*The Academic College of Tel Aviv-Yaffo*

*Web Application Development*

*Project Part A*

*Bar Wachtel ID: 301797445*

*Yotam Lender ID: 304858509*

*Dmitri Poleacov ID: 323468629*

*Application Specifications*

***The development team:***

**Bar Wachtel** – student on the 4th year of Computer Science BSc

Previous experience: C, C++, C#, Java, JS, HTML, CSS, SQL

Responsibilities: Client side, Server side, dB: MySQL

**Dmitri Poleacov** – student on the 4th year of Computer Science BSc

Previous experience: C, C++, C#, Java, JS, HTML, CSS, SQL

Responsibilities: Client side, Server side, dB: MySQL

**Yotam Lender** – student on the 4th year of Computer Science BSc

Previous experience: C, C++, C#, Java, JS, HTML, CSS, SQL

Responsibilities: Client side, Server side, dB: Redis

***Application Goals:***

*How many times you went to bar with your friends and at the end of the evening everyone is too drunk to remember and calculate all the items they order and you start looking for a superhero to make all the job for everyone.*

***X*** *Is your superhero! Our App makes splitting the bill as easy as taking a picture.*

***Functional Requirements:***

The application's purpose is to solve the age old problem of splitting the bill.

Intended for use on mobile devices, of which at least one must be equipped with a camera.

All users who wish to take part in the bills payment will log onto the application and be required to perform Facebook login.

The user will then be presented with two options –

Join an existing bill or start payment on a new bill.

***Methods Company's Activity:***

Each user logs in the application using his/her Facebook account, which makes it easy to access and require the user with no-registration-sign-in process.

After a user signs-in, he/she can enjoy all services our application provide:

* Starting a bill
* Joining a bill
* Watching previous bills

***Business Potential:***

**Related Field/s of Business:**

The food & nightlife industries would be the main focus, seeing as both of these have a tight relation to handling transactions between a parties of clients.

**Application environment and target audience:**

We will analyze the application environment with the Porter's Five Forces Framework.

**Customers:**  Our users will be everyone that wants to make his life easier in splitting the bill in bars/restaurants. The app suites to every age group.

**Suppliers:**  The app doesn't provide any material product, so there are no suppliers.

**Replacements:**  There is no identical application that offers all the services that Billy offers.

**Competition:** There are some alternatives to our application, applications that are helping to split the bill. The most commons are Tab, Billr, Plates, Tip-n-Split, we will discuss the difference in detail later.

**Entry of New products:**  The OCR technology is developed now to support all languages, so as quality of the technology will be good enough to support customer needs, new application will be in stores. Our mission is to make the most user friendly application, the development team will concentrate on the UI and the quality of the OCR scan the most, which will make no one to want to replace it in the future.

***Growth Potential:***

Since the application requires very little hassle to access, and has a usage that matches groups of individuals its adoption could be viral.

The main hinder on growth would be usage of the application in other languages,

Depending on the quality of OCR technologies in relation to new languages would not always be reliable.

In addition using the application with other languages would possibly require specific modifications in regard to the text parsing and analyzing phase, meaning that development to other markets would need additional development & support.

***Risk assessment:***

***Competitors to the application –***

As we said before there are competitors to the app so it's very important to make the application user friendly as possible, our mission is to make the life easier to our customers so the functionality of the app has to be easy

And well understood that it will fit every age group.

***Detailed comparison between X and the competitors –***

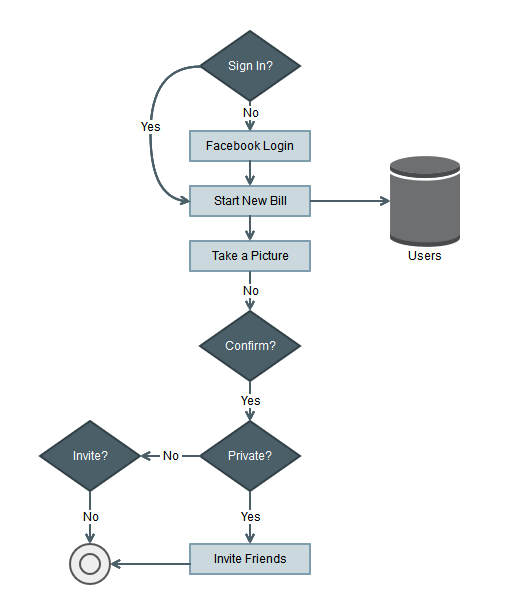
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***TAB*** | ***Billr*** | **Plates** | **Billy** | ***X***  *Our app* |
| *Available in Israel Store* | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png |
| *Free app* | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png |
| *Using OCR technology* | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png |
| *Having a user profile* | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png |
| *Possibility to see user history* | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png |
| *Sharing with fiends* | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\X.png | C:\Users\Dima\Desktop\סמסטר ב2015\OurAndroidApp\screenshots\V.png |

***Business Model:***

Seeing as the application has the purpose of easing a task that is possible to perform without its help, users would not be willing to spend money acquiring it.

Furthermore a user would not be able to use this application with other non-paid users. For these reasons the application will use a Freemium model.

***Business Processes***

******

***Starting a bill*** *-*

If the user selects to start a new bill, the application will switch to the users camera and ask him to take a clear photo of the bills content.

The photo will be displayed to the user, prompting him/her if the result is satisfying else an option to retake the photo.

On confirmation the photo will be uploaded to the server,

The server will perform OCR (Optical character recognition) in order to extract the photos content as text.

This text will be parsed and analyzed, in order to create an inventory from the bill,

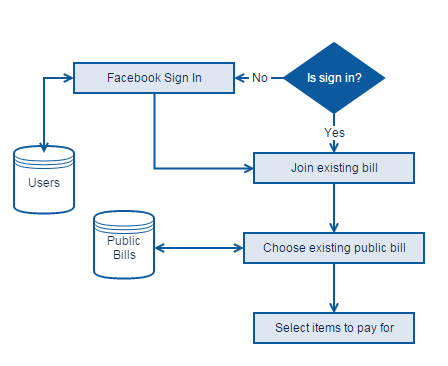
Each item on the bill will be matched a price.

The results will be returned to the user and each item's name & price will be displayed, the user has an option to edit any incorrect results whether in the items name or price.

When the user is satisfied with the results, they will be prompted if they wish to add the tip beforehand, either as an exact amount or a percentage of the bill.

Finally the user will be able to name the bill,

And they must select whether to invite users to the bill or open access to the bill publicly.

******

***Joining a bill*** *-*

If the user selects to join a bill, the screen will display bills the user has been invited to and public bills opened by friends (Facebook friends).

The user will select a bill, and then select to join the bill.

When paying a bill all the bills inventory is displayed,

The items name and price are displayed on a single row and +/- buttons for item selection, the user selects the items he/she wishes to pay for and the sum is calculated.

The user has the option to add tip to the calculation, either as a single amount or a percentage of the sum of their selected items.

When the user has selected all items they wish to pay for, they submit their selections to all the other users.

The user can edit their selections and resubmit at any time before the bill is closed.

Each user's view of the bill's inventory will be updated with other users' selections, until all items in the bill have been selected & submitted.

At this point the bills creator will be asked if everything is in order and the bill can be closed.

***ER Diagram –***

Invites

0..n

0..n

|  |  |  |  |
| --- | --- | --- | --- |
| User | | | |
| #Serial Number | <pi> | Integer | <M> |
| First Name |  | Variable characters (30) | <M> |
| Last Name |  | Variable characters (30) | <M> |

0…n

Opens

0…n

|  |  |  |  |
| --- | --- | --- | --- |
| Bill | | | |
| #Serial Number | <pi> | Integer | <M> |
| Is Private |  | Boolean | <M> |

0…n

Contains

0…n

|  |  |  |  |
| --- | --- | --- | --- |
| Item | | | |
| #Serial Number | <pi> | Integer | <M> |
| Price Per Unit |  | Float | <M> |
| Amount |  | Integer | <M> |

The main entities of the system are the User entity and the Bill entity. The User entity data is taken from the Facebook user public information. Each user can start a bill and invite friends to participate in the bill. These actions are described in the diagram above.

***Tables representing the ER diagram –***

|  |  |  |
| --- | --- | --- |
| Bills | | |
| Bill Identifier | int | <pk> |
| Facebook Identifier | int |  |
| Private | bool |  |

|  |  |  |
| --- | --- | --- |
| Users | | |
| Facebook Identifier | int | <pk> |
| First name | varchar(30) |  |
| Last name | varchar(30) |  |

|  |  |  |
| --- | --- | --- |
| Bills Summary | | |
| Unique Identifier | int | <pk> |
| Bill Identifier | int | <fk> |
| Item Identifier | int | <fk> |
| Member Facebook Identifier | int | <fk> |
| Amount Paid For | int |  |

|  |  |  |
| --- | --- | --- |
| Items | | |
| Item Identifier | int | <pk> |
| Bill Identifier | int | <fk> |
| Item Name | varchar(50) |  |
| Price (per unit) | float |  |
| Amount | int |  |

|  |  |  |
| --- | --- | --- |
| Bills Group/Members | | |
| Group identifier | int | <pk> |
| Bill identifier | int | <fk> |
| Member Facebook Identifier | int | <fk> |

***Tables’ description –***

* **Users table** – the table contains all the users in the application using their Facebook accounts. Each user has its unique identifier provided by the Facebook API. Each user has its first and last names saved in the table.
* **Bills table** – the table contains all the bills that were opened and paid by users.

The primary key of the table is a unique serial number for each bill. In addition to the primary key, the table has a foreign key to the Users table which represent the user who opened the bill.

* **Items table** – the table contains all items for a specific bill. The primary key of the table is a unique serial number. Also in the table, a foreign key to the Bills table which represents which bill the item belongs to.
* **Bills Members** – the table contains all the members participated in the bill. Each member has its unique serial number, which represents the primary key. The Bill identifier and Facebook identifier are foreign keys to the Bills table and Users table respectively.
* **Bills Summary table** – the table describes who paid for each item in a specific bill, using foreign keys to the Bill table, Items table and Users table. The Member Facebook Identifier must be a member in the Bills Members table for the specific bill and also a valid user in the Users table.

***MySql Database Queries –***

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Web request type | Query category | Explanation |
| getOpenBills | Get | complicated SELECT query | when a user enters "Join Bill" screen a list of friends open bills is displayed |
| createNewBill | Post | simple INSERT query | after a user opens a bill, takes photo and confirms content, the open bill is added to database |
| payForItems | Post | complicated UPDATE query | mark bill items as payed for, trying to pay for a payed item will fail |
| getPaymentHistory | Get | SELECT query using GROUP BY | get all items payed for by the user, add sum their cost |
| cancelBill | Post | DELETE query | If the bill creator cancels the bill, need to remove entity from database |

***In-Memory Database – Redis***

After a bill has been initiated, each bill member has to choose which items he is willing to pay from the bill’s items. To provide this functionality in an efficient way, the open bills, and all their related data, will be saved in the in-memory database, using Redis.

|  |  |
| --- | --- |
| Set – open bills | |
| user:1000:bill:200 | item\_1\_id, item\_2\_id, item\_3\_id … |

All open bills will be saved in a Set. The key will contain the unique identifier of the user who opened the bill and the new bill identifier. The value will contain all the items of the bill. Each item in the list will contain the item identifier.

|  |  |
| --- | --- |
| Hash – item | |
| bill:200:item:3 | {name:Goldstar,price:50,amount:3} |

The item’s data will be saved in a Hash. The key will contain the bill identifier and the item identifier. An item object in a Hash will contain the item’s price per unit, amount and name.

|  |  |
| --- | --- |
| List– user selection | |
| paidfor:user:1000:bill:200 | Item\_2\_id ,item\_3\_id |

Additionally, the users’ selection will be saved in a List. The key will contain the user and bill identifiers. The value will contain a list of items paid by the user.

|  |  |
| --- | --- |
| Hash– paid item | |
| user:1000:bill:200:item:3 | {amount:2,price:100} |

Each paid for item will be saved in a Hash containing the amount paid for and total price for the item. The key will be the user, bill and item identifier.