CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Computer Science

RESTAURANT SYSTEM MOBILE CLIENT

BACHELOR THESIS

Study Programme: **Software Technology and Management**

Branch of Study: **Software Engineering**

Thesis Advisor: **MEng. Martin Komarek**

Tomas Hogenauer

Prague 2016

DECLARATION

I hereby declare that this bachelor thesis is the product of my own independent work and that I have clearly stated all information sources used in the thesis.

Date Signature

ABSTRACT

The aim of this thesis is to develop an Android application for waiters in restaurants. The app will offer the ability to manage accounts, orders and activities related to it. It should offer good GUI for easy use and a faster way to complete user requirements. The app will communicate with restaurant web applications over REST API.

1 TABLE OF CONTENTS

2 INTRODUCTION

We are living in the era of information technologies. It is changing our lives and ma-king it easier. Since we have found the power of technology and embraced it as a helping tool, we are trying to make it more useful. Technology is not staying the same; it is changing and improving almost every day. We are preoccupied with the wish to make new technology better.

After the trend of desktop PCs and laptops, there was a newfound interest in mobile devices and tablets. In 2014, the amount of mobile users reached the same amount as desktop users (almost 1,700 million users) and that number is still increasing [1]. According to a forecast of the statistic portal Statista, since 2015, the number of shipped tablets will be bigger than the number of shipped tablets and laptops together [2]. Companies are realizing this new trend and starting to focus on the mobile industry more than ever before. Nowadays it is a quickly growing industry that offers many opportunities.

It is a huge opportunity for software engineers, but also big problem for them, because their products have to satisfy a huge quantity of customers. On the market, there are many devices with different software and hardware, making the job of the software engineers more complicated. They must look for the best solution for how to satisfy the market in the most convenient way possible. One of these steps was to replace desktop applications and web applications or redevelop them for mobile/tablet devices.

An advantage of web applications is that they do not depend on the platform where they run. There is also an easy way to update applications for all users. Another advantage is that all computations are solved on a server, so it can run on devices with outdated hardware as well. According to an increasing number of mobile and tablet users, mobile applications are convenient. They mostly communicate with web application over an API (Application Program Interface), so all computations are also solved on a server. Nowadays, mobile and web applications are the most convenient application and it is necessary to go with this trend if we wish to satisfy the market.

2.1 PROJECT CASHBOB

CashBob is a solution for restaurant management to bring a new and easier way of managing restaurants and carrying out all activities related to restaurants [3]. CashBob is a long-lived project focused on offering the best solution, because there are many similar solutions on the market.

Developing CashBob was the focus of many student projects and thesis [4]. The first CashBob application was developed as a desktop application with many modules. It was to be used by restaurant management (to manage employees, orders and restaurant storages), or by waiters or bartenders (to manage accounts and orders, make payments and print receipts).

CashBob was later developed as a mobile application. It worked on devices with Android and Apple operation systems. This application offered waiters the ability to manage restaurant accounts, order items, charge items and move items between accounts.

This year CashBob was developed as a web application. It would replace the desktop version, because of all the advantages that web applications offer in comparison to desktop applications. It would offer interface REST API, which could be used by other CashBob application. It would centralize the access to the database and provide control of who has this access.

3 PROJECT OVERVIEW STATEMENT

The aim of this thesis is to make CashBob as a mobile application. It is to be used by waiters in restaurants and offers them the ability to manage accounts and orders and perform activities related to it.

CashBob for mobile devices has been developed once before [5]. It worked on devices with Android and Apple operation systems and offered the same possibility as a web application, which is the aim of this thesis. The problem of this version was that it had not satisfied a submitter and it did not fit the expectations. Therefore, it was necessary to develop a new version.

Our version should avoid all previous failures and meet all requirements in the best way to satisfy the submitter. It will be logically designed and offer the ability to accomplish any task quickly and properly. It will also offer understandable GUI for any user. Our version will be offered only for Android devices, specifically focused to tablets. The app will communicate with restaurant web application over REST API. The result of this thesis is the app, which would be deployed and available for use.

Our application has to provide options for any actions that would be done by a restaurant waiter. These would be:

- Create and close accounts
- Make an order for an account
- Move items between accounts
- Make a payment of full account or parts of account
- Print a receipt on a mobile printer
- Print a receipt on a server printer

When the user creates a new account, they can choose the name of the account and attach a table. If they choose not to fill in a name, the chosen table will be the name of the account. If a table is also not chosen, the name of the account will be the time when the account was created. This account is called a quick account. The quick account feature is used to make a fast new order or to move items to another account.

3.1 SWOT ANALYSIS

A SWOT analysis is a structured method showing strengths, weaknesses, opportunities and threats of out project [6].

Strengths	**Weaknesses**
• Cheap price of student work • Home office work-lower expenses of company	• Weak student experiences • Poor GUI • Bad implementation of Cash Bob web application
Opportunities	**Threats**
• Better product than competing companies • Offers better solution • Offers cheaper product (according to student work)	• Competition • Accuracy of data (security)

Table 1: Problems and Opportunities

3.2 GOALS OF THE PROJECT

- Simple application to manage orders, restaurant tables and client accounts
- Offer a fast way to accomplish any required task within the application
- Application will communicate with a web application using REST API

3.3 FURPS MODEL

FURPS is a model for classifying software quality attributes (functionality, usability, reliability, performance and supportability) [7] [8] [9].

1. Functionality
   - Use of the app is only available to the logged in user (the waiter)
   - App is compatible with devices with Android systems
   - App will be designed for tablet devices
   - User can manage accounts (create account or close accounts)
   - User can manage orders (create orders and move items between accounts)
   - User can manage ordered items in account (move between accounts or make a payment on them)
   - User can print a receipt (server printer or mobile printer)
2. Usability
   - Understandable GUI for all users
   - Proper documentation of how to use the app
   - Code will be commented on for extensibility by other students
3. Reliability
   - Unit tests would show proper functionality of classes and functions
   - User tests for testing GUI
4. Performance
   - An App will have fast response time
   - Communication with server will not have an effect on app availability
   - All executions are done on the server
5. Supportability
   - The app will be offered with manual for user
   - The app will have the ability to gain future extensions and updates

4 PROJECT PLAN

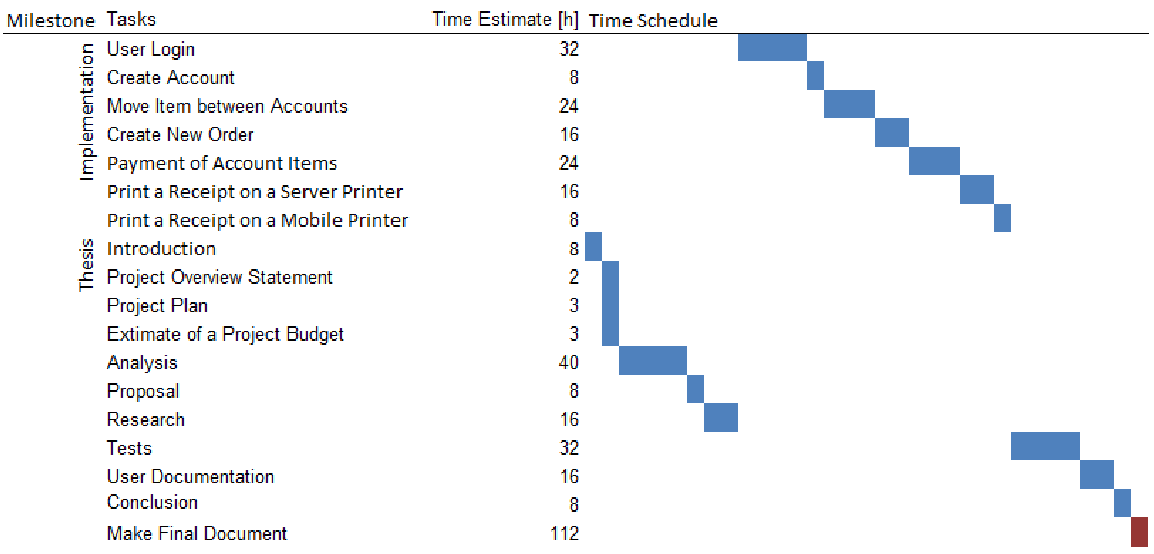
4.1 WORK BREAKDOWN STRUCTURE

- Implementation
    - User Login
    - Manage Accounts
        - Create Account
        - Move Items between Accounts
        - Create New Order
        - Close Account
            - Make Payment
            - Print a Receipt on a Server Printer
            - Print a Receipt on a Mobile Printer
- Thesis
    - Introduction
    - Project Overview Statement
    - Project Plan
    - Analysis
    - Proposal
    - Research
    - Tests
    - User Documentation
    - Conclusion
    - Make Final Document
        - Correction
        - Graphic Style and Structure
        - Print a thesis

Table 2: Work Breakdown Structure

4.2 TIME TABLE AND TIME ESTIMATE

Only one person realized this project, so he had to accomplish all tasks. However, if a problem appeared and extended the time taken to accomplish the task, it will affect to the whole project and extend the time for the delivery of the final product.



Milestone	Tasks	Time Estimate [h]	Time Schedule
Implementation	User Login	32	🟦
	Create Account	8	🟦
	Move Item between Accounts	24	🟦
	Create New Order	16	🟦
	Payment of Account Items	24	🟦
	Print a Receipt on a Server Printer	16	🟦
	Print a Receipt on a Mobile Printer	8	🟦
Thesis	Introduction	8	🟦
	Project Overview Statement	2	🟦
	Project Plan	3	🟦
	Extimate of a Project Budget	3	🟦
	Analysis	40	🟦
	Proposal	8	🟦
	Research	16	🟦
	Tests	32	🟦
	User Documentation	16	🟦
	Conclusion	8	🟦
	Make Final Document	112	🟥

Table 3: Time Table

Note: Task called Make Final Document has a time estimate of 14 days (one week correction and styling and one week printing Result document), but in the table it is displayed as one working day.

4.3 RISK PLANNING

It is necessary to make an estimation of risks, which if they appear, may affect our project. Our task is to identify, analyze, plan and monitor risks [10].

Rest API does not communicate properly

- Probability: moderate
- Effect: serious
- Solution: Rest API will be tested properly on a server side.

Developer does not have enough knowledge

- Probability: moderate
- Effect: tolerable
- Solution: Developer will properly study each requirement/task and research to find the best solution. Developer will find another developer with more experience to teach him what he needs to know.

Team member is unavailable

- Probability: very low
- Effect: catastrophic
- Solution: Developer will make a timetable for activities of the project with enough time to solve any unexpected problem.

Customer change requirements

- Probability: low
- Effect: serious
- Solution: Developer will discuss with the customer every part of the app before developing to avoid serious change.

The time required to develop the software is underestimated

- Probability: moderate
- Effect: serious
- Solution: Developer will make time schedule to solve every problem/task/requirement on time and monitor work efficiency to make estimate more accurate.

The rate of defect repair is underestimated

- Probability: moderate
- Effect: serious
- Solution: Each activity will be prepared before implementation or tested to minimalize defect repair. Project time schedule will include the time it takes to problem solve.

Customer will not communicate

- Probability: low
- Effect: catastrophic
- Solution: All requirements will be written before the project starts in order to minimalize effect of when a customer is unavailable.

Correction of thesis will take more time

- Probability: moderate
- Effect: tolerable
- Solution: Split thesis into chapters and make corrections of each chapter after completion.

Printing of thesis will take more time

- Probability: very low
- Effect: tolerable
- Solution: Research options for printing shops, find out what they offer and order document desks in advance.

5 ESTIMATE OF A PROJECT BUDGET

5.1 ESTIMATE FROM WBS

According to a project timetable, we estimate we will spend 376 hours to finish a project. Our salary is 150 CZK per hour, so the project cost of salary will be 56 400CZK.

5.2 ESTIMATE FROM FUNCTIONAL POINTS

Functional points is a unit of measurement for software projects to express the amount of business functionality a product provides to its user [11]. We divide the system into 5 components and each component is classified according to its complexity multiplied by the amount of things that have an effect on it.

Component	Value
External Inputs	12
External Outputs	15
External Inquiries	5
External Interface Files	20
Internal Logical Files	7

Table 4: Estimate from Functional Points

Sum of these values is 59 function points.

One function point is similar to 8 working hours, so our estimation is 472 working hours. Our salary is 150 CZK per hour, so project cost of salary will be 70 800CZK.

5.3 ESTIMATE FROM COCOMO

Constructive Cost Model calculate the cost of the project according to past, current and future project characteristics.

We used this program to count COCOMO on the website http://csse.usc.edu/tools/COCOMOII.php

Software Size [SLOC]	New	3000
	Reused	0
	Modified	0
Software Scale Drivers	Precedendness	Low
	Development Flexibility	Low
	Architecture / Risk Resolution	Low
	Team Cohesion	Very High
	Process Maturity	Low
Software Cost Drivers		
➢ Product	Required Software Reliability	Very High
	Data Base Size	Nominal
	Product Complexity	Low
	Developed for Reusability	High
	Documentation Match to Lifecycle Needs	Nominal
➢ Personnel	Analyst Capability	Low
	Programmer Capability	Low
	Personnel Continuity	Low
	Application Experience	Low
	Platform Experience	Low
	Language and Toolset Experience	Low
➢ Platform	Time Constraint	Very High
	Storage Constraint	Nominal
	Platform Volatility	Low
➢ Project	Use of Software Tools	High
	Multisite Development	High
	Required Development Schedule	Nominal

Table 5: COCOMO

Cost per person-month: 24 000 CZK

Effort: 26 person-months

Schedule= 10.8 months

Cost: 632 257 CZK

6 ANALYSIS

6.1 ANDROID

6.1.1 ANDROID VERSIONS

The initial release of this operating system was in 2008 and since then many updated versions have been released. Each version brought new features and new possibilities for developers. Android developers have to be careful about considering which version of Android their application will work with.

Our purpose is to develop an application that will work on the largest amount of devices. However, it also has to accomplish all requirements and the graphic interface will be implemented using the newest features. According to the number of devices running a given version of Android, we would like to develop an application for at least version 4.1.x (Jelly Bean), because Jelly Bean is still used by 29% of devices [12]. However, one requirement is to be able to print receipts on a printer and this feature is available since version 4.4 (KitKat), so our application will be available for just 63.7% of current devices.

Codename	Version	API	Distribution
Froyo	2.2	8	0.2%
Gingerbread	2.3.3 -2.3.7	10	3.8%
Ice Cream Sandwich	4.0.3 -4.0.4	15	3.3%
Jelly Bean	4.1.x	16	11.0%
	4.2.x	17	13.9%
	4.3	18	4.1%
KitKat	4.4	19	37.8%
Lollipop	5	21	15.5%
	5.1	22	10.1%
Marshmallow	6	23	0.3%

Table 6: Number of devices running a given Android version

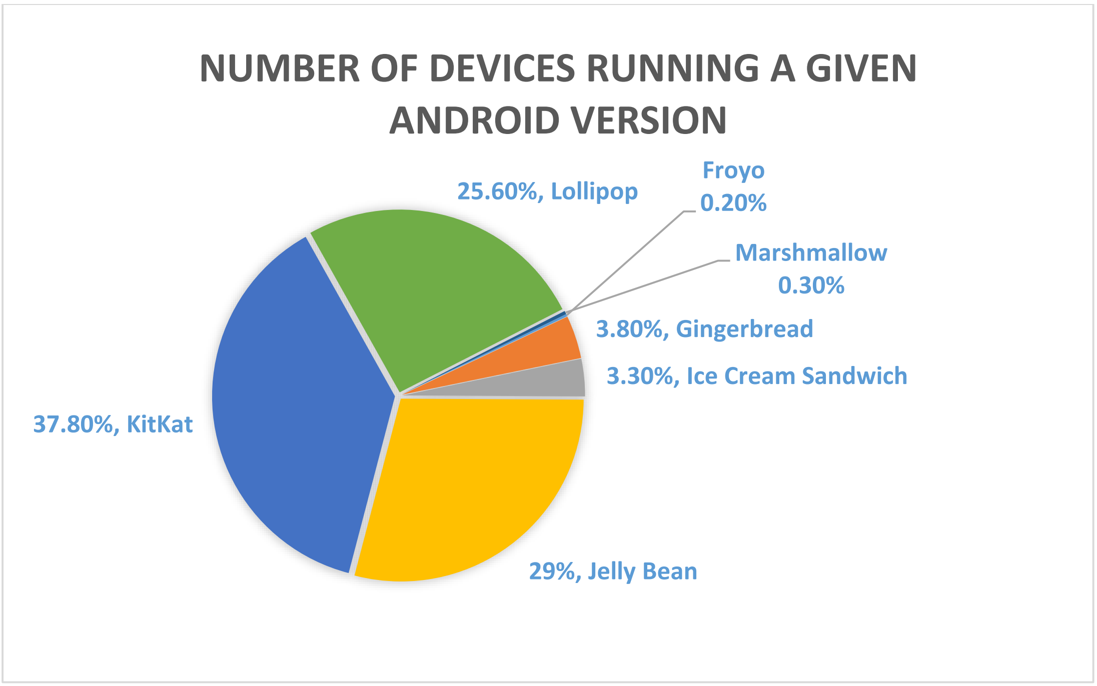


Figure 1: Number of devices running a given Android version

6.1.2 ANDROID STUDIO

For our project, we will use Android Studio as the official IDE for Android Application development [13]. Android studio offers a good environment with logical structuring. GUI of applications can be written as a XML file, by using graphic interface. The application would be running over an Android emulator or tested on a device with Android connected to our computer. Android Studio offers many features and makes the development of applications more intuitive.

6.2 FUNCTIONAL REQUIREMENTS

Functional requirements show the requirements of our application functionality [14].

Functional Requirement	**Description**
We can create account with any name, connected to a table or without any information- quick account	According to user needs, he can make any type of account
Ability to move items between accounts	From one account it is possible to move items to other existing accounts
For every account we can make orders	For one account we can create an unlimited amount of orders
Receipts can be printed on server printer	User can select printing on server printer for the receipt when the customer is making a payment
Receipts can be printed on mobile printer	User can select printing on mobile printer for the receipt when the customer is making a payment

Table 7: Functional Requirements

6.3 NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirement	**Description**
Only user with role WAITER can log in to the app	Application can be used only by waiters, so it is necessary to check the role of every user before log in
App will hold data if application process stops or is interrupted	If the app is stopped or interrupted, it will back-up data, which is not yet in the database of activity
App will keep non-closed orders if user interrupts order by another activity	If the app is closed , it will back-up data which is not in the database of activity yet
Available to update	App will be constructed to update and extend its use by new functionality
Platform	App will be available for Android system devices
Compatibility	App will be available on as many Android system versions as is possible

Table 8: Non-Functional Requirements

6.4 USE CASES

Use cases is a collection of scenarios of how the system is used by users. Only user with role WAITER can use the application. A user can:

- Create account
- Move items between accounts
- Create orders
- Make payments (whole or part of the account)
- Print a receipt on a printer (server or mobile printer)

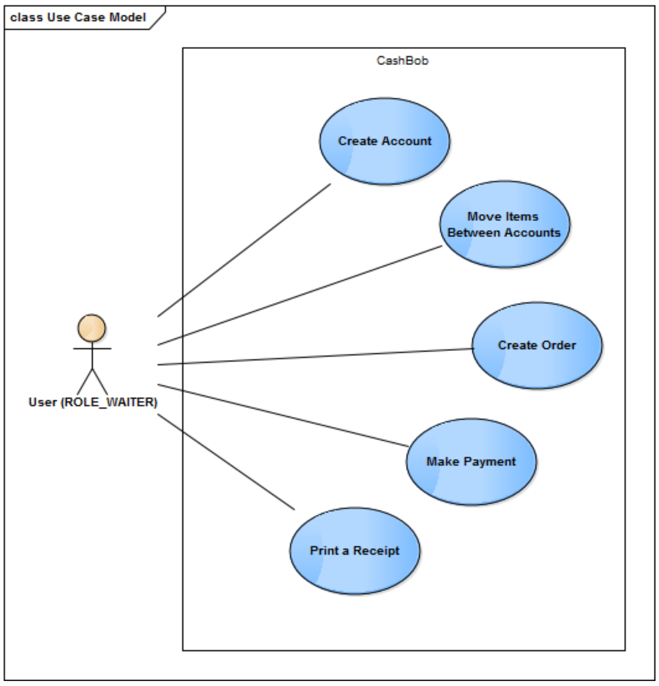


Figure 2: Use Case Model

6.5 USER ACTIVITY DETAILS

Create account

- Click to Add Account
- Fill in a name (not required)
- Choose linked table (not required)
- Click to Create

Make an order

- Choose an account
- Choose items by clicking on them
- Click to Order

Make a payment account

- Choose an account
- Choose items for payment
- Click to Pay

<u>Move items between one account to another</u>

- Choose an account
- Choose items to move
- Choose target account
- Click to Move

<u>Move items from one account to a new account</u>

- Choose an account
- Choose items to move
- Choose New Account (use case Create an account)
- Click to Move

7 DESIGN

7.1 USER INTERFACE DESIGN

Our target is offer a GUI that will be easy to understand and offer a fast way to achieve any requirement by user.

7.1.1 REQUIREMENTS TO GUI

Requirement	Description
Understandable GUI	
Minimal number of layouts	If it is possible, we would like to minimize the number of layouts. Then we can make the application more understandable for a non-tablet user. Minimization has to make sense; otherwise, application will be too complicated.
Similar tasks will have similar layout throughout	To achieve this, we will use similar placement of icons so user will be more sure of where to click when they are doing similar activities
Possibility to go back anytime	User can make a mistake and it will be easy to return/cancel any of their activity
All text is readable	

Table 9: Requirements to GUI

Our aim is to copy CashBob desktop version and make it compatible with an Android tablet. The desktop version offers a menu to go through an application.

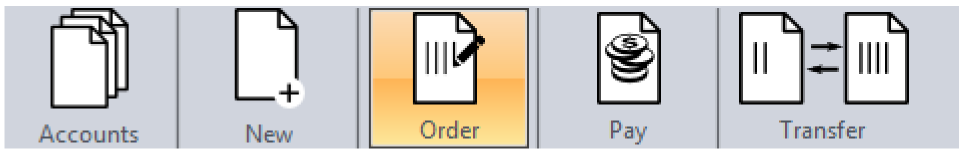


Figure 3: Menu from Desktop Version

- Accounts: Show list of accounts
- New: Create new account
- Order: Make an order for current account
- Pay: Make payment
- Transfer: Transfer items from one account to another

We will copy this menu and put it on every screen.

7.1.2 LOGIN SCREEN

Only waiters of a restaurant will use our app. To make a login for the waiter we need:

- Login name of waiter: It is necessary to distinguish who is using the app, especially if the user is a user with role ROLE_WAITER
- Password: Access to our system has to be secure
- Server IP: CashBob database have to run on a server, so it is necessary to identify the server by IP

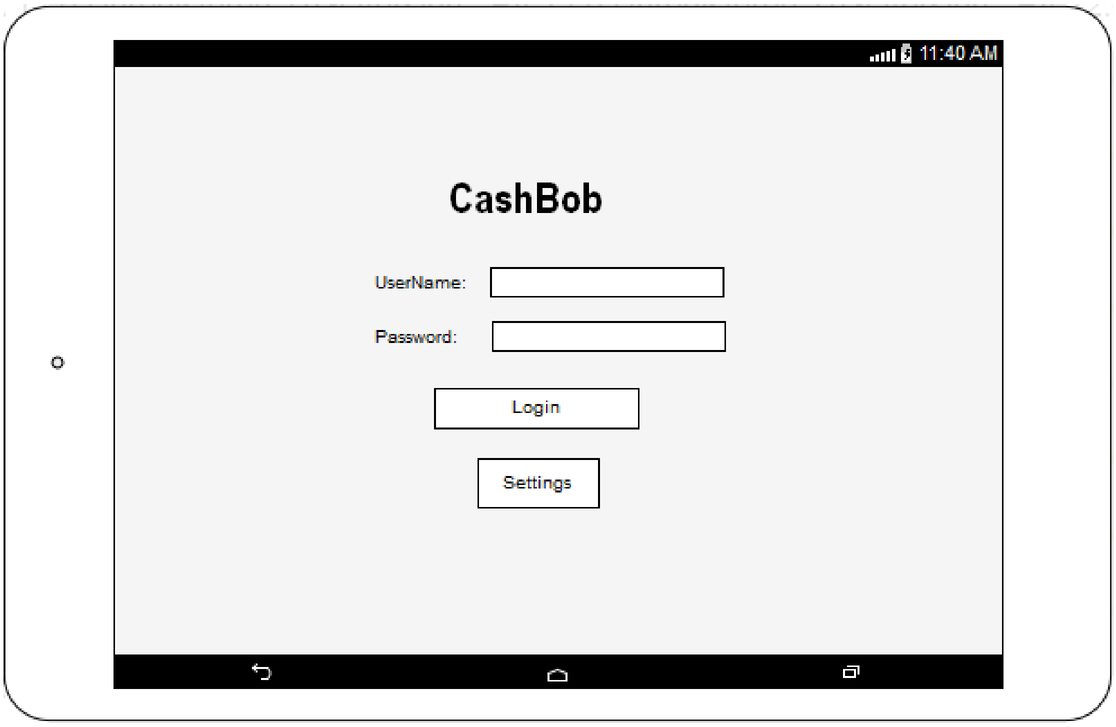


Figure 4: Login Screen

On this screen user could fill in their user name and password. To fill an IP address of server or choose a printing server he has to click the “Setting” button.

7.1.3 ACCOUNTS SCREEN

First tab is to show a list of all accounts. This tab is also the first window that will be displayed after login. User can choose an account by clicking on it.

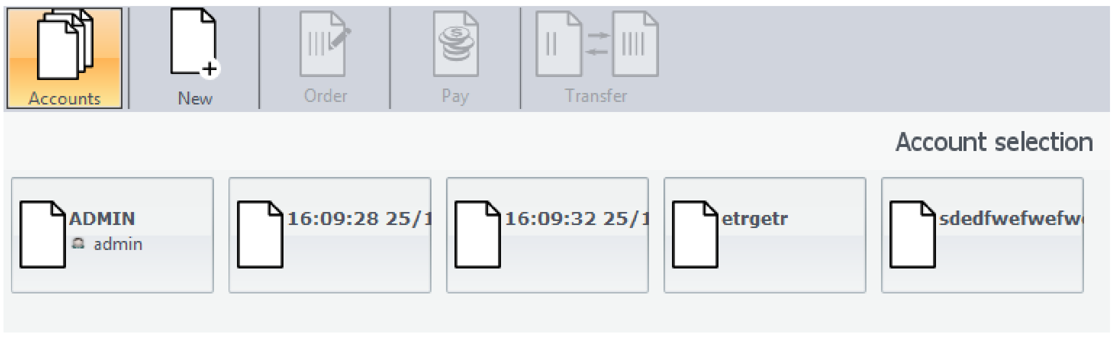


Figure 5: Desktop Accounts Screen

We will display the accounts as a scroll list, where you can click on the account you want to work with. After choosing an account, activities “Order”, “Pay”, and “Transfer Items” will be available to work with.

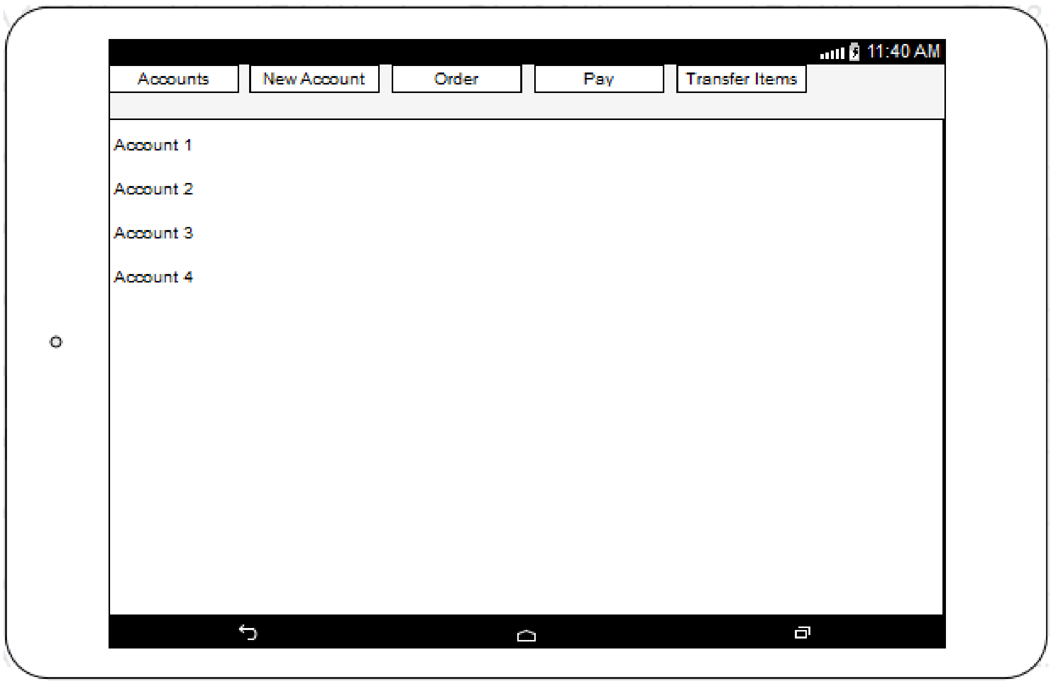


Figure 6: Accounts Screen

7.1.4 NEW ACCOUNT SCREEN

User will create a name for the new account and select a table for it (it is also possible not to choose a table). If user declines to create a name or choose a table, a quick account will be created (name of account will be the time when the account was created).

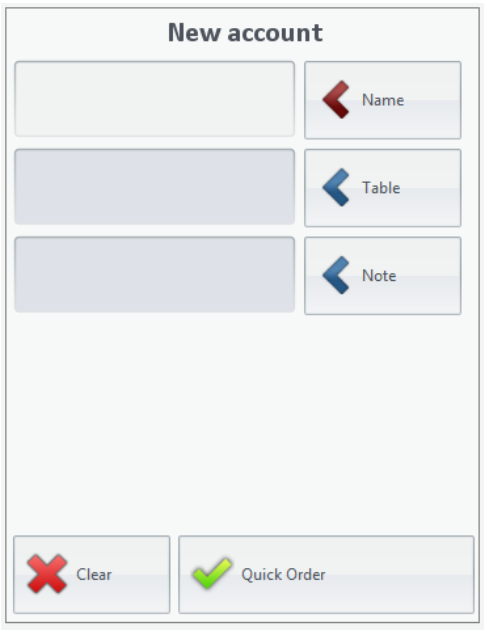
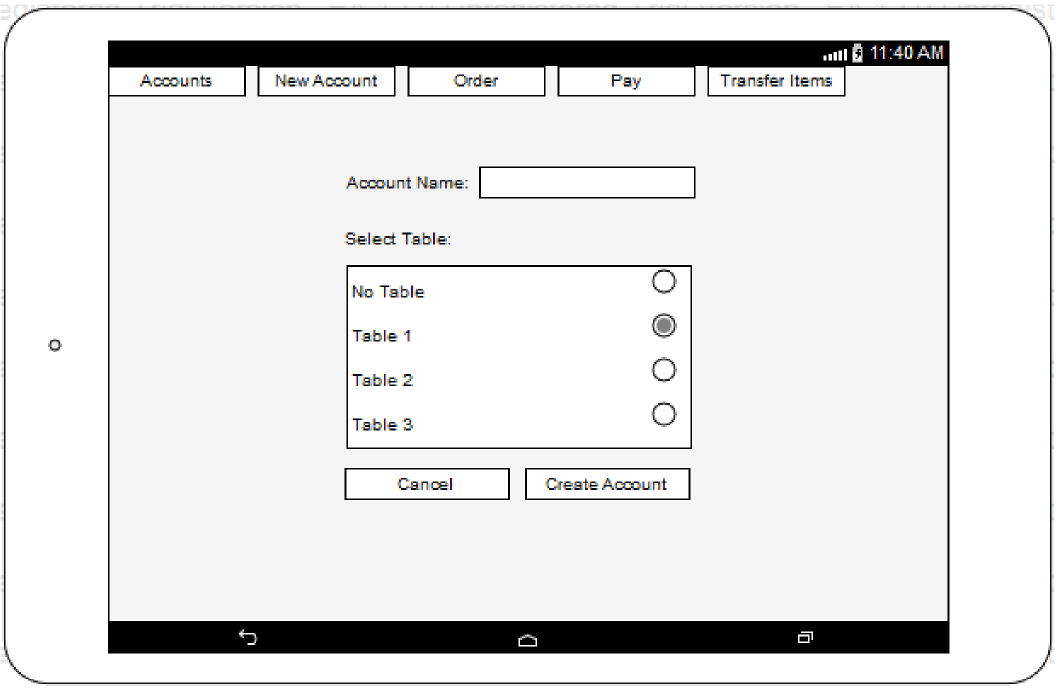


Figure 7: Desktop New Account

In a discussion with stakeholders, we agreed to removing filing note for an account because it does not display anywhere within the application, so it is useless to keep.



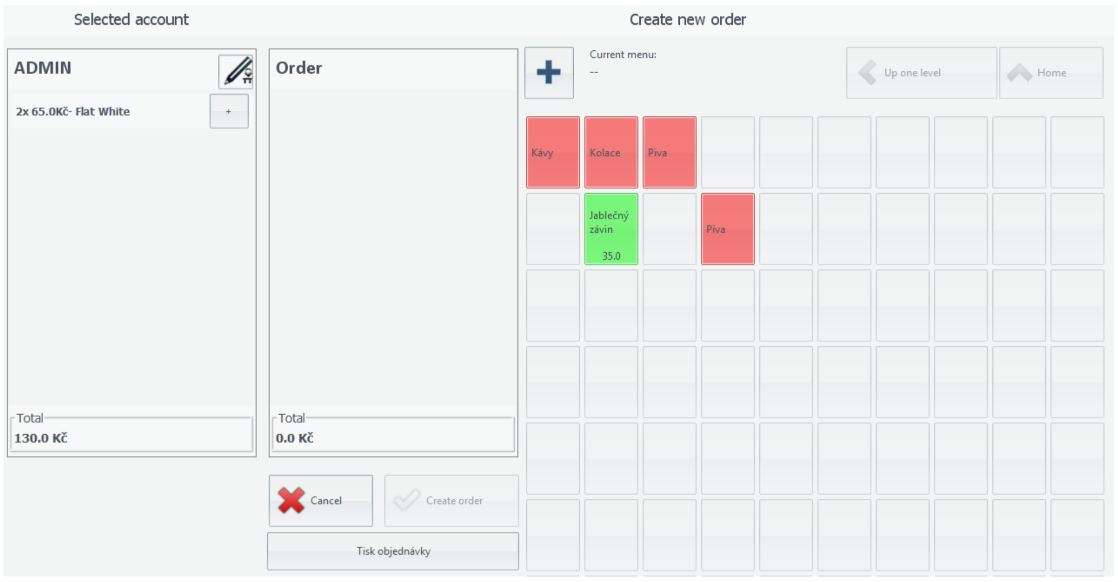
Account Name:

Select Table:

No Table ○
Table 1 ●
Table 2 ○
Table 3 ○

Figure 8: New Account Screen

7.1.5 ORDER SCREEN



ADMIN

2x 65.0Kč- Flat White

Total
130.0 Kč

Total
0.0 Kč

Figure 9: Desktop Order Screen

Desktop version is divide to three columns. The first shows a list of items in the currently selected account. The second is a list where the current order (unfinished, to finish it user has to click to “Create Order” button) is shown. The third is the menu card.

In Android version, the first column will be a selectable list. If one item is clicked, the item will be also added to order list. Under it is a list of information about price of account items.

The second column (order list) will also be designed as selectable list, but after clicking on any item, the item will be removed from that list. Under the order list and information about price are three buttons “Cancel”, “Create Order” and “Print Order”.

We decided to remove the last button “Print Order”. We think that printing a receipt should be available only after making payment. If a user would like to print the receipt, he should create a quick account, add items to it and make the payment. Another reason is that this list of items to order before confirmation is not saved in a database, only in the device. So, after clicking this button it would create a quick account, move the items from the order to the order of the quick account, delete items from the shown list and make the receipt available to print. We think it would be confusing for anyone who will be working on this application in a future. This button was connected to a currently selected account, which would appear confusing for the user. Application will offer the option of printing the receipt only under the tab “order”.

The third column is a menu card. In the desktop version, it is designed as a button area. However, we are trying to avoid this because we think it would complicate it. The design of the structure of buttons would make it unreadable on a device. In our design, it is shown as an expandable list and it would look like a paper menu card offered to a restaurant customer. It is possible that this part would change according to our development.

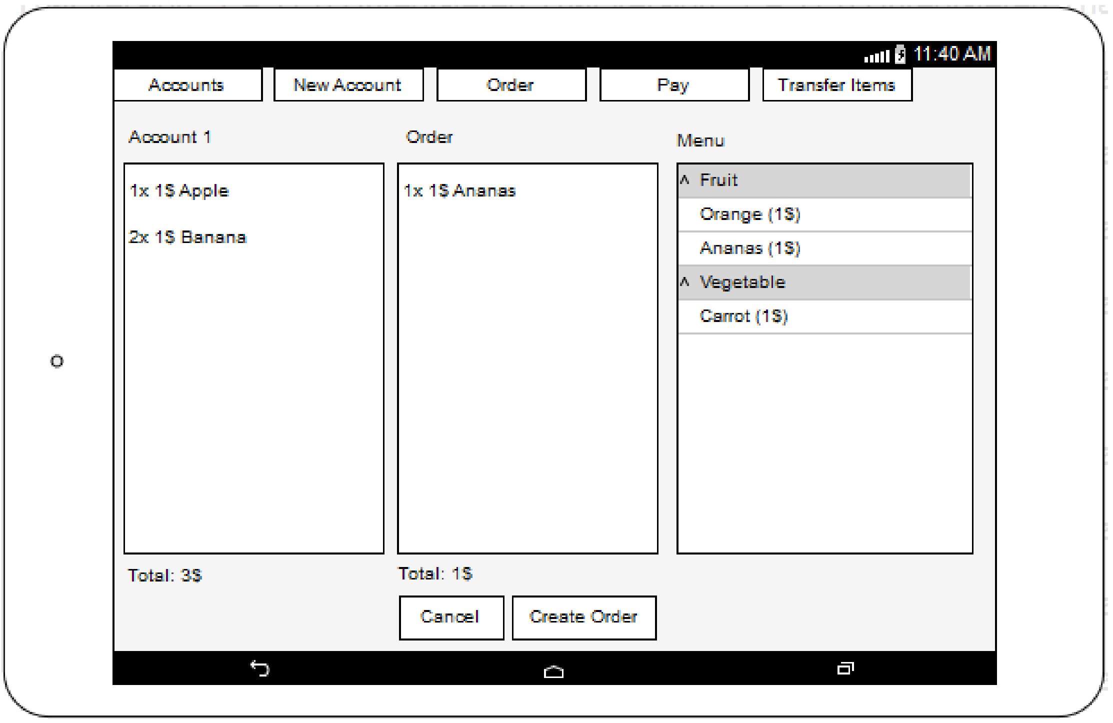


Figure 10: Order Screen

7.1.6 PAYMENT SCREEN

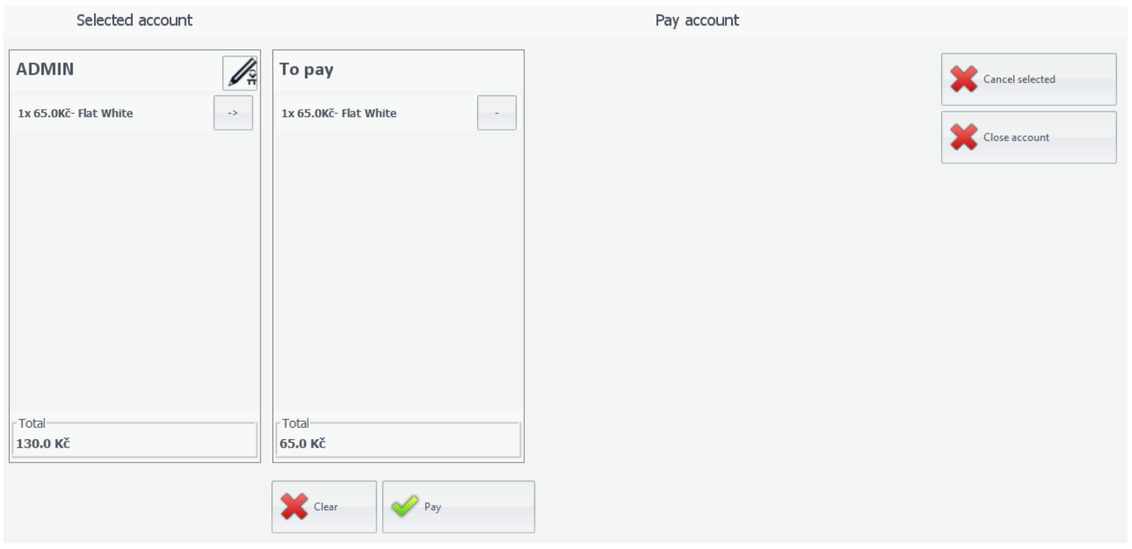


Figure 11: Desktop Payment Screen

Design of payment screen stays the same as on the desktop version.

- Click an item in account list- add to list to pay
- Click an item in to pay list- remove from list to pay
- Click “Cancel Selected”- delete item from account item list
- Click “Cancel Account”- delete account

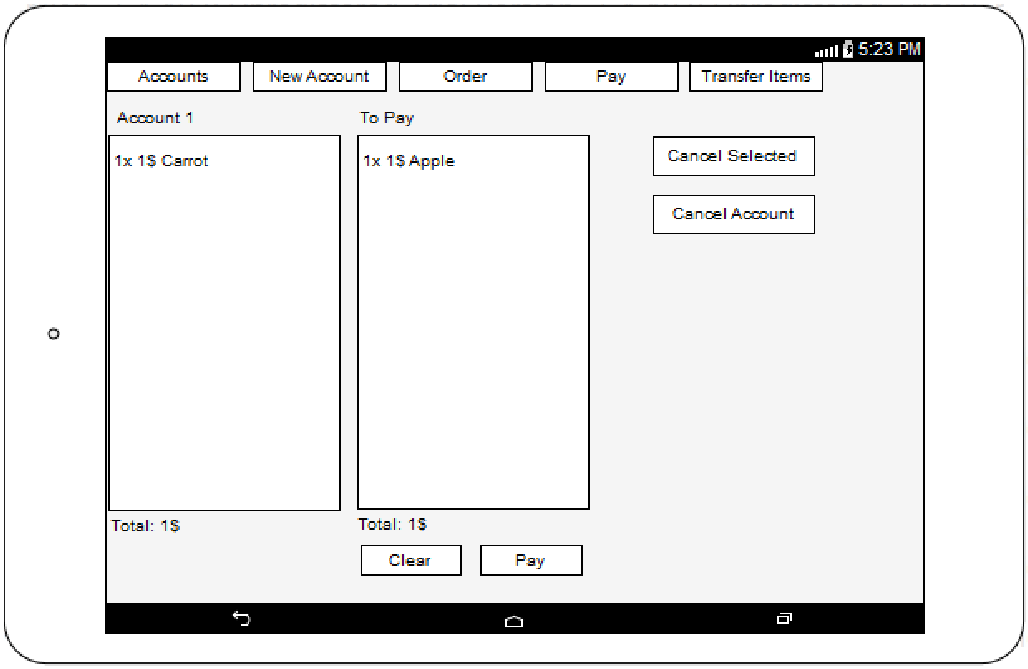


Figure 12: Payment Screen

7.1.7 TRANSFER ITEMS SCREEN

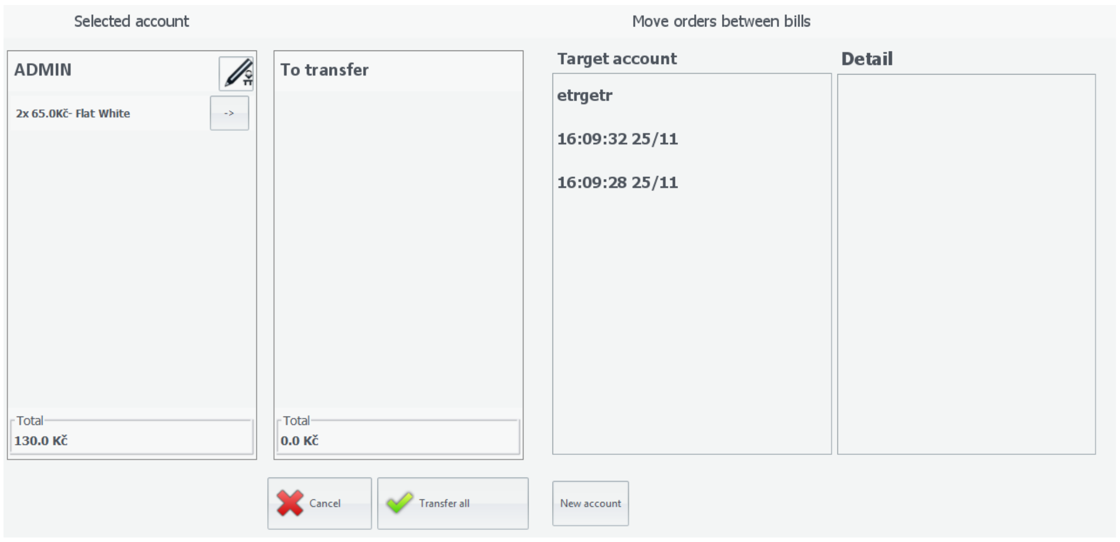


Figure 13: Desktop Transfer Screen

The transfer item screen is divided to three parts:

- List of items from account currently selected (click on item to move item to transfer list)
- List of items to transfer + confirm button + cancel button
- List of target account where items would be transferred with detail (account item list) + button to create quick account

The problem is that the last column has two lists, which means we would have four lists next each other. Tablet devices are devices with screen from 7” and that means there is a possibility it would make the screen not readable (too low width of each list to see content properly). To solve this problem we would:

- Remove list of accounts and exchange it to spinner (only selected name of account will be shown and by clicking on it, the list of accounts will appear). In that case, it would take longer to find target account if there are too many open accounts.
- Do not show details of target account. In that case, a user will not see which items are on target account.

We decided to choose second solution (showed as figure).

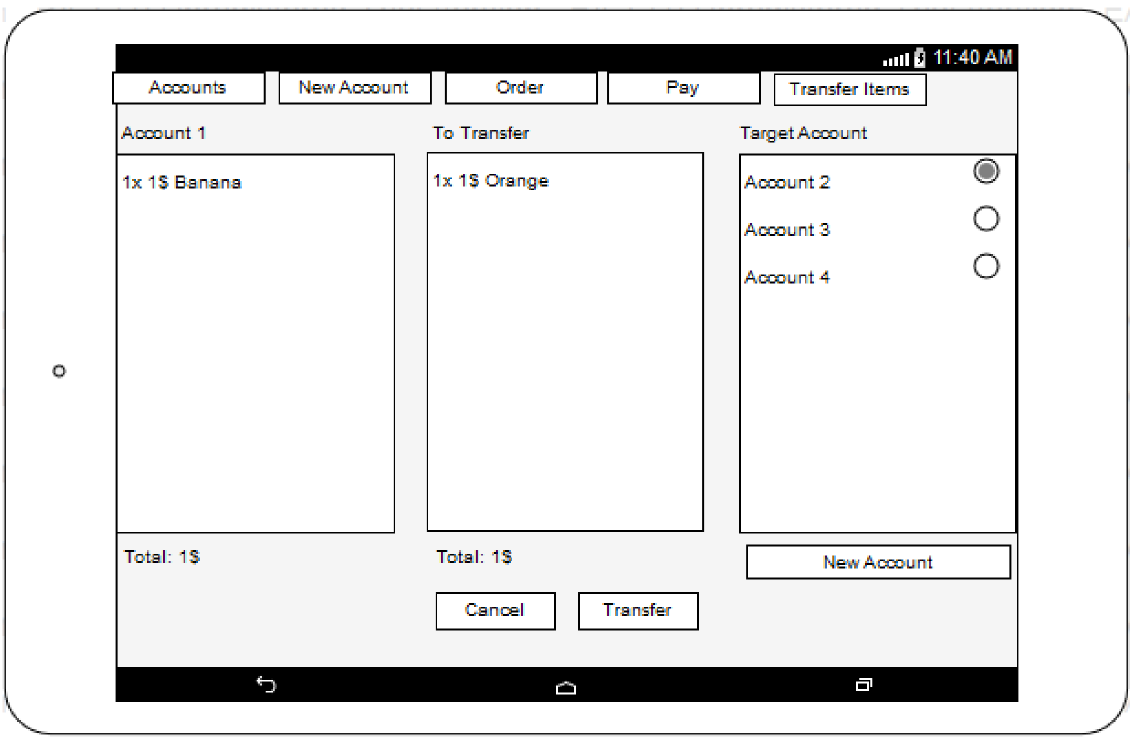


Figure 14: Transfer Screen

7.1.8 DISCUSSION ABOUT DESIGN WITH STAKEHOLDER

7.1.8.1 PREDESIGN MEETING

- We suggested deleting the attribute “note” in entity “account”, because after creating an account “note” is not shown anywhere, so it is useless. The stakeholder agreed.

7.1.8.2 FIRST DESIGN MEETING

- Login Screen: We placed a button for Setting before Login button. Stakeholder wanted the button for Login before the button for Settings and make the Settings button smaller.
- Tab Accounts: We designed one list of accounts over the whole screen, but stakeholder suggested we create a list account in that screen to make account more accessible to click. This stayed to a discussion on final application presentation.
- Tab Order: Stakeholder was not sure about expandable list for a menu card- it would not be accessible as a menu card in desktop version. We agreed to look for a better solution.
- Stakeholder was interested in adding an icon to each tab to make it more understandable. We agreed to add it if possible.
- Tab Order: After discussion with stakeholder about two possibilities of showing target account in this screen, stakeholder agreed that we would show only the list of the account, not the items.
- Stakeholder agreed with removing “print receipt” button.

7.1.8.3 SECOND DESIGN MEETING

- Tab New Account: Stakeholder suggested that if user chooses a table for an account and an account name is not created, it would automatically be named according to the table name.
- Tab Pay: Stakeholder decided to remove buttons “Delete Account” and “Delete items”. This feature will be available only on web application. The account will be deleted automatically if an account item list is empty after making a payment.
- Tab Order: We presented a new design for the menu card. Stakeholder agreed with the solution. New design:

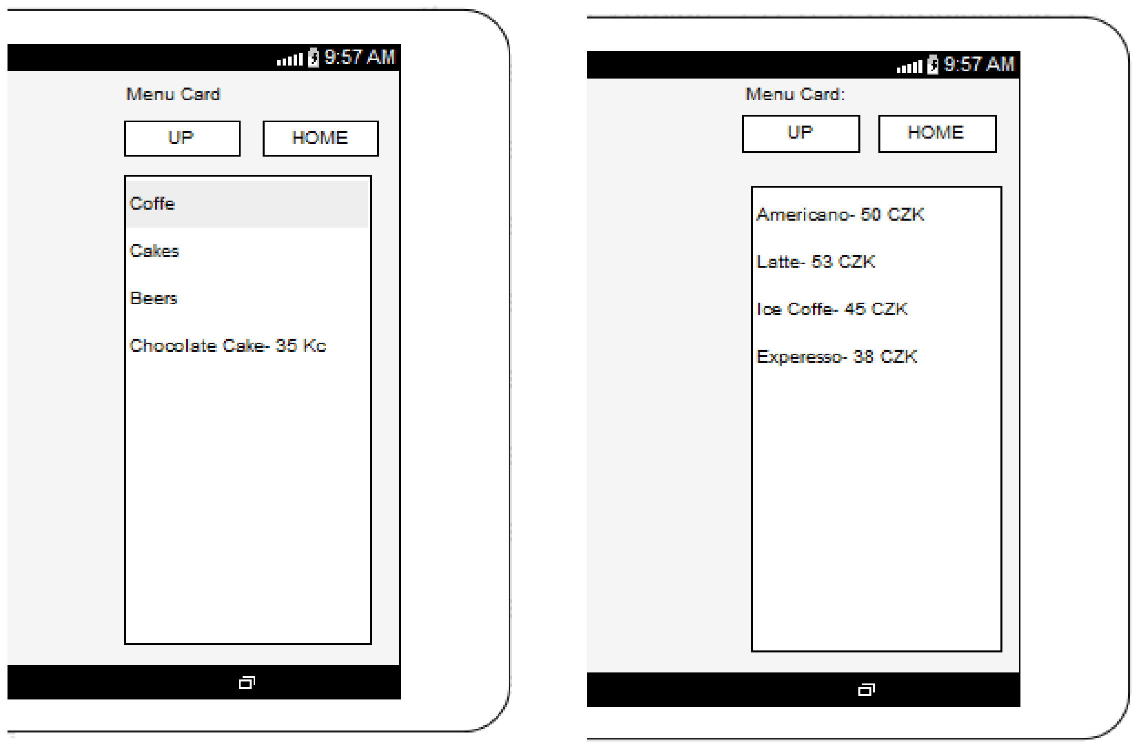


Figure 15: New Menu Card

The menu card contains two buttons and simple list. If the user clicks on a menu item and its category, the category items will show in this list. By clicking on a product, the item will be added to an order list. Button "Up" will show previous category and button "Home" shows first level list of menu card.

- Stakeholder decided to remove the ability to print receipts from a mobile application requirement. It will be developed only if developer has enough time to implement it.

7.1.8.4 THIRD DESIGN MEETING

- Tab Transfer: Button to create a new account was implemented to create quick account. Stakeholder would like to create an account with a given name.
- Stakeholder wanted all prices with currencies be showed with a blank space between them.
- Tab New Account: Stakeholder would like buttons be moved to the right side of the field for a new account name.

7.2 DATA DESIGN

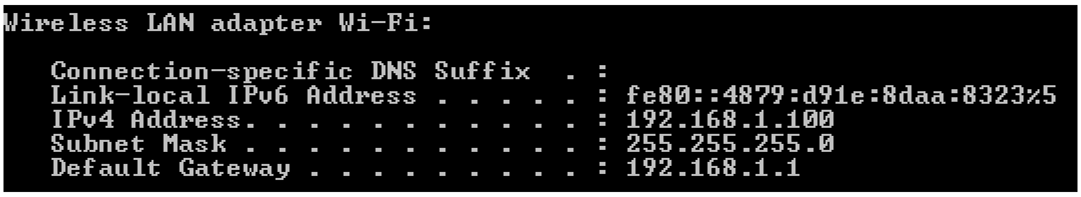
Our application is developed as a client that will communicate with a server over REST API. The app would be used together with a web application, so it is necessary to work with most recent data. We will download the new data every time the user wants to switch an activity (tab). This means we will not save any data in our app at any time and what is not posted on the database will be lost. In that case, communication with REST API is critical and since we do not have connection, we cannot use our application properly.

8 IMPLEMENTATION

8.1 COMMUNICATION WITH A SERVER

We would like to test our application against a server provided by students developing CashBob as a web application. This server is running on our computer and it is necessary to access to it remotely from our Android system device.

To connect to our computer server (localhost: 9000) we need to get the IP address of our PC connected to the same wireless network (server and client have to be in same wireless network). In Windows, we can get this information over Command Prompt by command "ipconfig"



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::4879:d91e:8daa:8323%5
   IPv4 Address. . . . . . . . . . . : 192.168.1.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

Figure 16: Command Prompt

In our case IP address is 192.168.1.100, so connection to our server is over link http://192.168.1.100:9000

So, for example, if we would like to communicate with a server and we want a list of tables, we will use link http://192.168.1.100:9000/rest/table

We wanted to test against the server that will be used in the future, but many mistakes appeared over that time on the server side:

- Connect to a link was not available (for example http://192.168.1.100:9000/rest/table)
- We cannot get to a server from another device (from device where there is not a server)
- Method POST did not work
- Get a menu card did not work
- Proper documentation is not available (problem in communication with the server)
- Server does not show ordered items of an account
- Tree structure of menu card was implemented with mistakes (wrong pointers between items)

After all the problems with the new server REST API we decided change the server back to the old server provide by CashBob desktop version. This version works only with older version of Java, so we decided run it in our virtual machine using VirtualBox [15].

For that virtual machine we had to set two network adapters in settings:

- NAT (Adapter Type: Intel PRO/1000 MT Desktop 82540EM, check Cable Connected)
- Bridged Adapter (Name: Intel(R) WiFi Link 5100 AGN, Adapter Type: Intel PRO/1000 MT Desktop 82540EM, Promiscuous Mode: Allow All, check Cable Connected)

Settings may be different in other machines. In a virtual OS, we have to find the IP address of adapter like before and a port where a server is listening. Connection to a server will also be available for all clients connected in a same network.

8.2 REST API INTERFACE

Our application communicates with a server over REST API over methods GET, POST and PUT. Latest version of REST API provided by desktop application is being redesign and available by web application. However, it is tested against desktop application servers. In the table below are all queries used by our application.

URL path	Method	What we used
user/{name}	GET	roles: [{name}]
account	GET	accs: [{opened, id, name}]
table	GET	tables: [{id, tablenumber}]
account	POST	name,table: {id}
account/{id}	GET	order: [{ispaid, iscanceled, menuItem: [{menuitemid, name, price}]
account/{id}	PUT	id, name, opened
cell	GET	cells: [{parentMenu: {menuId, name}, itemId, name, price, isMenu}]
account/{id}/order	POST	items: [{manuItemId, count}]
account/{id}/payItems	PUT	items: [{manuItemId, count}]
account/{id}/moveItems	PUT	targetAccId, items: [{manuItemId, count}]

Table 10: REST requirement table

9 TESTING

9.1 UNIT TESTING

It is necessary to test our application to ensure that everything is implemented properly. We created tests to check the proper functionality of implemented method. We wrote and ran the test in the application Android Studio [13], which we also used as an environment to develop our application.

To be sure that all our implemented features work properly, during the time of development, we tested each new feature in all possible ways that a user could use it. During that testing, if we found any failure, we immediately solved it before developing a new feature. This testing took a lot of time, but gave us valuable information about developing features.

9.2 ACCEPTING TEST

Acceptance testing shows stakeholder that all given requirements were implemented.

Application would communicate with CashBob server

- Data from server would be shown in the application.
- Result: All the data loaded and was saved to the server.

Only user with a role waiter would have access into application

- Application is meant to be used by waiters and no one else should have access to it.
- Result: User will Login to the application.

<u>User would create new account</u>

- Create name of an account and select a table. If a name were not created, the account name would be selected table. If table is also not selected, the name would be current time and date.
- <u>Result</u>: New account will be created.

<u>User would create new order</u>

- User would order item by selecting item from previous orders or choose item from menu list.
- <u>Result</u>: New order for selected account will be created.

<u>User would make a payment of account items</u>

- User would select account items to be paid. If the account will not have any unpaid item, account will be deleted.
- <u>Result</u>: Paid items will not showed. If account is without items, it will be deleted.

<u>User would move items between</u>

- User would select items to move and the target account. User would create new account and name it.
- <u>Result</u>: Items are moved from previous account to another account.

All criteria were successfully met. All requirements given at the meetings with the stakeholder were accomplished.

9.3 USABILITY TEST

For better quality of our application is also necessary to do a usability test. Usability test is to test the application by user, when the user has tasks and when they accomplish them. User activity is monitored and offers the ability to show application errors.

According to many recommendations, usability testing does not require more than 5 people, because the main problems would be discovered by the first or second person testing the application. The others discover the same problem as the original testers and usually do not discover a new one [16].

Before the test we set the application database to:

- Create 5 accounts
- Create 5 tables
- Create menu card with 14 items in 4 submenus

Any type of person would use our application, so we were not looking for user testers with any specific characteristic (age, sex or android mobile experiences). There were five people to our usability test. Four of them are Android mobile phone user, one iOS mobile user. The tested person was introduced to the idea of CashBob and received a tablet (10.1" tablet, Android version 4.1.1) with installed CashBob application and instructions to accomplish given tasks. Before the test, we told them about activities done by a waiter in a restaurant and about our application without showing them the application yet.

9.3.1 SCENERY

- Log in with a given user name and password
- Customer Tomas came into the restaurant and sat at table two. He would like to order 2x beer Plzen 12 and 3x red wine
- From table one people divided. Move all beers to account called Petr and cakes to new account Martin
- Tomas would order three more beers Plzen 12 and one Gambrinus 10
- Michal would like to make a payment for two apple pies
- Create account for table 5
- Tomas would like to pay for all Plzen 12 and one wine

9.3.2 RESULT

According to out testing we found few faultiness:

Orientation in Menu Card

User has problem get up one level in menu card

Occurrence: high

Source: User does not connect button "Up one level" and "Home" with menu card or Czech text on buttons does not make sense to a user

Solution: It is on a discussion with stakeholder, because it would have effect to other CashBob applications

Create Account under Tab Account

User wanted create new account and looked for that in tab “Accounts”

Occurrence: high

Solution: This problem would be solved by removing tab “New Account” and adding button for new account under tab “Account”. We had requirement keep design of desktop version, so we did not implemented it. This change would be decision of a stakeholder.

Problem Confirm Activity

Users had problem confirm activity by clicking to confirm button.

Occurrence: high

Source: Buttons were too small and hidden down on a screen

Solution: Size of buttons was extended and text size highlighted

Problem Create New Account

Users had problem recognize he could fill account name and select table

Occurrence: high

Source: Activities were described wrongly and too close to each other

Solution: Added more space and added labels

User has problem recognize field to fill

In our Android version user has problem recognize if showed text is normal text or a field to write input.

Occurrence: high

Source: Form of hints in these fields was written wrongly

Solution: Hints in text field were transformed to be more understandable

One Useless Step

User was redirected to a tab “Account” after creating a new account and then he had to select that account to make an order. Different from desktop version.

Solution: Faultiness was removed. After creating account user is redirected to a tab “Order” for this account.

User has problem With Tabs

User had problem recognize tabs and use them to switch to different screen.

Occurrence: low

Source: Tabs had same background color as rest of the screen

Solution: Change background color of tabs

User has problem recognize selected account

User has problem recognize which account is selected and made tasks under different accounts.

Occurrence: low

Source: Account name had same size as other text

Solution: Highlight name of selected account in tabs

Return Activity by Using Android Return Button

User did something he did not want and tried return it by using android return button. It returned him to previous screen.

Occurrence: low

Solution: Android return button was blocked

Our usability test was mostly provided to find faultiness of application CashBob. Founded faultiness were founded and corrected. Few of them did not, because our application GUI was implemented like in the desktop version, and any changes could have affected the GUI of all CashBob apps, so it have to be decided by a stakeholder.

Tester’s recommendations:

- Accounts for tables would be created as default and no need create them everytime
- Creating new account under tab accounts

10 CONCLUSION

The aim of this thesis was to develop an Android application for waiters in restaurants and in our thesis we accomplished this aim. This application offers waiters the ability to manage accounts, orders and perform activities related to it. All accepting criteria and requirements given by meetings with the stakeholder were implemented and properly tested by developer. Because of a time problem, the stakeholder changed the requirement of printing receipts to optional and the option to implement it only if there was time before the deadline. However, a few problems appeared during developing time, so this feature has not been implemented.

The biggest problem I had to solve was communication with server. I tried to test the application against the server provided by a developer, who develops web application servers, but an interface was not implemented properly and it was not possible to test functionality. We started to work with a server provided by the desktop application, but the documentation was written poorly and some parts were missing, so we had to find out how to communicate with server properly.

Application was implemented on tablet devices, so the text size and design was adjustable to devices with bigger screens. On small devices, it would cause problems to control the application. The application is provided in two languages: Czech and English.

We developed an application that meets all requirements and ideas given by stakeholder. Every new implemented functionality was tested to ensure it works properly. We hope our application will satisfy stakeholder and will be implemented and used in real restaurants to satisfy customers.

Developing our application gave us a lot of experience. We learned new methods of how to implement application features, how to find the best solutions and improve our problem solving. Writing this thesis was part of subject Project Control, so, many learned topics were used in this thesis and gave us the opportunity to understand them better.

I would like to thank my supervisor for his time spent helping me with my thesis and advice given during that time. I think I have learned a lot from this, which will be helpful in my future IT career.

10.1 REVIEW WORK BREAKDOWN STRUCTURE

In our work breakdown structure, time spent on thesis milestone was not estimated properly. A problem with estimation appeared in implementation milestone, where the time it took to pass some tasks was expanded because problems needed to be solved, mostly finding the correct way to communicate with REST API and how get a correct response, because documentation of REST API was not written properly and parts were missing. In the end, we did not reach the estimated 376 hours, but we only spent around 300 hours on this project. Estimation depends on our experiences. With more experiences we would reach more precise estimation.

10.2 FINAL APPLICATION DESIGN

10.2.1 LOGIN SCREEN

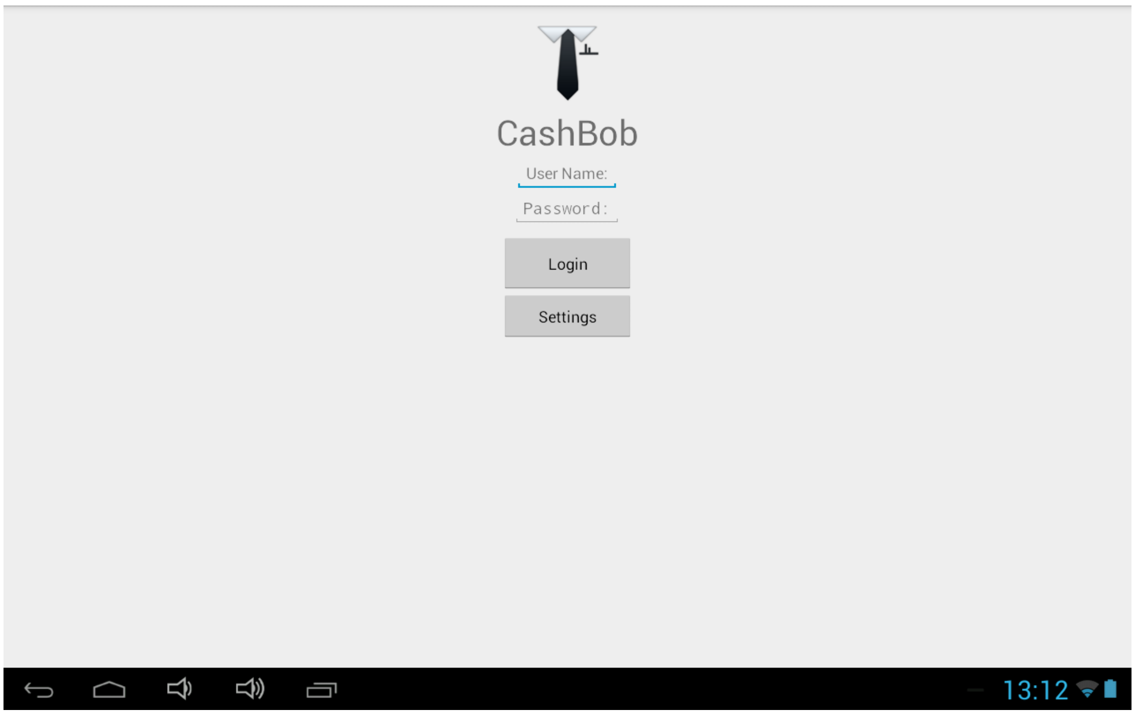
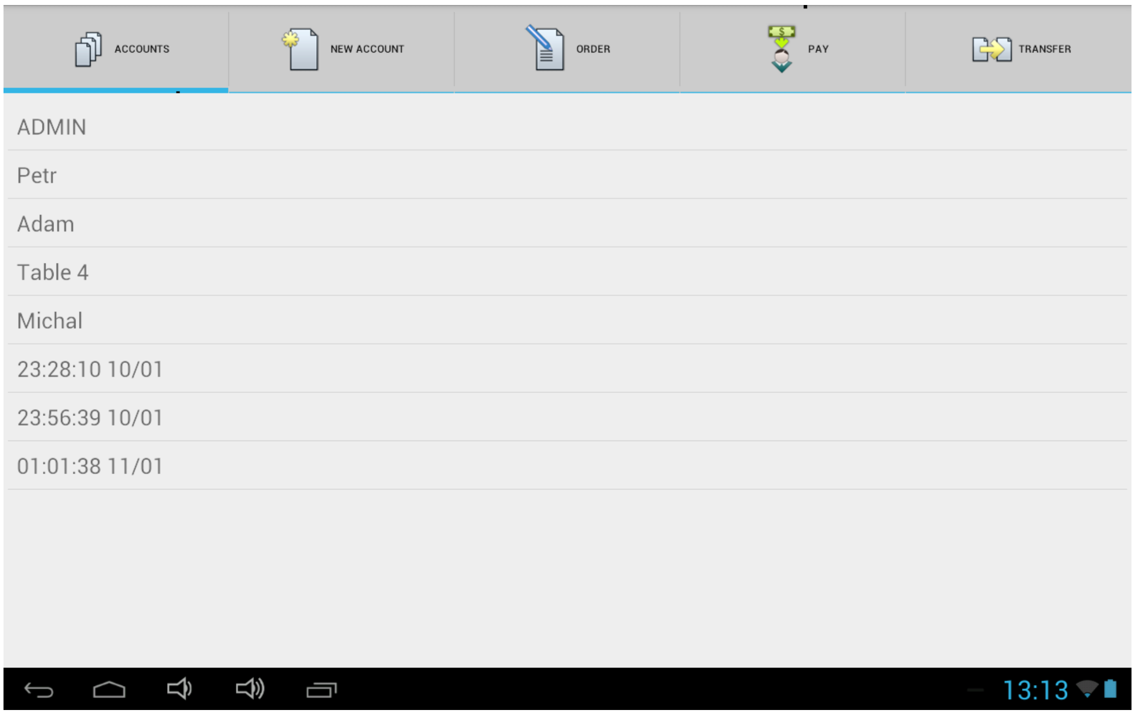


Figure 17: Login Screen

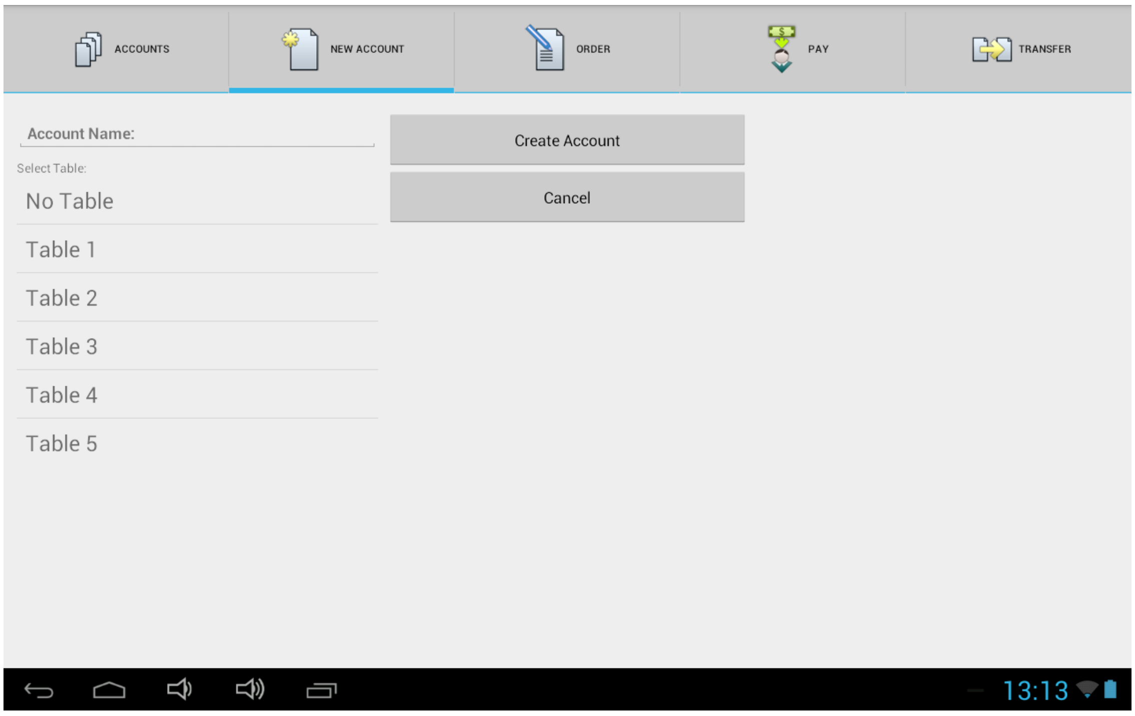
10.2.2 ACCOUNT SCREEN



ADMIN
Petr
Adam
Table 4
Michal
23:28:10 10/01
23:56:39 10/01
01:01:38 11/01

Figure 18: Account Screen

10.2.3 NEW ACCOUNT SCREEN



Account Name:
Select Table:
No Table
Table 1
Table 2
Table 3
Table 4
Table 5

Figure 19: New Account Screen

10.2.4 ORDER SCREEN

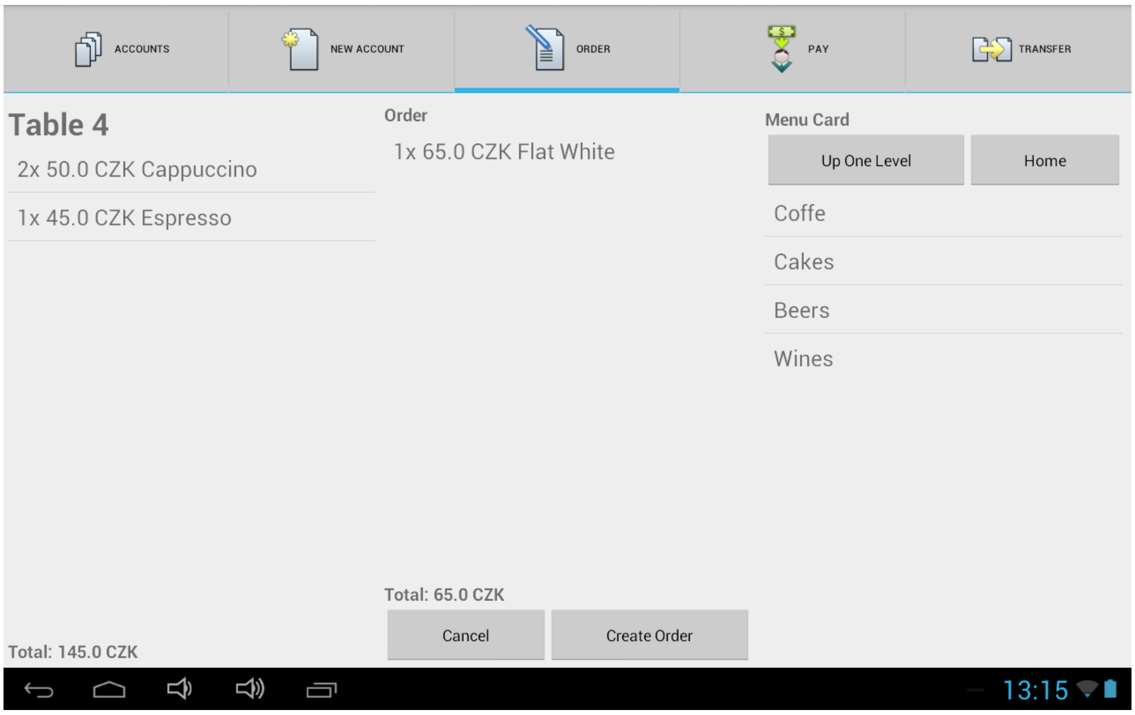


Table 4

2x 50.0 CZK Cappuccino

1x 45.0 CZK Espresso

Total: 145.0 CZK

Order

1x 65.0 CZK Flat White

Total: 65.0 CZK

Menu Card

Coffe

Cakes

Beers

Wines

Figure 20: Order Screen

10.2.5 PAYMENT SCREEN

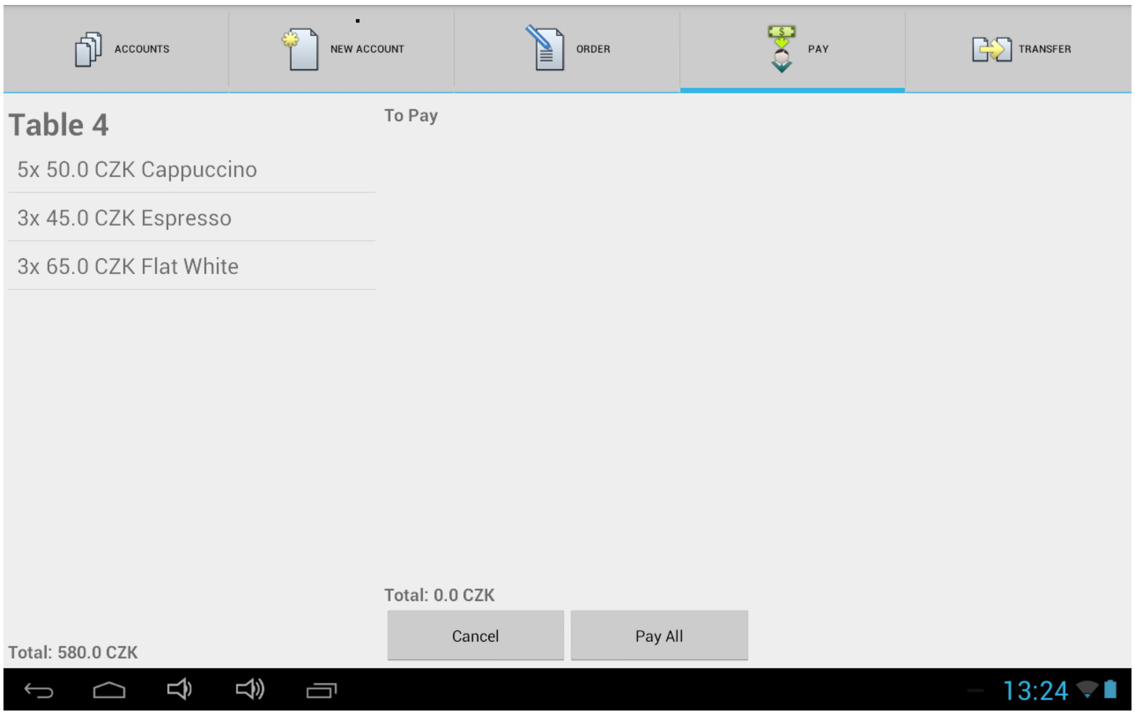


Table 4

5x 50.0 CZK Cappuccino

3x 45.0 CZK Espresso

3x 65.0 CZK Flat White

Total: 580.0 CZK

To Pay

Total: 0.0 CZK

Figure 21: Payment Screen

10.2.6 TRANSFER ITEMS SCREEN

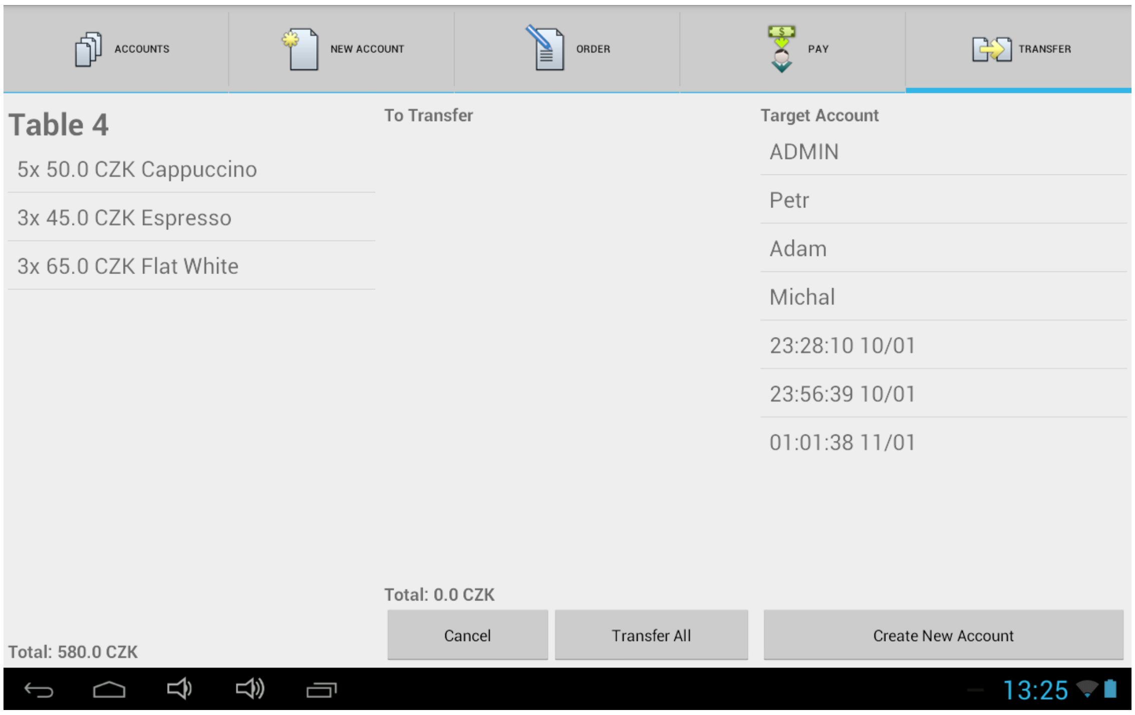


Table 4

5x 50.0 CZK Cappuccino

3x 45.0 CZK Espresso

3x 65.0 CZK Flat White

Total: 580.0 CZK

To Transfer

Total: 0.0 CZK

Target Account

ADMIN

Petr

Adam

Michal

23:28:10 10/01

23:56:39 10/01

01:01:38 11/01

Figure 22: Transfer Items Screen

10.3 NEW IDEAS FOR CASHBOB

Throughout the time spent developing this application I had a few ideas of extensions for CashBob:

- Item price is saved in one currency. According to settings, application will automatically convert price to another currency.
- If item will be selected, application will show side dishes for that item (for example: if grilled chicken will be selected, application will offer add potatoes or bread).
- Possibility to list in previous payments and reprint them.
- Make application to show the cook new orders and inform the waiters of the order that items are prepared to be served.
- Add meal menu to a menu card-one item in menu card can order more items (e.g. menu1- select soup and chicken).
- List of ingredients will be available for waiters within the application.
- Ordered items of an account will be paid for from a deposit (money payed to restaurant before any order was done) - good for company events in a restaurant.
- Possibility to select amount of an item (e.g. for red wine, we can select 1dcl or 2dcl).

11 LIST OF SHORTCUTS

GUI- Graphical User Interface

App- Application

API- Application Program Interface

SLOC- Source Line of Code

OS- Operation System

12 REFERENCES

[1] D. Bosomworth. Mobile Marketing Statistics 2015. 22 July 2015. Smart Insights [Online]. Available: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[2] Forecast for Global Shipments of Tablets, Laptops and Desktop PCs from 2010 to 2019 (in Million Units). 2015. Statista [Online]. Available: http://www.statista.com/statistics/272595/global-shipments-forecast-for-tablets-laptops-and-desktop-pcs/

[3] CashBob [Online]. Available: http://cashbob.cz/

[4] CVUT DSpace [Online]. Available: https://dspace.cvut.cz/discover?scope=%2F&query=cashbob&submit=&rpp=10&sort_by=dc.date.issued_dt&order=asc

[5] Restaurant System Mobile Client. CVUT DSpace [Online]. Available: https://dspace.cvut.cz/handle/10467/61643

[6] SWOT Analysis. Wikipedia [Online]. Available: https://en.wikipedia.org/wiki/SWOT_analysis

[7] FURPS. Wikipedia [Online]. Available: https://en.wikipedia.org/wiki/FURPS

[8] Architect’s Checklist: a Guide to FURPS+. Brent Arias [Online]. Available: http://www.ariasamp.net/brain-dump/guide-to-furps/

[9] Satzinger, John, Robert Jackson, Stephen Burd. System Analysis and Design in a Changing World. 7th ed. 2015. Cengage Learning [Print]

[10] Sommerville, Ian. Software Engineering. 9th ed. 2009. Pearson [Print]

[11] Function Point. Wikipedia [Online]. Available: https://en.wikipedia.org/wiki/Function_point

[12] Dashboards. 2 November 2015. Android [Online]. Available: https://developer.android.com/about/dashboards/index.html

[13] Android Studio. Android [Online]. Available: http://developer.android.com/sdk/index.html

[14] Function Points [Online]. Available: http://www.functionpoints.org/

[15] Virtual Box [Online]. Available: https://www.virtualbox.org/

[16] Why You Only Need to Test with 5 Users. J. NIELSEN, 19 March 2000. Nielsen Norman Group [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

13 LIST OF FIGURES

14 LIST OF TABLES

15 APPENDIX: MANUAL

CashBob is a solution for restaurant management to deliver a new and easier way of how to manage their restaurant and carry out all activities related to restaurants. CashBob mobile application would be used by waiters in restaurants and offer them ability to manage accounts, orders and do activities related to it.

15.1 FIRST LOGIN

- Fill your **Server IP** and **Currency** under button **Settings.** Server IP would be filled in format IP:
- Fill your **User Name** and **Password.** If your account does not have attached role WAITER, application would not be accessible for you.
- Click **Connect**

15.2 CREATE AN ACCOUNT

- Select tab **New Account**
- Create an account name (optional)
- Select a table (optional)
- Click **Create Account**

Note: If a name of an account will not be created but the table will be selected, the name of account will be selected table.

Note: If a name of an account will not be created and the table will not be selected, quick account will be created and account name will be time when account was created.

15.3 MAKE AN ORDER

- Select tab **Accounts**
- Select an account when you want do an order (this will move you to the tab **Order**)
- If you want to order an item that you ordered to this account before, click that item there
- If you want to order a new item, find it in the menu card
- If you want remove an item from the new order, click to the item in order
- Confirm your order by clicking on **Order**

15.4 ORIENTATION IN MENU CARD

Menu card contains more menus and items, which would be ordered to an account. If you want to show items in submenu, click on that item.

To get back to a menu one level back, click the button **UP**

To get back to root menu list, click the button **HOME**

To add an item to an order list, click on that item in menu card

15.5 MAKE PAYMENT

- Select tab **Accounts** and select account to make a payment on
- Select tab **Pay**
- Add item to pay into list of items to pay by clicking to them
- If you want to remove items from item list to pay, click on it
- Click on **Pay**
- Confirm that by clicking on button **Pay** again

15.6 MOVE ITEM BETWEEN ACCOUNTS

- Select tab **Accounts** and select account to make a payment on
- Select tab **Transfer**
- Add item to pay for into the list of items to move by clicking to them
- If you want to remove items from the item list to move, click on it
- Select the account where you want to move these items, you can create a quick account by clicking on the button **Create Quick Account**
- Click on **Transfer**

16 APPENDIX: COMPACT DISK

To thesis has been attached a compact disk. It contains:

- CashBob (Application source code)
- CashBob.apk (Android application package)
- Thesis_Tomas_Hogenauer.pdf (thesis text in PDF file)