add a task + Add a milestone
1.7.3 Developing main parts in project 02/11/2017 23/12/2017 44
1.7.3.1 The scan funtion 02/11/2017 30/11/2017 25
1.7.3.2 The sign up form 30/11/2017 16/12/2017 14
1.7.3.3 The login form 16/12/2017 23/12/2017 7
1.7.3.4 Markets form 30/11/2017 16/12/2017 14
1.7.3.5 The visa form 30/11/2017 16/12/2017 14
1.7.3.6 Finishing the Scan function and app layouts 23/12/2017 23/12/2017
+ Add a task + Add a milestone
+ Add a task + Add a milestone
1.8 Exams 01/01/2018 17/01/2018 15
1.9 Filling in the Document 18/01/2018 29/01/2018 10
1.10 Preparing the presentation 29/01/2018 05/02/2018 7
Analysis
Design
Designing application layouts
The scan funtion

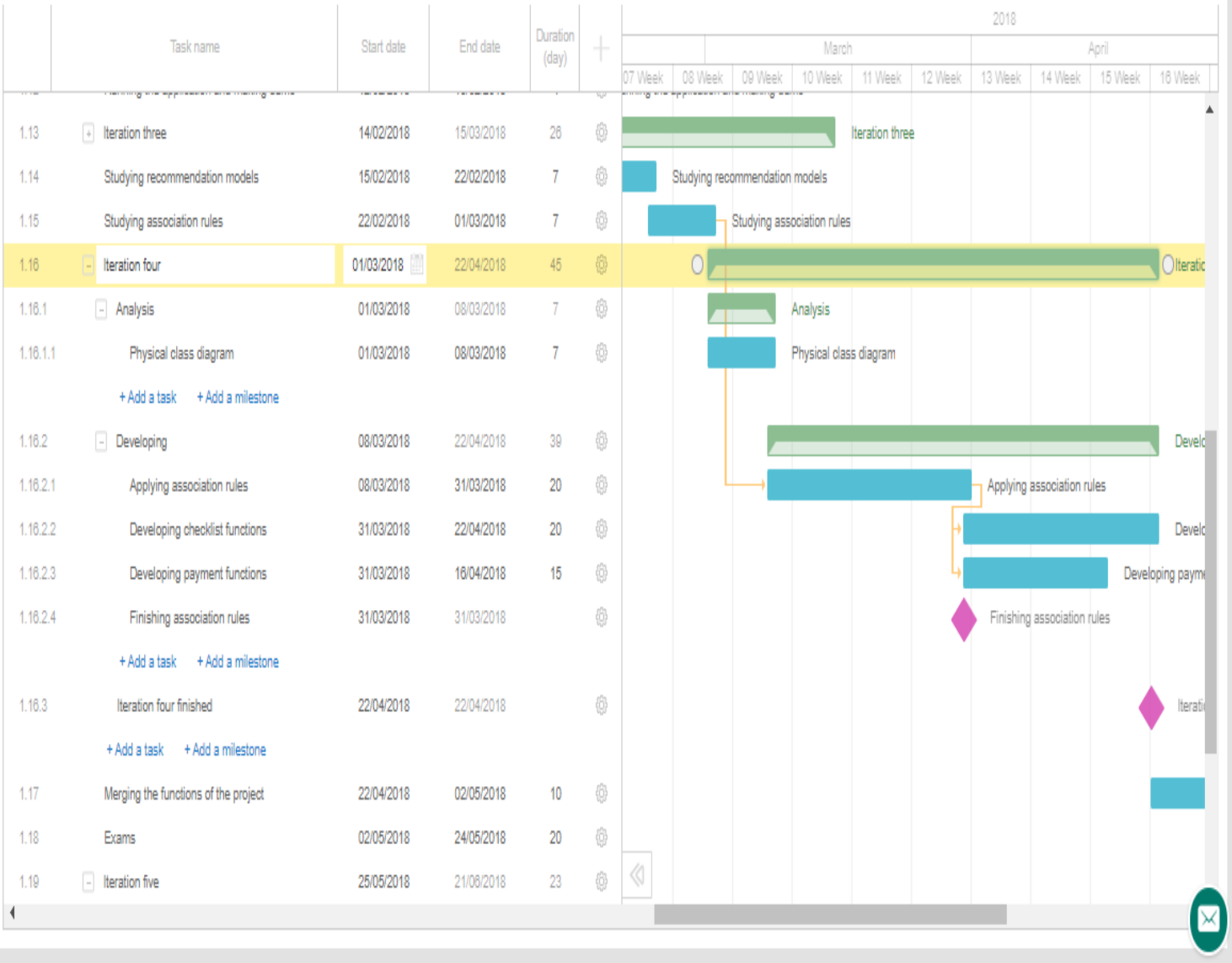

Task name
Start date
End date
Duration (day)
November
December
Week
44 Week
45 Week
46 Week
47 Week
48 Week
49 Week
50 Week
51 Week
52 Week
1.5 Studying android course 27/08/2017 18/09/2017 20
1.6 Iteration one 19/09/2017 02/10/2017 12
1.7 Iteration two 03/10/2017 23/12/2017 70
1.7.1 Analysis 03/10/2017 21/10/2017 16
1.7.2 Design 22/10/2017 01/11/2017 10
1.7.2.1 Designing application layouts 22/10/2017 01/11/2017 10
+ Add a task + Add a milestone
1.7.3 Developing main parts in project 02/11/2017 23/12/2017 44
1.7.3.1 The scan funtion 02/11/2017 30/11/2017 25
1.7.3.2 The sign up form 30/11/2017 16/12/2017 14
1.7.3.3 The login form 16/12/2017 23/12/2017 7
1.7.3.4 Markets form 30/11/2017 16/12/2017 14
1.7.3.5 The visa form 30/11/2017 16/12/2017 14
1.7.3.6 Finishing the Scan function and app layouts 23/12/2017 23/12/2017
+ Add a task + Add a milestone
+ Add a task + Add a milestone
1.8 Exams 01/01/2018 17/01/2018 15
1.9 Filling in the Document 18/01/2018 29/01/2018 10
1.10 Preparing the presentation 29/01/2018 05/02/2018 7
Iteration two
Design
Designing application layouts
Developing main parts in
The scan funtion
The sign up form
The login form
Markets form
The visa form
Finishing the Scan funct

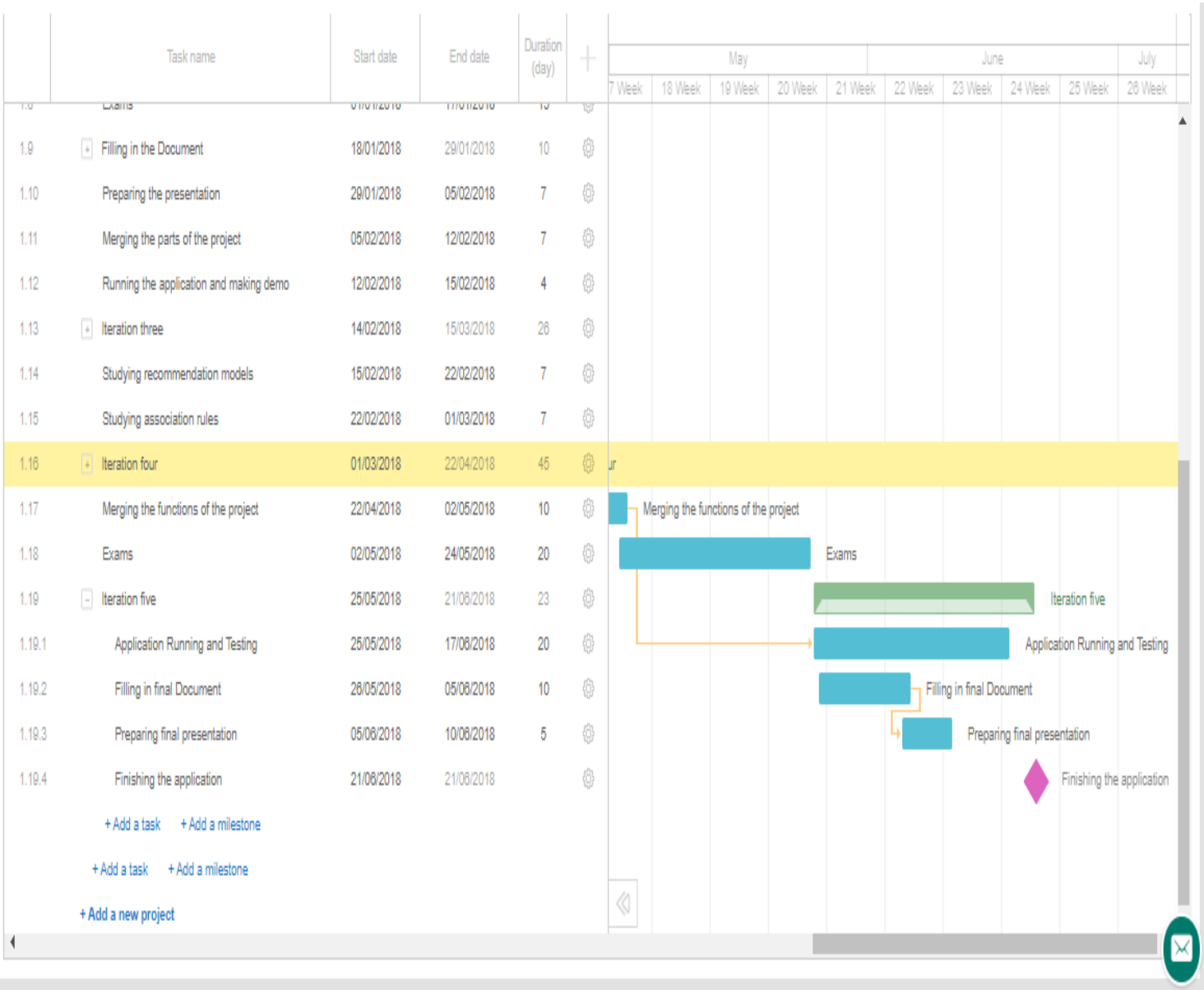


	Task name	Start date	End date	Duration (day)
1.8	Exams	01/01/2018	17/01/2018	15
1.9	Filling in the Document	18/01/2018	29/01/2018	10
1.9.1	Chapter one	18/01/2018	29/01/2018	10
1.9.2	Chapter two	18/01/2018	29/01/2018	10
1.9.3	Chapter three	18/01/2018	29/01/2018	10
1.9.4	Finishing Document	29/01/2018	29/01/2018	
	+ Add a task + Add a milestone			
1.10	Preparing the presentation	29/01/2018	05/02/2018	7
1.11	Merging the parts of the project	05/02/2018	12/02/2018	7
1.12	Running the application and making demo	12/02/2018	15/02/2018	4
1.13	Iteration three	14/02/2018	15/03/2018	26
1.14	Studying recommendation models	15/02/2018	22/02/2018	7
1.15	Studying association rules	22/02/2018	01/03/2018	7
1.16	Iteration four	01/03/2018	22/04/2018	45
1.17	Merging the functions of the project	22/04/2018	02/05/2018	10
1.18	Exams	02/05/2018	24/05/2018	20
1.19	Iteration five	25/05/2018	21/06/2018	23
1.19.1	Application Running and Testing	25/05/2018	17/06/2018	20



	Task name	Start date	End date	Duration (day)
1.8	Exams	01/01/2018	17/01/2018	15
1.9	Filling in the Document	18/01/2018	29/01/2018	10
1.10	Preparing the presentation	29/01/2018	05/02/2018	7
1.11	Merging the parts of the project	05/02/2018	12/02/2018	7
1.12	Running the application and making demo	12/02/2018	15/02/2018	4
1.13	Iteration three	14/02/2018	15/03/2018	26
1.13.1	Developing	14/02/2018	15/03/2018	26
1.13.1.1	The menu	14/02/2018	25/02/2018	10
1.13.1.2	Build database	17/02/2018	27/02/2018	10
1.13.1.3	Make the connection with the DB	27/02/2018	15/03/2018	15
	+ Add a task + Add a milestone			
	+ Add a task + Add a milestone			
1.14	Studying recommendation models	15/02/2018	22/02/2018	7
1.15	Studying association rules	22/02/2018	01/03/2018	7
1.16	Iteration four	01/03/2018	22/04/2018	45
1.17	Merging the functions of the project	22/04/2018	02/05/2018	10
1.18	Exams	02/05/2018	24/05/2018	20
1.19	Iteration five	25/05/2018	21/06/2018	23


Task name
Start date
End date
Duration (day)
2018
March
April
07 Week
08 Week
09 Week
10 Week
11 Week
12 Week
13 Week
14 Week
15 Week
16 Week
1.13 Iteration three 14/02/2018 15/03/2018 26
1.14 Studying recommendation models 15/02/2018 22/02/2018 7
1.15 Studying association rules 22/02/2018 01/03/2018 7
1.16 Iteration four 01/03/2018 22/04/2018 45
1.16.1 Analysis 01/03/2018 08/03/2018 7
1.16.1.1 Physical class diagram 01/03/2018 08/03/2018 7
+ Add a task + Add a milestone
1.16.2 Developing 08/03/2018 22/04/2018 39
1.16.2.1 Applying association rules 08/03/2018 31/03/2018 20
1.16.2.2 Developing checklist functions 31/03/2018 22/04/2018 20
1.16.2.3 Developing payment functions 31/03/2018 16/04/2018 15
1.16.2.4 Finishing association rules 31/03/2018 31/03/2018
+ Add a task + Add a milestone
1.16.3 Iteration four finished 22/04/2018 22/04/2018
+ Add a task + Add a milestone
1.17 Merging the functions of the project 22/04/2018 02/05/2018 10
1.18 Exams 02/05/2018 24/05/2018 20
1.19 Iteration five 25/05/2018 21/06/2018 23
Iteration three
Studying recommendation models
Studying association rules
Analysis
Physical class diagram
Applying association rules
Finishing association rules

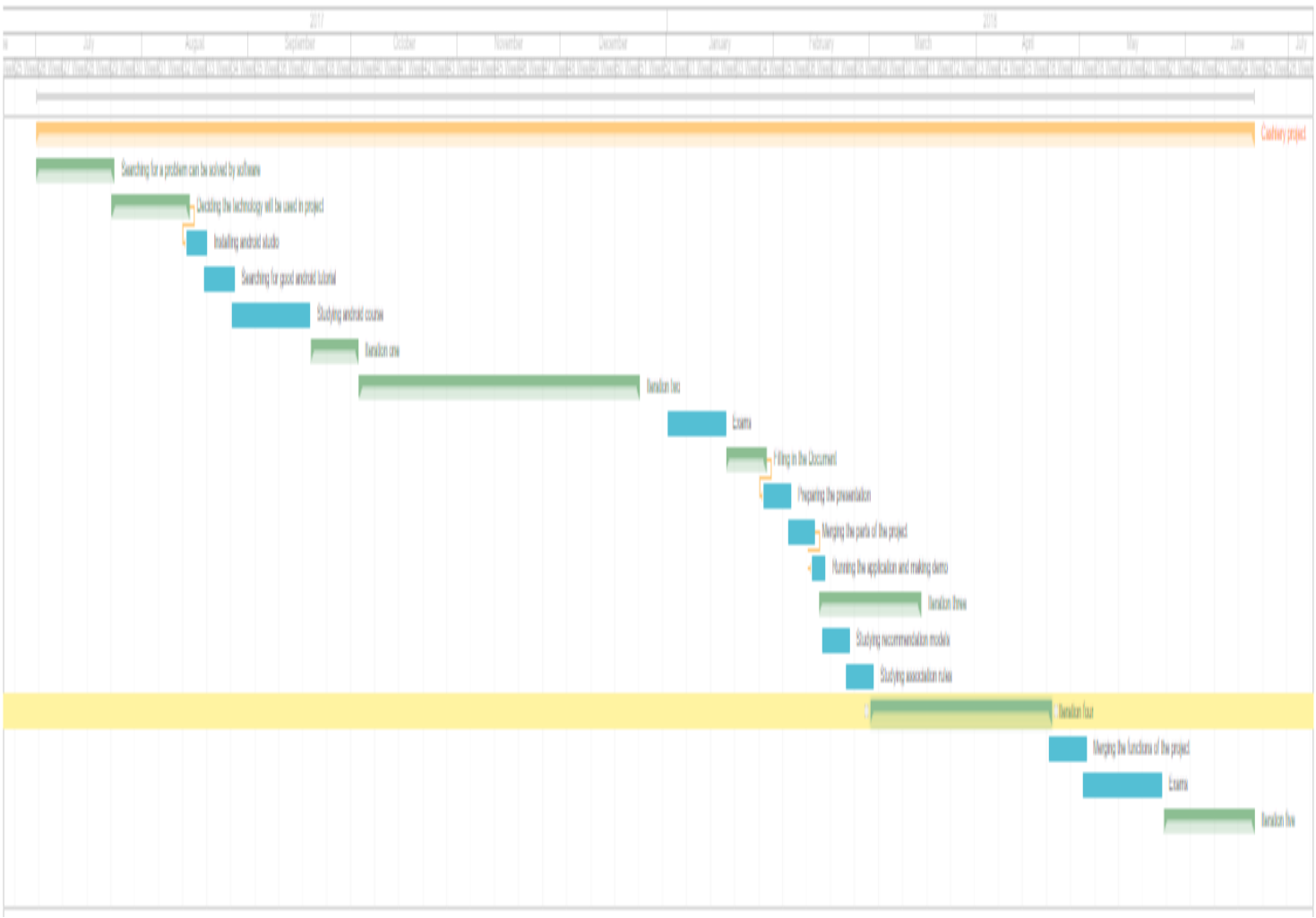


	Task name	Start date	End date	Duration (day)
1.9	Filling in the Document	18/01/2018	29/01/2018	10
1.10	Preparing the presentation	29/01/2018	05/02/2018	7
1.11	Merging the parts of the project	05/02/2018	12/02/2018	7
1.12	Running the application and making demo	12/02/2018	15/02/2018	4
1.13	Iteration three	14/02/2018	15/03/2018	26
1.14	Studying recommendation models	15/02/2018	22/02/2018	7
1.15	Studying association rules	22/02/2018	01/03/2018	7
1.16	Iteration four	01/03/2018	22/04/2018	45
1.17	Merging the functions of the project	22/04/2018	02/05/2018	10
1.18	Exams	02/05/2018	24/05/2018	20
1.19	Iteration five	25/05/2018	21/06/2018	23
1.19.1	Application Running and Testing	25/05/2018	17/06/2018	20
1.19.2	Filling in final Document	26/05/2018	05/06/2018	10
1.19.3	Preparing final presentation	05/06/2018	10/06/2018	5
1.19.4	Finishing the application	21/06/2018	21/06/2018	

+ Add a task + Add a milestone

+ Add a task + Add a milestone

+ Add a new project


2017
2018
July
August
September
October
November
December
January
February
March
April
May
June
July
Cashiery project
Searching for a problem can be solved by software
Checking the technology will be used in project
Installing android studio
Searching for good android tutorial
Studying android course
Iteration one
Iteration two
Exams
Filling in the Document
Preparing the presentation
Merging the parts of the project
Running the application and making demo
Iteration three
Studying recommendation models
Studying association rules
Iteration four
Merging the functions of the project
Exams
Iteration five

1-11 Current state

We now in iteration Four, this iteration mainly focuses on running the application, testing and evaluation. We now try test cases, fix bugs and make improvements in the backend code, UI and model accuracy.

1-12 Conclusion

Our application creates something like online shopping cart on the android using scanning within specific markets and enable paying too with different methods, you can know your total amount of your receipt before paying, you can cancel some products, you can see the recommendations based on your interests and your shopping history, you can see the related items you can buy, you can check offers of the market.

2-System Analysis

2-1 Project stakeholders

Market	- The markets which use Cashiery application have big impact on the application, as the application main goal is to facilitate markets payment process And use the offline data of purchasing transactions done in these markets.
Market Customers'	- The market customers are the customers of our application , they also have main influence on the application as the main goal of the application is to collect the purchasing data the customers done and provide to the self –scan to their products and online payments.
Clerk	- The clerk has impact on the application as he has its own scenario to validate the purchasing process.

Online Shopping Cart Applications	- Online Shopping Cart Applications, they are our competitors they will influenced by our application, as we provide simple design, self-scan to the products easily, and provide recommendations to the customer before and after scanning the item he desired to buy which facilitate to him shopping process.

2-2 Functional Requirements

Function	Description
Sign up	The customer register the application for the first time
Login	The customer already has accounts and enter their username and password to enter the application
Choose Market	The customer choose the desired market
See recommendations	The application give some recommendations to the customer according to his history ex: customer x always buy chocolate so the system will introduce to him the chocolate offers this market present
Scan	The application will open the camera and enable the user to scan products bar code

Show products	The application views the scanned products in a list which contains each product information(price, quantity)
Remove products	The user can remove a scanned product from the list
Generate association rules	The application recommend to the customer other products which may be related to the products he scanned ex: customer x scanned a pencil so the application will recommend to him buying Pencil sharpener
Pay	The customer chooses the payment option whether he enters his visa card info to pay online(pay online) or to pay in the market(pay on cashier)
See offers	The application will list all the offers represented by the markets
Logout	The user can sign out his account from the application
Validate user information	This function validate user information while signing in to check if the user name and password correct
Enter card information	This function takes card information of a customer
Validation of card information	This function makes the sequence of customers' card validation(through the bank)
Show receipt number	This function generate a unique receipt number for each customer to pay and take the

	receipt paper from clerk
Validate products	This function validates the products in the receipt's customer for security reasons.
Choose branch	The customer should choose the branch of the market if the market have.
Generate receipt number	The clerk generate receipt after validating products

2-3 Non-Functional Requirements

Function	Description
Reliability	- Save data of customers on online server. - safe by 90%
Usability	- The application is usable because it is simple. - Any complex function has instructions. - The application have cancel function to remove any scanned products before confirm order.
Performance	- During scan, the application catches the bar code in milliseconds. - The application takes milliseconds to move between pages. - The application give customers recommendations.
Robustness /scalability	- Our application can cope with user erroneous inputs as there are input validations - Can also cope with errors during execution

Availability	- The application available 24/7
Efficiency	- The application is light on the phone battery. - Network connectivity is automatic as well as the Wi-Fi is open and asked for connection when mobile data is opened.
Portability	- The application is portable for any android system not less than ice-cream sandwich version.
Appearance	- Our application has UI comfortable to eye. - Easy to be understand. - Our application follow the UX principles .
Usefulness	- Our application is useful for markets and their customers.
Testability	- Our application follows SOLID principles testing which are applied to. the system functionality.
Security	- Each user has his own account to make his visa card info secure.
Privacy	- The application asked the user only for his user name and password to login and forward asked for visa info. - All these info are encrypted and no one can use it without the user confirmation.

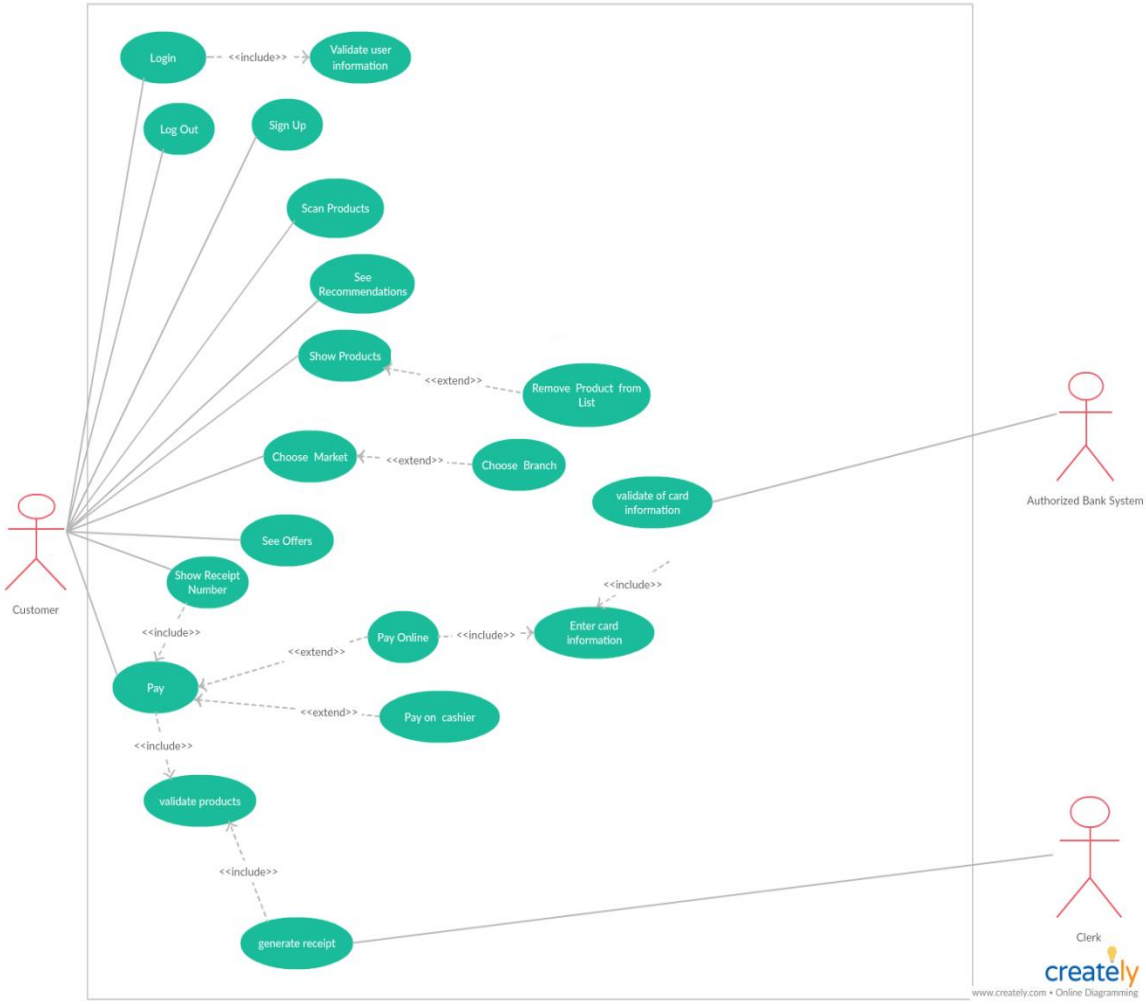
2-4 Simple Scenario

First, the customer enters his username and password, then choose his desired market and the branch, the application will display a page to indicate the beginning of the scan and include recommendation option based on customer history, then the customer will begin to scan the bar code of his products, a list with all the scanned products with their prices and the total cost so far will appear, the customer has the ability to remove any of the scanned products, then associated products(the mining results) appeared, then he will proceed to pay whether with visa info to pay online or in market, Finally a receipt number will be generated to the customer.

When a customer arrived to make his receipt verified the clerk will choose the receipt number, then the clerk will scan the customer products', if all the products validated by the clerk are already the products found in the generated receipt then this receipt will be checked as verified and so on.

3- System Design

3-1 Use Case Model



3-2 Use Case tables

<table>
<tr><td>Use Case ID:</td><td colspan="2">one</td></tr>
<tr><td>Use Case Name:</td><td colspan="2">Login</td></tr>
<tr><td>Actors:</td><td colspan="2">Customer</td></tr>
<tr><td>Pre-conditions:</td><td colspan="2">should have an account on the Application(create account)</td></tr>
<tr><td>Post-conditions:</td><td colspan="2">User can perform any transaction (Scan products, Show offer, recommendations and perform any feature in the Application) on the Application.</td></tr>
<tr><td rowspan="5">Flow of events:</td><td>User Action</td><td>System Action</td></tr>
<tr><td>1- User enters username and password.</td><td></td></tr>
<tr><td>2-User click on button login in The login page of the Application.</td><td></td></tr>
<tr><td></td><td>3-System validates the login information(username, password) against account table in data base</td></tr>
<tr><td></td><td>4-Display the personal page for the user</td></tr>
<tr><td rowspan="3">Exceptions:</td><td>User Action</td><td>System Action</td></tr>
<tr><td>1- User enter invalid username, password</td><td></td></tr>
<tr><td></td><td>2- system display invalid message then request from user to re-enter the information again</td></tr>
<tr><td>Includes:</td><td>Seven</td><td></td></tr>
</table>

Use Case ID:	Two

Use Case Name:	Sign up	
Actors:	Customer	
Pre-conditions:	Display Login page	
Post-conditions:	User can perform any transaction (Scan products, Show offer, recommendations and perform any feature in the Application) on the Application.	
Flow of events:	**User Action**	**System Action**
	1- User enter name, password, Name, Age, Email(his data).	
	2-User Click on Register Button	
		3-system save Data on Data Base
		4-System display Customer Information after saving it in Customer's Information page
		5-Display Done message
Exceptions:	**User Action**	**System Action**
	1- User Enter invalid username , password or email	
		2- system display invalid message

Use Case ID:	Three

Use Case Name:	Choose Market and choose branch	
Actors:	Customer	
Pre-conditions:	Login	
Post-conditions:	Choose branch if any, Scan any product of the market using mobile Camera, see recommendations	
Flow of events:	**User Action**	**System Action**
	1- User login from login page	
		2- System Verify user data
		3-System Display Markets that support our Application and its branch if they have
	4-Customer Choose Market and then choose market branch if market have	
		5-Check customer location if Customer on the market place that choose it or not.
	User Action	**System Action**
Exceptions:	1-if user enters invalid information in form of login.	2-The system displays wrong message then request enter the information again
	3- if user select specific market and system detect its wrong place	4-System Display error message and request to select the right place.
Extends	Choose market extends choose branch	

Use Case ID:	Four

Use Case Name:	Scan products	
Actors:	Customer	
Pre-conditions:	Choose Market and its branch.	
Post-conditions:	Show product list	
Flow of events:	**User Action**	**System Action**
	1- User enter name , password, email	
		2- System Verify user data.
		3-System Display Markets that support our Application
		4-System Display Markets branches
	5-Customer Choose Market and then choose market branch if market have	
		6- System display page that include recommendations and scan button.
	7-customer can see recommendations or scan after clicking on scan button.	
		8-system open mobile camera
	9-customer can scan bar code of product easily using mobile camera.	
Exceptions:	**User Action**	**System Action**
	1- User Enter invalid username , password or email	
		2- system display invalid message
	3- user choose wrong market	4-if customer scan product that not

		included in market DB product table, system display error message as user not in right place.

Use Case ID:	five	
Use Case Name:	Show product	
Actors:	customer	
Pre-conditions:	Choose Market Choose market branch Scan products	
Post-conditions:	remove product from list , show associated products	
Flow of events:	**User Action**	**System Action**
	1- User enter name , password, email	
		2- System Verify user data.
		3-System Display Markets that support our Application
		4-System Display Markets branches
	5-Customer Choose Market and then choose market branch if market have	
		6- System display recommendation button to customer based on his history and scan button.
	7- Customer can scan after choosing scan button.	
		8- system open mobile camera

	9- customer can scan bar code of products	
	10- User chooses finish after scanning all products he needs.	
		11-then system will display All product that customer scanned will specific description and price for each one and he can cancel some products.
Includes	Nine	
Extends	Six	

Use Case ID:	Six	
Use Case Name:	Remove product from list	
Actors:	Customer	
Pre-conditions:	Product should be scanned first	
Post-conditions:	Able to pay and show recommendation	
Flow of events:	**User Action**	**System Action**
	1-Customer choose scan button	
		2-system open mobile camera
	3-customer can scan bar code of products.	
	4- customer clicks next in order to display product list	

		5- System will display All product that customer scanned with name and price for each.
	6-customer can remove product from list by clicked on “X” button	
		7- System will removed product from product list.

Use Case ID:	seven	
Use Case Name:	Validate user information	
Actors:	System	
Pre-conditions:	Login	
Post-conditions:	Choose market and branch	
Flow of events:	**User Action**	**System Action**
	1- User enter the username and password	
		2-system will takes information to validate it from database to make sure that this data form customer who has account already in the application
		3-system will move to welcome page when information is valid
Exceptions:	**User Action**	**System Action**
	1- User Enter invalid username , password	

		2- system display invalid message

Use Case ID:	Eight	
Use Case Name:	See recommendations	
Actors:	Customer	
Pre-conditions:	Login and it's information is valid	
Post-conditions:	Choose markets and branch	
Flow of events:	**User Action**	**System Action**
	1-user enter user name and password	
		2-system validate information from customer database
		3-system will display the first page which contains recommendation button based on its history only.

Use Case ID:	Nine	
Use Case Name:	Generate association rules	
Actors:	System	
Pre-conditions:	Scan	
Post-conditions:	Show products and scan again (if wanted) then pay using online payment or in market	
	User Action	**System Action**
	1-user scan wanted products	

Flow of events:		2-system will show associated products based on different customers data
	3-user close recommendation window	
		4-system remove recommendations screen and show the scanned product list

Use Case ID:	Ten	
Use Case Name:	See offers	
Actors:	Customer	
Pre-conditions:	Login, Choose market and branch	
Post-conditions:	Scan, see recommendations	
Flow of events:	**User Action**	**System Action**
	1-user choose market	
		2-system will retrieve branches from database to selected market.
	3-user choose branch	
		4-system will retrieve offers from database for selected branch

Use Case ID:	Eleven
Use Case Name:	Validate products and generate receipt
Actors:	Clerk
Pre-conditions:	Have an account on application (done by system)

Post-conditions:	Scan purchased products	
Flow of events:	**User Action**	**System Action**
		1-system display the welcome page for clerk
	2-clerk will enter receipt number	
		3-system will retrieve from data base all products in this receipt based on receipt's number
	6-Clerk will scan product in basket's customer.	
		7-system will save scanned product in the screen and compare scanned product with product in receipt.
		8- if All product in receipt are same as scanned product the system sent verified message as a notification on clerk
		9-sytem will be generated receipt
	10- clerk can be able to print receipt to give it to customer	
Exceptions:	**User Action**	**System Action**
	1-if there is a product in customer's basket not scanned by him and clerk detects it.	
		2-system cannot generate receipt and it will display error message to clerk to notify him that there a product in customer basket did not matched with receipt's products.

Use Case ID:	Twelve	
Use Case Name:	Logout	
Actors:	Customer	
Pre-conditions:	Have an account on our Application	
Post-conditions:		
Flow of events:	**User Action**	**System Action**
	1-User Click on Menu.	
		2-system will Display Menu's option
	3-User Choose Log Out	
		4-System will kill the User's Session.
		5-System will return user to login page.

Use Case ID:	thirteen	
Use Case Name:	Show receipt number	
Actors:	System	
Pre-conditions:	The customer should be pay	
Post-conditions:	Validation products by clerk	
Flow of events:	**User Action**	**System Action**
	1- customer should choose payment method	
		2-system will transfer the money to hypermarket account
		3-system will show receipt number

CASHIERY
Just scan it

Use Case ID:	Fourteen	
Use Case Name:	Enter Card Information , pay online	
Actors:	Customer	
Pre-conditions:	Should be Scanned products first.	
Post-conditions:	System can generate its receipt and customer able to see receipt number.	
Flow of events:	**User Action**	**System Action**
	1- user clicked on pay online Button	
		2-System request from user to enter card Information
	3-user entered valid card information.	
		4-system take user's card information and sent it to authorized bank to be verified .
Exceptions:	**User Action**	**System Action**
	1- User enter invalid card information	
		2- system display invalid message then request from user to re-enter the information again.
Includes:	Sixteen	

Use Case ID:	Fifteen
Use Case Name:	pay , pay on cashier
Actors:	Customer
Pre-conditions:	Should be Scanned products first.

Post-conditions:	Able to see receipt number and system can generate its receipt.	
Flow of events:	**User Action**	**System Action**
	1- User clicked on pay on cashier Button.	
		2-sytem will be generate a receipt number to pay on cashier and able to take its receipt.
Includes:	Thirteen , Eleven	

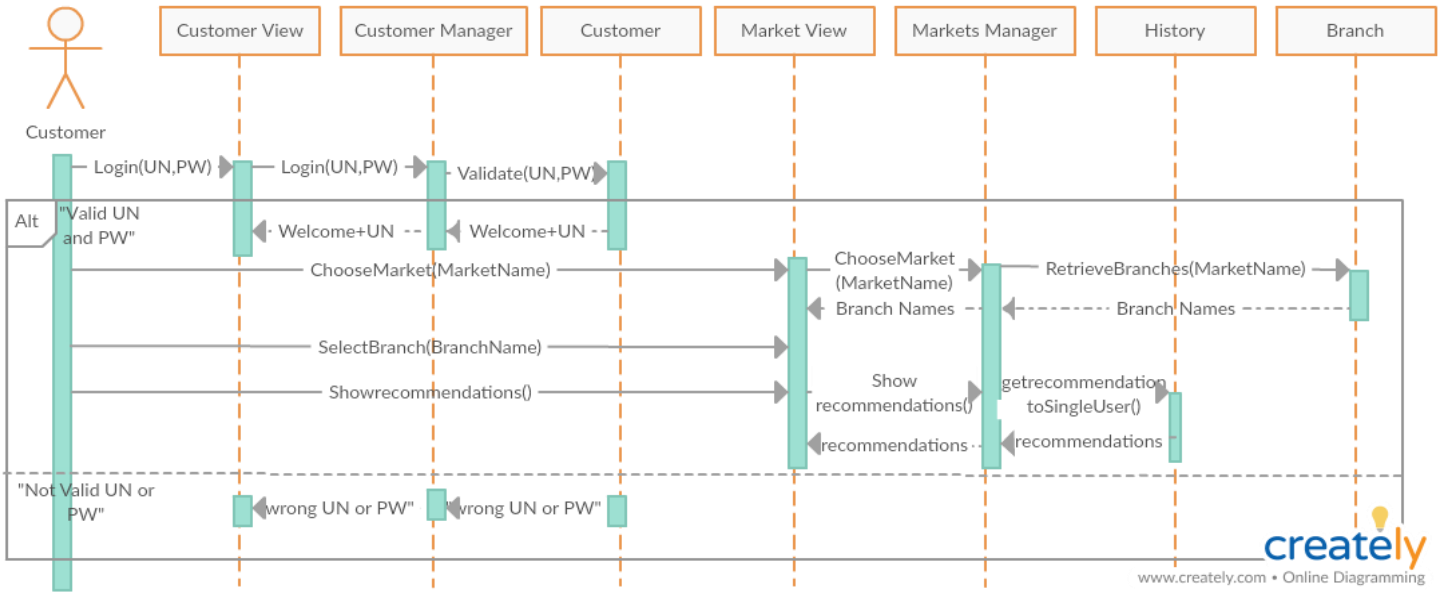
Use Case ID:	Sixteen "black-box Use case".	
Use Case Name:	Validate Card Information.	
Actors:	Authorized Bank system .	
Pre-conditions:	Products should be scanned by user.	
Post-conditions:	Able to see receipt number and system can generate its receipt.	
Flow of events:	**User Action**	**System Action**
	1- user clicked on pay online button	
		2-Sytem requested to enter card information
	3-user entered card information	
		5-System will be response a verified message from bank system.
Exceptions:	**User Action**	**System Action**
	1- User enter invalid Card	

	Information	
		2-Authorized bank system will be sending an invalid message to the system.
		3-system will send to user to re-enter his information again.

3-3 Sequence Diagram

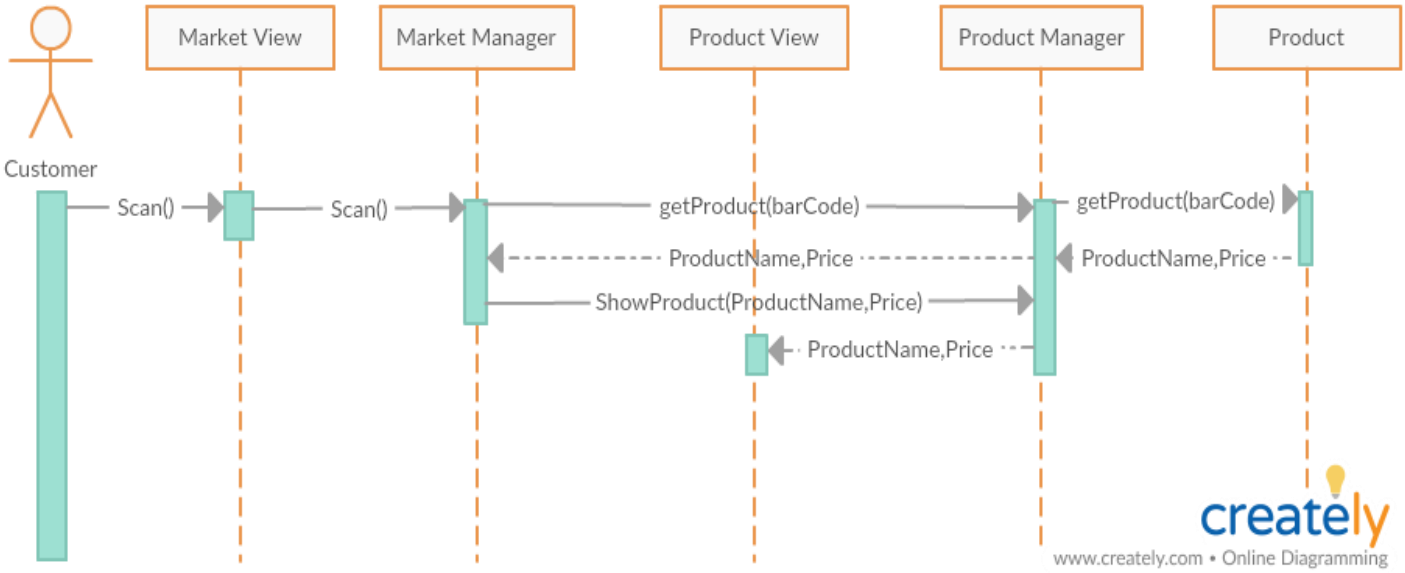
Note that 'UN'=user name, 'PW'=password.

Show recommendations:



Scan products:

(Login steps skipped as mentioned in the previous sequence diagram)

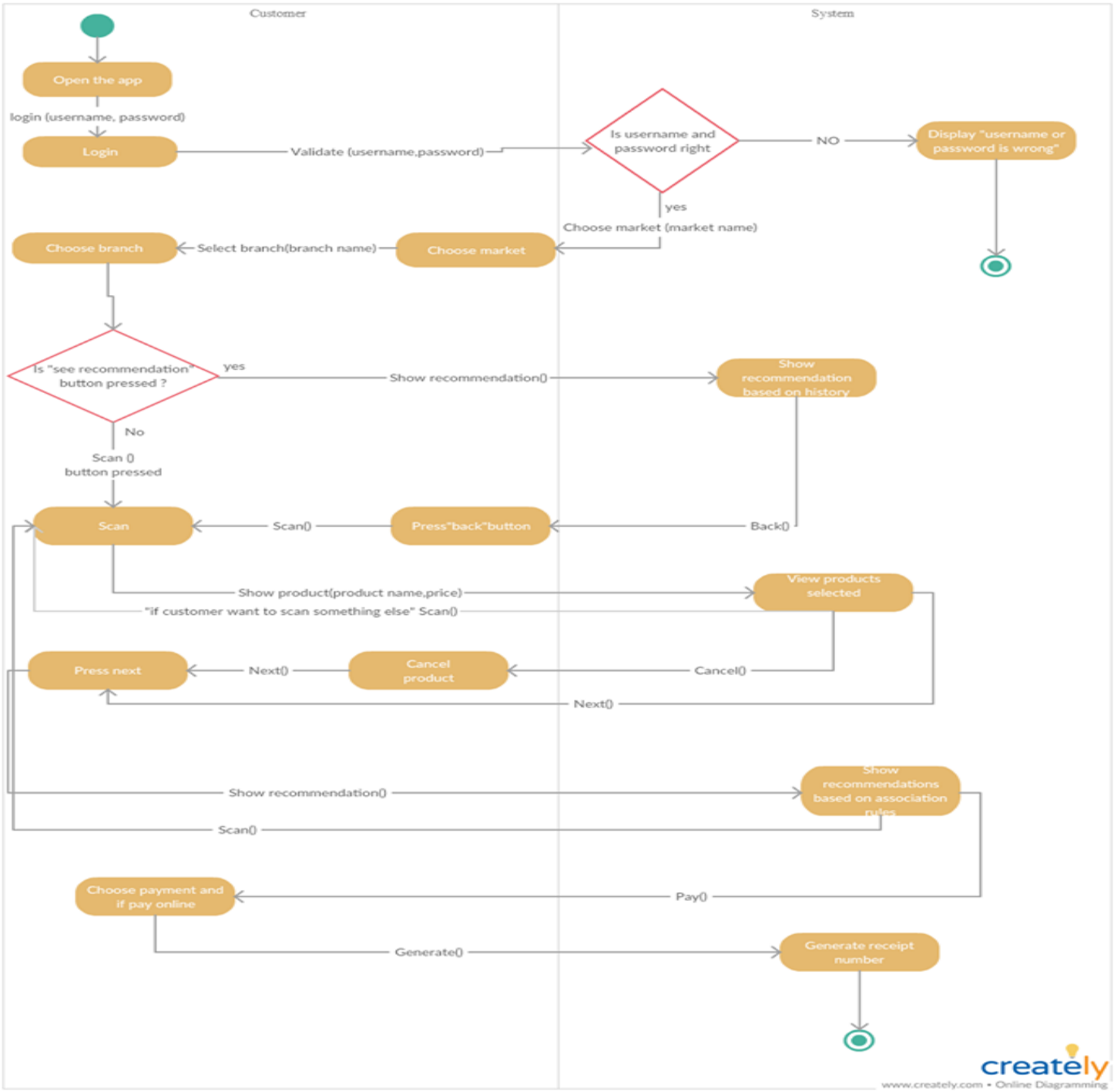


3-4 Activity Diagram

Generate receipt number:

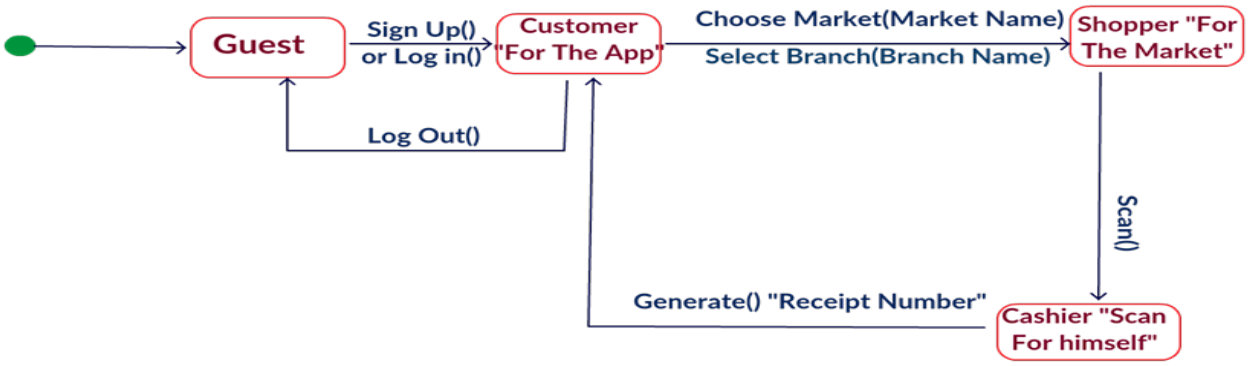
Assume sign up done before.

Assume 'pay' black box so no details for it here.

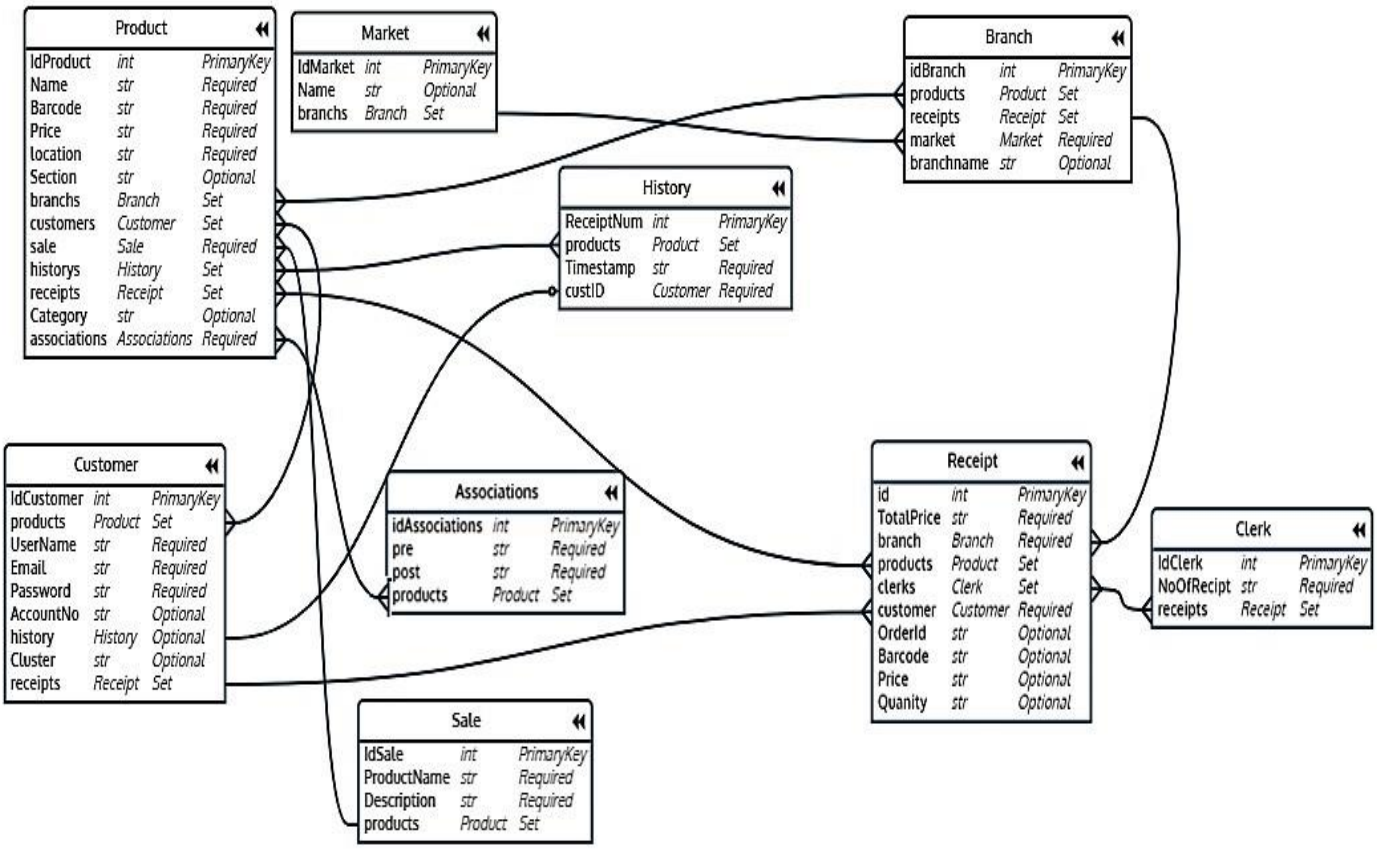

Customer
System
Open the app
login (username, password)
Login
Validate (username,password)
Is username and password right
NO
Display "username or password is wrong"
yes
Choose market (market name)
Choose market
Select branch(branch name)
Choose branch
Is "see recommendation" button pressed ?
yes
Show recommendation()
Show recommendation based on history
No
Scan () button pressed
Scan
Scan()
Press"back"button
Back()
Show product(product name,price)
View products selected
"if customer want to scan something else" Scan()
Press next
Next()
Cancel product
Cancel()
Next()
Show recommendation()
Show recommendations based on association rules
Scan()
Choose payment and if pay online
Pay()
Generate()
Generate receipt number
creately
www.creately.com • Online Diagramming

3-5 State Diagram

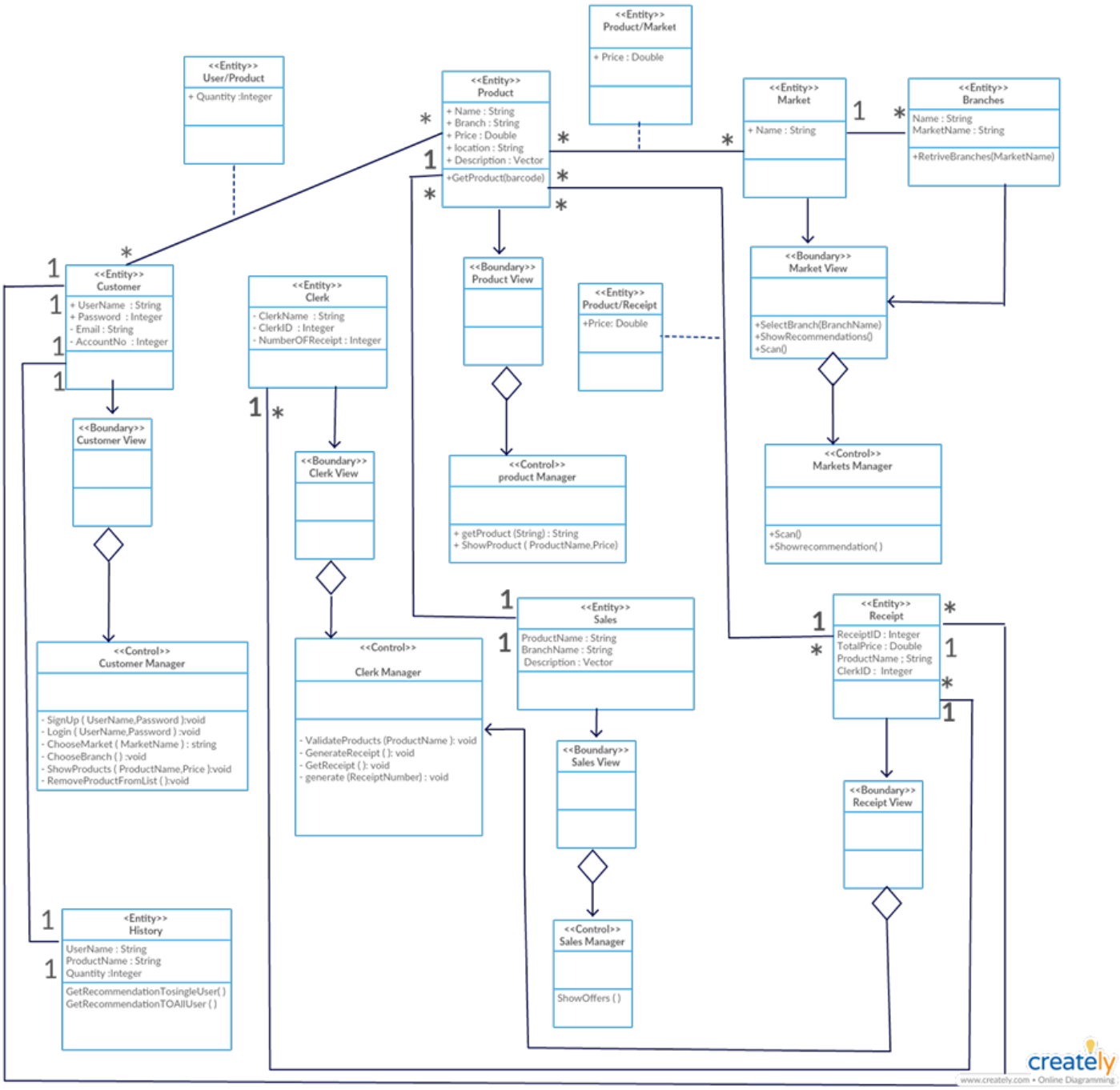
Customer State Diagram:



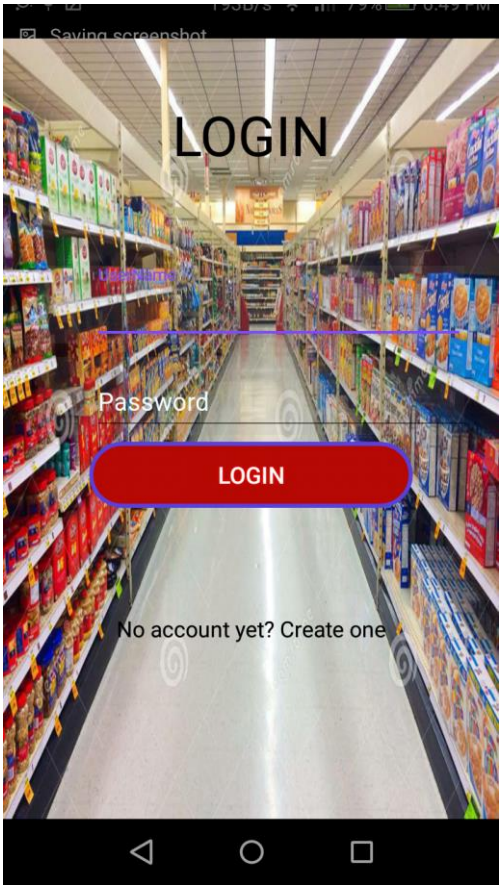
3-6 ERD

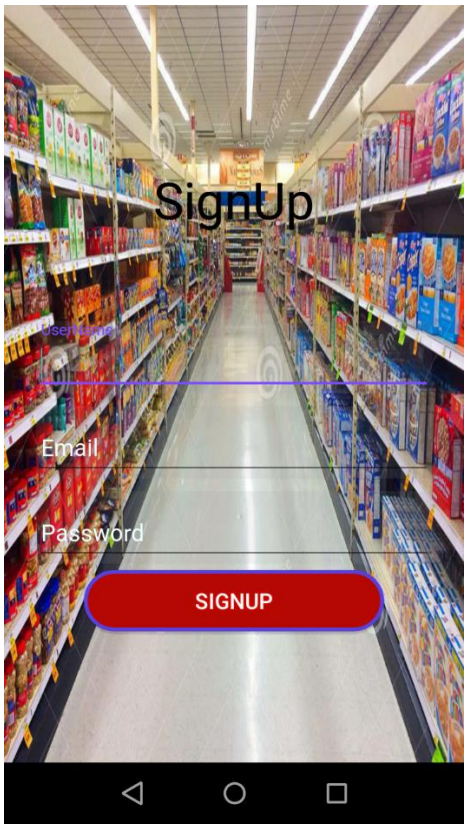

Product
IdProduct int PrimaryKey
Name str Required
Barcode str Required
Price str Required
location str Required
Section str Optional
branchs Branch Set
customers Customer Set
sale Sale Required
historys History Set
receipts Receipt Set
Category str Optional
associations Associations Required
Market
IdMarket int PrimaryKey
Name str Optional
branchs Branch Set
Branch
idBranch int PrimaryKey
products Product Set
receipts Receipt Set
market Market Required
branchname str Optional
History
ReceiptNum int PrimaryKey
products Product Set
Timestamp str Required
custID Customer Required
Customer
IdCustomer int PrimaryKey
products Product Set
UserName str Required
Email str Required
Password str Required
AccountNo str Optional
history History Optional
Cluster str Optional
receipts Receipt Set
Associations
idAssociations int PrimaryKey
pre str Required
post str Required
products Product Set
Receipt
id int PrimaryKey
TotalPrice str Required
branch Branch Required
products Product Set
clerks Clerk Set
customer Customer Required
OrderId str Optional
Barcode str Optional
Price str Optional
Quanity str Optional
Clerk
IdClerk int PrimaryKey
NoOfRecipt str Required
receipts Receipt Set
Sale
IdSale int PrimaryKey
ProductName str Required
Description str Required
products Product Set

3-7 Class diagram



3-8 Prototype




SignUp
Email
Password
SIGNUP


299B/s 79% 6:50 PM
Cashiery
Carrefour
SINCE 1970
METRO

CASHIERY

Just scan it

Carrefour

Your Favorite.....

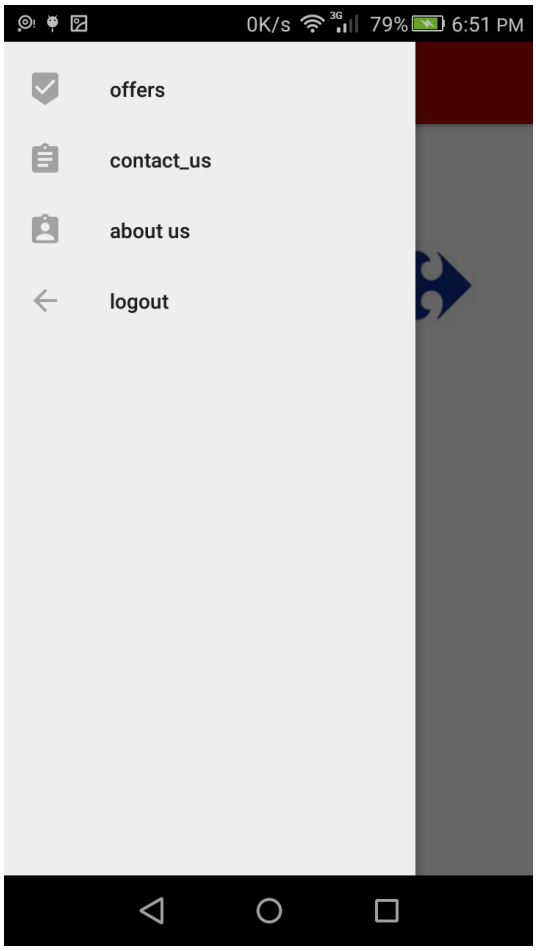
orange joice -->12 packets with 1 litter with price 258 LE (sale 40%)
Robust Golden Unsweetened Oolong Tea-->buy one and take one free

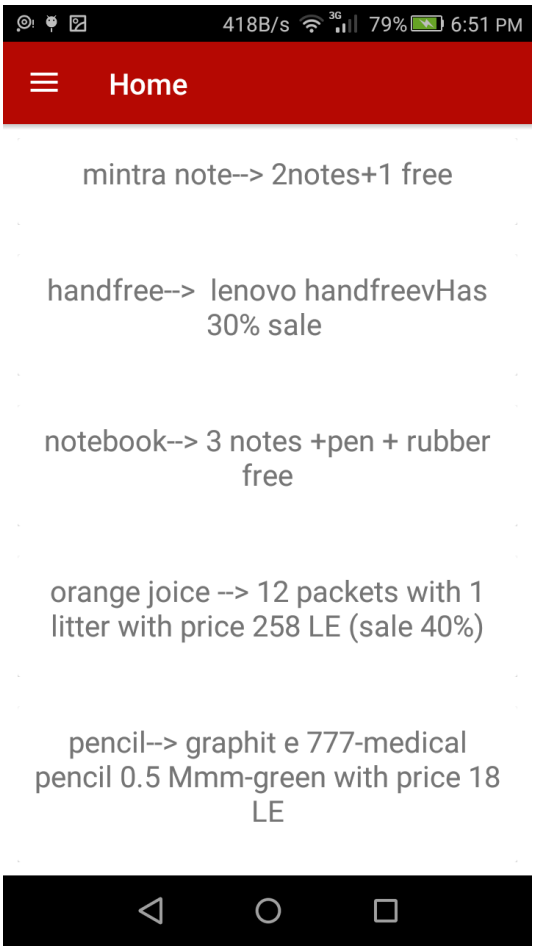
SKIP

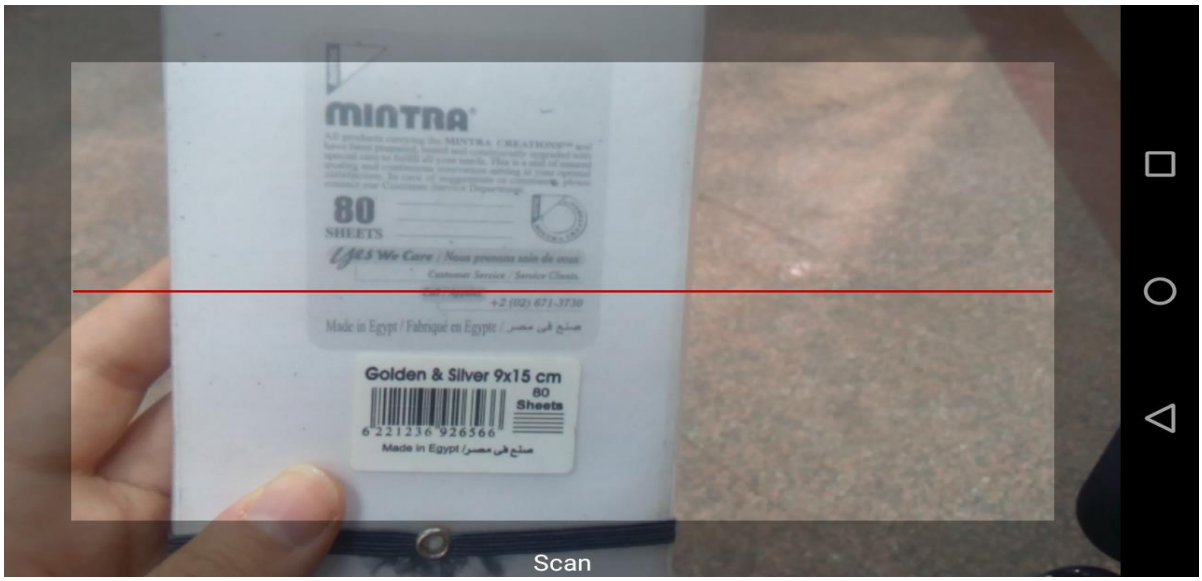
Carrefour

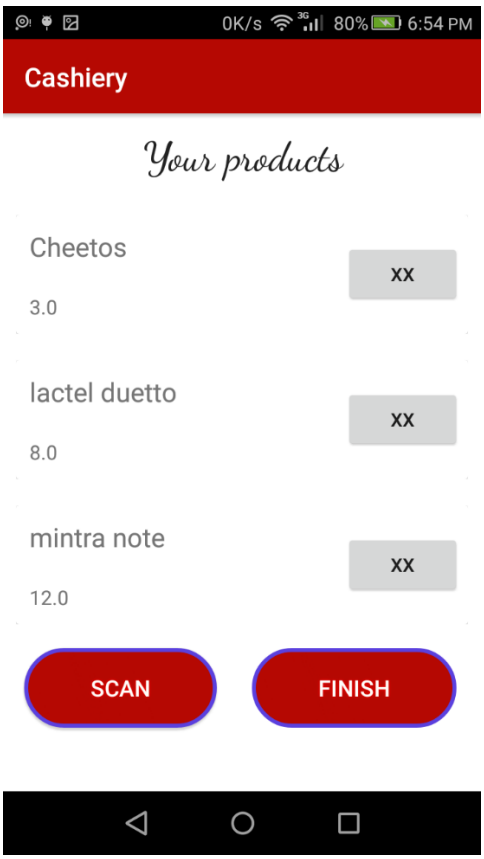
Welcome

GO TO SHOPPING


0K/s 79% 6:51 PM
offers
contact_us
about us
logout

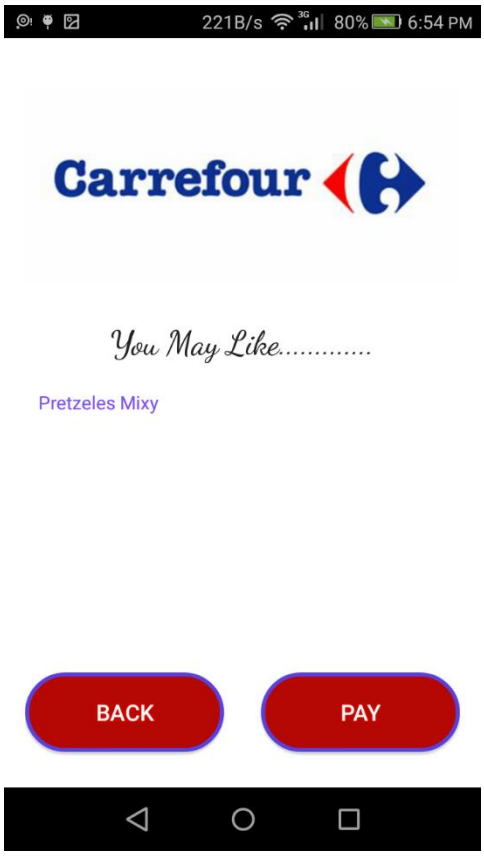

418B/s 79% 6:51 PM
Home
mintra note--> 2notes+1 free
handfree--> lenovo handfreevHas 30% sale
notebook--> 3 notes +pen + rubber free
orange joice --> 12 packets with 1 litter with price 258 LE (sale 40%)
pencil--> graphit e 777-medical pencil 0.5 Mmm-green with price 18 LE

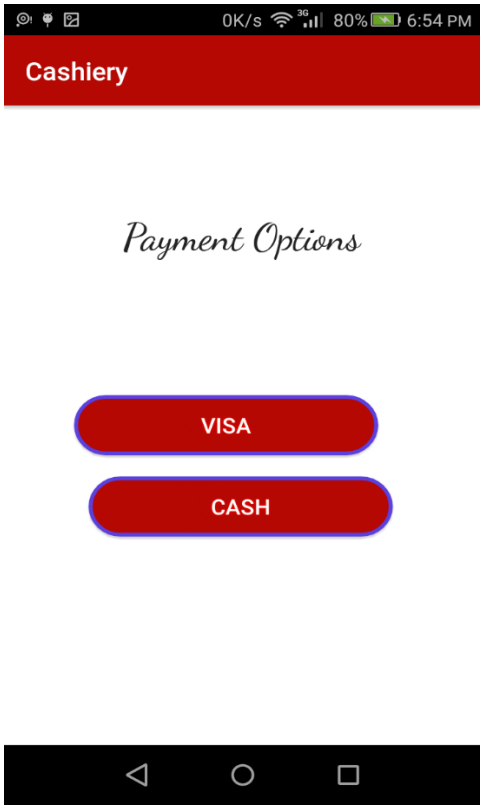

MINTRA
80
SHEETS
Golden & Silver 9x15 cm
80
Sheets
Made in Egypt
Scan

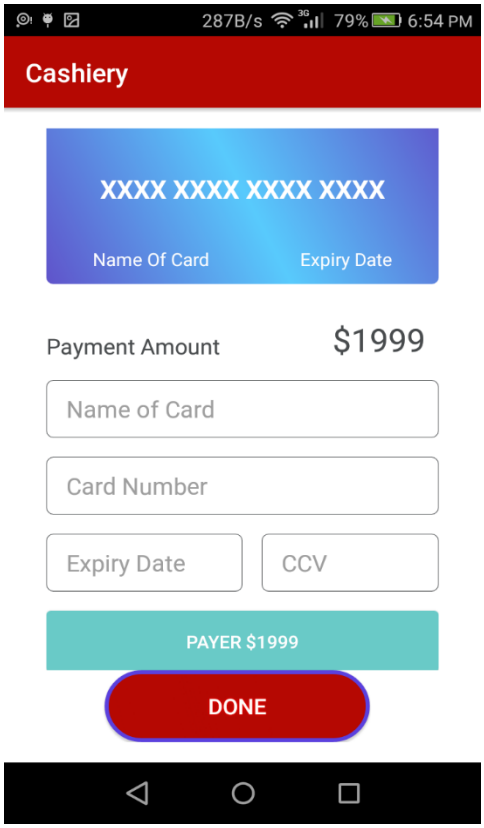

0K/s 80% 6:54 PM
Cashiery
Your products
Cheetos
3.0
XX
lactel duetto
8.0
XX
mintra note
12.0
XX
SCAN
FINISH


CASHIERY
Just scan it


0K/s 80% 6:54 PM
Cashiery
Your products
Cheetos
3.0
xx
lactel duetto
8.0
xx
SCAN
FINISH


221B/s 80% 6:54 PM
Carrefour
You May Like.....
Pretzeles Mixy
BACK
PAY


0K/s 80% 6:54 PM
Cashiery
Payment Options
VISA
CASH


287B/s 79% 6:54 PM
Cashiery
XXXX XXXX XXXX XXXX
Name Of Card
Expiry Date
Payment Amount
$1999
Name of Card
Card Number
Expiry Date
CCV
PAYER $1999
DONE


Cashiery
Recicipt Number
225

4-Implementation

4-1 Cashiery Main technique

Our Application implementation is following MVC (Model-View-Controller) Architecture, the model is represented in our database which is found locally on MySQL, the view is represented in the android view where we call the services found in the controller, and the controller is represented in the backend code which is written in java and using spring boot as a framework.

We built the functions of our application in java like Login, Signup, and the recommendation system which is represented in Apriori algorithm and clustering but using Spring boot framework, this framework convert the functions into services which we can call these services in the android application, in the android application we built the user interface and call the services. We will put the spot on the main functions in our application in the following pages.

4-1-1 The Backend Code

The login function written in java, this function take parameters the username and the password, then check if this data is found in the data base or not, if not found it will display cannot login message.

```
//////////////////////////////// login//////////////////////////////////////////
public customer login(String un, String pw) throws ClassNotFoundException, SQLException {
    Class.forName("com.mysql.jdbc.Driver");
    java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_gp", "root", "ro
    java.sql.Statement stmt = con.createStatement();
    ResultSet custWithpw = stmt.executeQuery("select username,password,idcustomer from customer where user
            + un + "' and password= '" + pw + "'");
    customer c = new customer();
    if (custWithpw.next()) {
        c.customerid = custWithpw.getString("idcustomer");
        c.username = custWithpw.getString("username");
        c.mssg = "success";
        return c;// obj contain id
    } else {
        c.mssg = "Fail";
        return c;

    }
}
```

CASHIERY
Just scan it

Then make it as a service,

```
////////////////////// login service/////////////////
@RequestMapping(value = "/login", method = RequestMethod.POST)
public customer Login(@RequestParam("uname") String username, @RequestParam("pw") String password)
        throws ClassNotFoundException, SQLException {
    MainModel mm = new MainModel();
    customer dd = new customer();

    dd = mm.login(username, password);

    return dd;
}
```

The signup function written in java, this function take the parameters username, password and email, then it add the new user data in the database.

```
//////////////////////// sign up//////////////////////////////////
public customer SignUp(String un, String email, String pw) throws ClassNotFoundException, SQLException {
    Class.forName("com.mysql.jdbc.Driver");
    java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_gp", "root", "rooot");
    java.sql.Statement stmt = con.createStatement();
    customer c = new customer();

    if (email.contains("/") || email.contains("\\") || email.contains("|") || !email.contains("@")
            || email.contains("$") || email.contains("#") || email.contains("%") || email.contains("^")
            || email.contains("&") || email.contains("*") || email.contains("(") || email.contains(")")
            || email.contains("+") || email.contains("=") || Character.isDigit(email.charAt(0))) {

        c.mssg = "Please enter the email in the required form";
    }
    ResultSet CkUniqueness = stmt.executeQuery(
            "select username,email from customer where username = '" + un + "' or email= '" + email + "'");
    if (CkUniqueness.next()) {
        c.mssg = "Please enter the email in the required form";
    } else {
        stmt.executeUpdate("INSERT INTO customer (username, email, password,accountno,cluster)values" + "('" + un
                + "','" + email + "','" + pw + "','" + "00000000000" + "','" + "general" + "');");
        ResultSet custWithpw = stmt
                .executeQuery("select username,password,idcustomer from customer where username = '" + un
                        + "' and password= '" + pw + "'");

        if (custWithpw.next()) {
            c.customerid = custWithpw.getString("idcustomer");
            c.username = custWithpw.getString("username");
            c.mssg = "success";

    }

    }
    return c;
}
```

Then make it as a Service.

```
//////////////////// sign up service//////////////////////////
@RequestMapping(value = "/SignUp", method = RequestMethod.POST, produces = "application/json")
public customer signup(@RequestParam("uname") String username, @RequestParam("email") String email,
        @RequestParam("pw") String password) throws ClassNotFoundException, SQLException {
    MainModel mm = new MainModel();
    customer dd = mm.SignUp(username, email, password);
    return dd;
}
```

CASHIERY
Just scan it

The first recommendation algorithm (Clustering), the first method classify the customers in into categories by changing its category in the database from general to specific category, the second function display the offers related to the this customer according to his category

```
//////////////////////////////////// clusering for customer....( metod-1/)//////////////////////////////

public void Clustering(String CustID) throws SQLException, ClassNotFoundException {

    Class.forName("com.mysql.jdbc.Driver");
    java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_gp", "root", "rooot");
    java.sql.Statement stmt = con.createStatement();

    ResultSet GetCluster = stmt
            .executeQuery("select sum(quanity), category from history,receipt,product where custID= " + CustID
                    + " and receiptnum=orderid and receipt.barcode=product.barcode group by category order by sum(quanity) desc");
    int c = 0;
    while (GetCluster.next() && c == 0) {
        c++;

    }
    stmt.executeUpdate("update customer set cluster= " + "'" + GetCluster.getString("category")
            + "' where idcustomer= " + "'" + CustID + "'");

}


 // method --2
 public Vector<String> RecommendationBasedOnClusterAndOffers(String CustId)
         throws ClassNotFoundException, SQLException {
     Class.forName("com.mysql.jdbc.Driver");
     java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_gp", "root", "rooot");
     java.sql.Statement stmt = con.createStatement();
     ResultSet Recommendations = stmt.executeQuery("select productname,description from sale,product,customer where"
             + " product.name=sale.productname and cluster=category and customer.idcustomer= '" + CustId + "'");

     Vector<String> Recomm = new Vector<String>(); // return values as vector

     while (Recommendations.next()) {
         System.out.println(
                 Recommendations.getString("productname") + "-->" + Recommendations.getString("description"));
         Recomm.addElement(
                 Recommendations.getString("productname") + "-->" + Recommendations.getString("description"));
     }

     return Recomm;
 }
```

Then make it as a service.

```
//////////////////////////// First recommendation service//////////////////////
@RequestMapping(value = "/clusteringg", method = RequestMethod.POST)
public obj clustering(@RequestParam("custid") String custid) throws ClassNotFoundException, SQLException {
    CustomerModel k = new CustomerModel();
    obj d = new obj();
    k.Clustering(custid);
    d.result = k.RecommendationBasedOnClusterAndOffers(custid);

    return d;
}
```

The second recommendation algorithm (Apriori), the first method take parameter the items and make combination of 2 items. The second method take the combination of 2 items and build the association rules.

CASHIERY
Just scan it

```
////////////////////////////// association rules ////////////////////////////
/// method 1
public Vector<Vector<String>> CreateCombinationsOf2(Vector<String> Attributes) {

    Vector<Vector<String>> Combinations = new Vector<Vector<String>>();
    Vector<String> Temp = new Vector<String>();

    int C = Attributes.size() - 1;

    for (int i = 0; i < Attributes.size(); i++) {
        while (C >= 0) {
            int j = i;
            for (j = i; j < Attributes.size() - C; j++) {
                Temp.add(Attributes.elementAt(j));

            }
            if (!Temp.isEmpty() && Temp.size() == 2) {
                Combinations.add(Temp);
                // System.out.println(Temp);
            }

            C--;
            Temp = new Vector<String>();
        }
        C = Attributes.size() - 1;
        Temp = new Vector<String>();
    }

    C = Attributes.size() - 1;
    Temp = new Vector<String>();

    for (int i = 0; i < Attributes.size(); i++) {
        Temp.add(Attributes.elementAt(i));
        // System.out.println("hsht9l 3la da "+Attributes.elementAt(i));
```


```
        // System.out.println("hsht9l 3la da "+Attributes.elementAt(i));
        int Step = 2;
        while (Step < Attributes.size()) {
            // System.out.println("el step >>> "+Step);
            while (C >= 0) {
                // System.out.println("bbd2 l while ml awl");
                for (int j = i + Step; j < Attributes.size() - C; j++) {
                    Temp.add(Attributes.elementAt(j));
                    // System.out.println("zwdt >>> "+Attributes.elementAt(j));

                }
                if (!Temp.isEmpty() && !Combinations.contains(Temp) && Temp.size() == 2) {
                    Combinations.add(Temp);
                }

                C--;
                Temp = new Vector<String>();
                Temp.add(Attributes.elementAt(i));

            }
            Step++;
            C = Attributes.size() - 1;
        }
    }

    for (int i = 0; i < Combinations.size(); i++) {
        for (int j = 0; j < Combinations.elementAt(i).size(); j++) {
            System.out.print(Combinations.elementAt(i).elementAt(j));
        }
        System.out.println();
    }

    return Combinations;
}
```

CASHIERY
Just scan it

```
/////// method 2 (for scduler task)
public void BuildAssociation() throws SQLException, ClassNotFoundException {

    Class.forName("com.mysql.jdbc.Driver");
    java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_gp", "root", "rooot");
    java.sql.Statement stmt = con.createStatement();
    ResultSet ItemWithCount = stmt
            .executeQuery("select distinct barcode ,count(distinct orderid) as c from receipt group by  barcode ");
    Vector<ItemWithCount> IWC = new Vector<ItemWithCount>();
    while (ItemWithCount.next()) {
        ItemWithCount temp = new ItemWithCount();
        temp.Item = ItemWithCount.getString("barcode");
        temp.count = ItemWithCount.getInt("c");
        IWC.add(temp);
    }
    // should return 1 number only
    ResultSet NumOfTrans = stmt.executeQuery("select count(distinct orderid) as c from receipt");
    int numofTransactions = 0;
    while (NumOfTrans.next()) {
        numofTransactions = NumOfTrans.getInt("c");
    }

    System.out.println(numofTransactions);

    int MinSupCount = 5;
    for (int i = 0; i < IWC.size(); i++) {
        if (IWC.elementAt(i).count < MinSupCount) {
            IWC.remove(i);
            i--;
        }
    }

    Vector<String> IWCINVector = new Vector<String>();
    for (int i = 0; i < IWC.size(); i++) {
        IWCINVector.add(IWC.elementAt(i).Item);
    }
    Vector<Vector<String>> combinationsof2 = new Vector<Vector<String>>();
```

CASHIERY
Just scan it

```
Vector<String> IWCINVector = new Vector<String>();
for (int i = 0; i < IWC.size(); i++) {
    IWCINVector.add(IWC.elementAt(i).Item);
}
Vector<Vector<String>> combinationsof2 = new Vector<Vector<String>>();
combinationsof2 = CreateCombinationsOf2(IWCINVector);
Vector<ItemWithCount> TwoItemSet = new Vector<ItemWithCount>();
for (int i = 0; i < combinationsof2.size(); i++) {
    ItemWithCount temp = new ItemWithCount();
    temp.Item = combinationsof2.elementAt(i).elementAt(0) + "," + combinationsof2.elementAt(i).elementAt(1);
    ;
    TwoItemSet.add(temp);
    // TwoItemSet.elementAt(i).Item=combinationsof2.elementAt(i).elementAt(0)+","+combinationsof2.elementAt(i).elementAt(1);
}
for (int i = 0; i < combinationsof2.size(); i++) {
    ResultSet CountOfComb2 = stmt.executeQuery("select count(distinct orderid) as c from receipt T1 "
            + "where exists(select count(distinct orderid) from receipt T2 "
            + "where orderid=T1.orderid and barcode= " + "'" + combinationsof2.elementAt(i).elementAt(1) + "'"
            + " and T1.barcode= " + "'" + combinationsof2.elementAt(i).elementAt(0) + "'" + " ) ");
    if (CountOfComb2.next()) {
        TwoItemSet.elementAt(i).count = CountOfComb2.getInt("c");
    }
}
// System.out.println("after query----------> ");

for (int i = 0; i < TwoItemSet.size(); i++) {
    if (TwoItemSet.elementAt(i).count < MinSupCount) {
        TwoItemSet.remove(i);
        i--;
    }
}

Vector<Integer> IWCINVectorCounts = new Vector<Integer>();
for (int i = 0; i < IWCINVector.size(); i++) {
    ResultSet CountofThis = stmt.executeQuery("select count(distinct orderid) as c from receipt "
            + "where barcode= " + IWCINVector.elementAt(i) + " ");
    if (CountofThis.next()) {
```


```
            if (CountofThis.next()) {
                IWCINVectorCounts.add(CountofThis.getInt("c"));
            }

        }
        double minConfidence = 0.2;
        Vector<String> Pre = new Vector<String>();
        Vector<String> Post = new Vector<String>();
        Vector<Double> Confidence = new Vector<Double>();
        for (int i = 0; i < TwoItemSet.size(); i++) {
            String[] temp = TwoItemSet.elementAt(i).Item.split(",");
            Pre.add(temp[0]);
            Post.add(temp[1]);
            for (int j = 0; j < IWCINVector.size(); j++) {
                if (IWCINVector.elementAt(j).equals(temp[0])) {
                    Confidence.add(TwoItemSet.elementAt(i).count / IWCINVectorCounts.elementAt(j));
                    break;
                }
            }
            Pre.add(temp[1]);
            Post.add(temp[0]);
            for (int j = 0; j < IWCINVector.size(); j++) {
                if (IWCINVector.elementAt(j).equals(temp[1])) {
                    Confidence.add(TwoItemSet.elementAt(i).count / IWCINVectorCounts.elementAt(j));
                    break;
                }
            }
        }
        for (int i = 0; i < Confidence.size(); i++) {
            if (Confidence.elementAt(i) < minConfidence) {
                Confidence.remove(i);
                Pre.remove(i);
                Post.remove(i);
                i--;
            }

        }

            Post.remove(i);
            i--;
        }

        }

        // hn create table fady fl DB asmo associations(pre,post,confidence)
        // System.out.println("before delete query----------> ");

        stmt.executeUpdate("DELETE FROM associations ;");
        // System.out.println("after delete query----------> ");

        for (int i = 0; i < Confidence.size(); i++) {
            stmt.executeUpdate("INSERT INTO associations(pre,post,confidence) VALUES ('" + Pre.elementAt(i) + "','"
                    + Post.elementAt(i) + "','" + Confidence.elementAt(i) + "');");
        }

        // System.out.println("after insert query----------> ");

    }
```

Then make it as service.

```
////////////////////// build assocation for scheduler task//////////////////
@RequestMapping(value = "/BuildAssociation/", method = RequestMethod.GET)
public void buildAss() throws ClassNotFoundException, SQLException {
    CustomerModel m = new CustomerModel();
    m.BuildAssociation();

}

    ////////////////////// Second recommendation services //////////////////////
   @RequestMapping(value = "/association", method = RequestMethod.POST)
   public obj Association(@RequestParam("barcode") ArrayList<String> barcodes)
           throws ClassNotFoundException, SQLException {
       CustomerModel m = new CustomerModel();
       obj d = new obj();
       Vector<String> recomm = new Vector<String>();
       for (int i = 0; i < barcodes.size(); i++) {
           Vector<String> r = m.getAssociation(barcodes.get(i));
           for (int j = 0; j < r.size(); j++) {
               recomm.addElement(r.elementAt(j));
           }
       }
       d.result = recomm;
       return d;

   }
```

CASHIERY
Just scan it

4-1-2 The Frontend Code

This Code is written in java language in the android studio as the frontend code which is calling services.

Login calling service

```
ApiManager.getApiService().Login(username,password)
        .enqueue(new Callback<User>() {
            @Override
            public void onResponse(Call<User> call, Response<User> response) {
                HideProgressDialog();
                User myresponse=response.body();
                Log.e( tag: "resp",response.toString());
                if (myresponse.getMssg().equals("Fail")){
                    ShowMessage( title: "error", mess: "wrong username or password");
                    Log.e( tag: "erro", msg: "fail");
                }else {
                    ShowMessage( title: "", mess: "success login");
                    Log.e( tag: "success", msg: "success");
                    AppData.user=myresponse;
                    Log.e( tag: "username", AppData.user.getCustomerid());
                    Log.e( tag: "username", AppData.user.getUsername());

                    //complete
                    Intent intent = new Intent(getApplicationContext(), Markets.class);
                    intent.putExtra( name: "username", us);   // send  us to first recomm
                    Toast.makeText(getApplicationContext(),  text: "username" + us, Toast.LENGTH_SHORT).show();
                    startActivity(intent);
                }
            }

            @Override
            public void onFailure(Call<User> call, Throwable t) {
                HideProgressDialog();
                Log.e( tag: "loginerror",t.getLocalizedMessage());
                ShowMessage( title: "error ", mess: "cannot login");
            }
```

The signup calling service

```
ApiManager.getApiService().SignUp(username, emails, password)
        .enqueue(new Callback<User>() {
            @Override
            public void onResponse(Call<User> call, Response<User> response) {
                HideProgressDialog();
                User myresponse = response.body();
                Log.e( tag: "resp", response.toString());
                if (myresponse.getMssg().equals("Please enter the email in the required form")) {
                    ShowMessage( title: "error",  mess: "please try again");
                } else if (myresponse.getMssg().equals("This username or Email already exist, please try again")) {
                    ShowMessage( title: "error",  mess: "please try again");
                } else {
                    ShowMessage( title: "",  mess: "welcome to Cashiery");
                    AppData.user=myresponse;
                    //complete
                    Intent intent = new Intent(getApplicationContext(), Markets.class);
                    startActivity(intent);
                }
            }

            @Override
            public void onFailure(Call<User> call, Throwable t) {
                HideProgressDialog();
                ShowMessage( title: "error ",  mess: "cannot signup");
            }
        });
```

CASHIERY
Just scan it

The clustering calling function

```
public void  cluster_method(String Unn){
    String custid=AppData.user.getCustomerid();
    final TextView[] stringTextView = {(TextView) findViewById(R.id.recomm)};
    ApiManager.getApiService().clustering(custid)
            .enqueue(new Callback<FirstRecommResponse>() {
                @Override
                public void onResponse(Call<FirstRecommResponse> call, Response<FirstRecommResponse> response) {
                    // HideProgressDialog();
                    FirstRecommResponse myresponse = response.body();
                    Log.e( tag: "resp", response.toString());
                    if (myresponse.getResult().isEmpty()){

                        ShowMessage( title: "",  mess: "u dont have any recommendations");
                    } else  {

                        recommendations=new ArrayList<String>();
                        for(int i=0; i <  myresponse.getResult().size(); i++) {
                            recommendations.add(  myresponse.getResult().get(i));

                        }
                        for(int i=0; i < recommendations.size(); i++) {

                            stringTextView[0].setText(stringTextView[0].getText() + recommendations.get(i) + "\n");
                        }}} @Override
                public void onFailure(Call<FirstRecommResponse> call, Throwable t) {
                    // HideProgressDialog();
                    ShowMessage( title: "error ",  mess: "can not get any recommendations");
                }
            });
```

CASHIERY
Just scan it

The association calling service

```
public void  Association_method(ArrayList<String> realproducts2)
{
    ApiManager.getApiService().Association(realproducts2)
            .enqueue(new Callback<SecondRecommResponse>() {
                @Override
                public void onResponse(Call<SecondRecommResponse> call, Response<SecondRecommResponse> response) {
                    // HideProgressDialog();
                    SecondRecommResponse myresponse = response.body();
                    Log.e( tag: "resp", response.toString());
                    if (myresponse==null||myresponse.getResult()==null||myresponse.getResult().size()==0){
                        ShowMessage( title: "",  mess: "u dont have any recommendations");

                    }
                    else  {

                        recommendations=new ArrayList<String>();
                        for(int i=0; i <  myresponse.getResult().size(); i++)
                            recommendations.add(myresponse.getResult().get(i));
                        for(int i=0; i < recommendations.size(); i++)
                            stringTextView.setText(stringTextView.getText() + recommendations.get(i) + "\n");
                    }
                }

                @Override
                public void onFailure(Call<SecondRecommResponse> call, Throwable t) {
                    // HideProgressDialog();
                    ShowMessage( title: "error ",  mess: "can not get any recommendations");
                }
            });
```

4-2 Used tools

We used number of tools, APIs and libraries which help us so much to get our application in real life. We used

(1) *Android Studio* were we build the application frontend, android studio has good capabilities and APIs which we used like: *Zxing*, which enable us to get connected to the mobile camera and make the camera able to read different barcodes, *Volley*, *Retrofit* enable us to call the services in the backed code and run them quickly.

(2) *Postman* this tool enable us to execute and test the APIs and services we implemented in spring.

(3)*Spring boot* this a framework with java platform which enable us to convert our functions and recommendation system into services which can be called and used in any other application.

(4) *MySQL* we used this database server to design and implement our database.

(5) *NetBeans and Eclipse* are the two IDEs used to implement the backend code and converting it into services.

5-Testing and Evaluation

5-1 Test Cases

We used unit testing to test our application functions, here are some test cases to the most important functions in our system.

Test Case	Input Data	Expected Output	Actual Output	Status
Login	User name→User_2 Password →1235	Display “success login” message and go to markets page	Displayed “success login” message and went to markets page	Passed

Test Case	Input Data	Expected Output	Actual Output	Status
Login	User name→User_2 Password →1235	Display “success login” message and go to markets page	Displayed “cannot login” message	Failed , Due to the application and the database not on the same network

Test Case	Input Data	Expected Output	Actual Output	Status
Signup	User name→ zain Email →zain@gmail.com Password →1234	go to login again page	Go to the login page	Passed

Test Case	Input Data	Expected Output	Actual Output	Status
Signup	User name→aya Email →aya@gmail.com Password →1234	go to login again page	Displayed “cannot signup” message	Failed , Due to the username and the email found already in the database

Test Case	Input Data	Expected Output	Actual Output	Status
Clicking market buttons	Click Carrefour button	go to home page	Went to home page	Passed

Test Case	Input Data	Expected Output	Actual Output	Status
Clicking market buttons	Click metro button	go to home page	Stay at the same page	Failed , due to bug in the code

Test Case	Input Data	Expected Output	Actual Output	Status
Scan function	Scan chocolate bar	The barcode will appear in a toast and the chocolate bar will be added in the checklist	Displayed message “barcode is not found”	Failed , Due to the chocolate bar not found in the database

Test Case	Input Data	Expected Output	Actual Output	Status
Scan Function	Scan mini notebook	The barcode will appear in a toast and the mini notebook will be added in the checklist	The barcode will appear in a toast and the mini notebook will be added in the checklist	Passed

Test Case	Input Data	Expected Output	Actual Output	Status
Click market buttons (to test clustering)	Click Carrefour button	Show the offers in Carrefour related to this user	Displayed message “cannot found recommendations”	Failed , Due to bug in the calling service code

Test Case	Input Data	Expected Output	Actual Output	Status
Click market buttons (to test clustering)	Click Carrefour button	Show the offers in Carrefour related to this user	Showed the user offers in Carrefour	Passed

Test Case	Input Data	Expected Output	Actual Output	Status
Scanning 2 items (to test association rules)	Scan chocolate barcode and chipsy barcode	Show recommendation to buy Pepsi cane	Show recommendation to buy Pepsi cane	Passed

Test Case	Input Data	Expected Output	Actual Output	Status
Scanning 2 items (to test association rules)	Scan chocolate barcode and chipsy barcode	Show recommendation to buy Pepsi cane	Displayed message “no recommendation found “	Failed , Due to bug in association rules code

Test Case	Input Data	Expected Output	Actual Output	Status
Menu buttons	Click offers button in menu	Display all the offers in this market	Display all the offers in this market	Passed

Test Case	Input Data	Expected Output	Actual Output	Status
Menu buttons	Click about us button in menu	Display the about us page	Display the connect us page	Failed , Due to bug in the code

5-2 Model Accuracy

As mentioned above we used some mining algorithms to make simple recommendation system. We measure the accuracy of association rules models and its results by calculating the minimum support first we tried the min support to be half of the transaction numbers but it caused zero items satisfying the min support so we minimized the min support to be 5 as we have few transactions only but the higher the min support the higher accuracy we got as well as we used the confidence measure first we assumed the minimum confidence to be 0.5 but it is impossible to find specific two products in the half of the transactions so we minimized the minimum confidence to be 0.2 to find 2 item sets satisfy the min confidence and it could be by chance as the higher the min confidence the more accurate results.

5-3 Possible Future Work

We intend to improve and widen our application:

- Make our application support more markets.
- Get the access to these markets server and database to get more accurate results.
- Add complex recommendation systems to help customers more.
- Make our database globally not locally.
- Develop the paying part in the application.
- Delivery facilities.
- Add branches feature to each market.

6-Lessons Learnt

We learnt a lot from working on such project, we acquired technical, teamwork, presentation skills.

The technical skills like: learning Spring boot technology and how converting java code into service, dealing with postman and how test and execute the service, dealing with android and how building application UI , also dealing with Zxing, Volley and Retrofit libraries to make camera able to read different types of barcodes and to be able to call the services written in java as backend

The soft skills like: working in a team and how we can aggregate the tasks of the team members, dealing with big failure in a part of the projects and solve it well, the ability to make good presentations and how to present well under stress and limited time restriction.

6-1 Limitations

We faced a lot of difficulties like:

- Dealing with new technologies we have been never dealt with before like: Android, Zxing Library , Volley Library , Retrofit Library , Postman Environment and Spring boot framework.
- The difficulty in deciding the suitable mining algorithms to build our recommendation system and the suitable language to implement it.
- The difficulty to find library deals with mobile camera and make the camera able to read different types of barcodes and it is too complex to understand how to make it work, this step take us a lot of time as it is a new thing we learn.
- The difficulty of finding the data and designing the database , first we design it globally on global server, but after that we realized that there is no need to make it global, as we are working on training data set and demo application, then we designed it locally, this step take a lot of time to be developed.

7-References

7-1 Knowledge Main References

https://nb.vse.cz/~cernym/preprinty/p3.pdf

http://www.parkeravery.com/pov_Retail_Clustering_Methods.html

https://www.slideshare.net/zafarjcp/data-mining-association-rules-basics

https://dzone.com/articles/running-on-time-with-springs-scheduled-tasks

https://www.mkyong.com/java/how-to-run-a-task-periodically-in-java/
https://automationrhapsody.com/introduction-postman-examples/

http://toolsqa.com/postman/post-request-in-postman/

https://spring.io/guides/gs/spring-boot/

https://developer.android.com/training/volley/

http://www.monmouthcountyparks.com/Documents/13/Systems%20Analysis%20and%20Design.pdf

http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf

7-2 APIs, Tools and Libraries

https://www.funcode-tech.com/Fun2D_QRCode_and_Barcode_SDK_for_Mobile_en.html?gclid=CjwKCAjwyMfZBRAXEiwA-R3gM4e7lVqulCxqAb446GRqTMPjEXW3mRCLdlbnrGtsuAufvmMqbO-MpRoCBf0QAvD_BwE

https://jar-download.com/?search_box=mysql-connector-java-5.1.44

http://square.github.io/retrofit/

https://plugins.jetbrains.com/plugin/8113-pojo-generator

https://www.getpostman.com/

https://mvnrepository.com/artifact/com.squareup.retrofit2/converter-gson/2.4.0

https://android.googlesource.com/platform/frameworks/volley

https://github.com/google/volley

https://app.ganttpro.com/#!/app/home

https://creately.com/

http://www.eclipse.org/downloads/eclipse-packages/

https://netbeans.org/downloads/

https://www.mysql.com/downloads/