School of Electronic Engineering and Computer Science

MSc Software Engineering

Project Report 2018

Intelligent Expense Manager

Natalia Govor

August 2018



Acknowledgements

I would first like to thank my project supervisor Dr. Thomas Roelleke of the School of Electronic Engineering and Computer Science at Queen Mary University of London, for his valuable guidance and constructive suggestions during planning and implementation of this project. I am particularly grateful for his useful critiques of my research and evaluation parts of the project. He always allowed this paper to be my own work but drove me in the right direction whenever I needed it.

I wish to acknowledge the help provided by Jane Reid, PG Project Coordinator of the School of Electronic Engineering and Computer Science at Queen Mary University of London. Her prompt assistance and consultations were greatly appreciated.

I would also like to extend my thanks to my former colleague and good friend, QA Engineer, Alexandra Zhelyaeva, for her recommendations on the testing process, provided time-management advice and encouragement in keeping my progress on schedule.

Special thanks should be given to my boyfriend Nick Lucas, for his technical recommendations and indication of easier solutions, valuable comments, and unfailing support.

Finally, I want to thank my parents and other friends for providing me with constant support and encouragement throughout my studies and writing this report.

**Abstract**

Table of contents

[Introduction 6](#_Toc522490296)

[1. Background research 7](#_Toc522490297)

[1.1 Analysis of flagships’ accountancy solutions 7](#_Toc522490298)

[1.1.1 SAP accountancy solutions 7](#_Toc522490299)

[1.1.2 Oracle accountancy solutions 9](#_Toc522490300)

[1.1.3 Microsoft accountancy solutions 10](#_Toc522490301)

[1.1.4 Sage accountancy solutions 12](#_Toc522490302)

[1.2 Overspending issues and microloans system in the UK 13](#_Toc522490303)

[1.3 Analysis of existing budgeting solutions for individuals 16](#_Toc522490304)

[1.4 Identifying the gap 17](#_Toc522490305)

[2. Requirements for Intelligent Expense Manager 19](#_Toc522490306)

[3. Design 21](#_Toc522490307)

[3.1 Monetisation strategies 21](#_Toc522490308)

[3.2 UI design 24](#_Toc522490309)

[3.3 Architecture design 26](#_Toc522490310)

[3.4 Database design 29](#_Toc522490311)

[4. Implementation 33](#_Toc522490312)

[4.1 Server-side implementation 33](#_Toc522490313)

[4.2 Client-side implementation 34](#_Toc522490314)

[4.3 Implementation of smart notification algorithm 36](#_Toc522490315)

[4.4 Final application screenshots 41](#_Toc522490316)

[5. Testing and evaluation 42](#_Toc522490317)

[5.1 Testing of correctness 43](#_Toc522490318)

[5.2 Testing of helpfulness and user experience 46](#_Toc522490319)

[6. Conclusions 49](#_Toc522490320)

[7. Further work 49](#_Toc522490321)

[References 50](#_Toc522490322)

[Appendix A. List of requirements for Intelligent Expense Manager 53](#_Toc522490323)

[Appendix B. List of performed test cases to check correctness of Intelligent Expense Manager 56](#_Toc522490324)

# Introduction

Finance management is one of the most common problem different subjects encounter daily. Companies analyse their incomes and outcomes in order to distribute their assets better and gain more profit from their business. Ordinary people also have to control their money. Algorithm is similar, but they operate with smaller amounts and have less rules and risks in their mundane life. However, inefficient control of money can lead individuals to dramatic consequences. High overspendings and approaching mandatory bills, such as rent, can force individuals to seek for credit. Some types of credit have very high interest rate, and in the worst case may bring individuals into loan cycle; this is the situation when person takes new loan just to pay the previous one. Loan cycles are very dangerous and can lead to severe financial problems.

Taking everything into account, expense management is a very important aspect of welfare and people should be able to deal with it easily and efficiently. This project investigates existing accounting solutions and focuses on building the prototype which attempts to improve the competitors’ applications. The emphasis is done on the development of a preventive algorithm to help people avoid overspending. The report contains seven chapters and is organised as follows. Chapter one contains an overview of existing solutions and presents the necessary background information. Chapter two introduces the requirements for the building application. Chapter three and four discuss design and implementation issues and challenges. In chapter five the implemented software is evaluated. Finally, chapters six and seven conclude the paper and describe the future improvements of the project.

# 1. Background research

## 1.1 Analysis of flagships’ accountancy solutions

The problem of finance management is not new, and plenty solutions have been already developed to address it. To learn more about important issues and aspects of accounting, the large business solutions from flagship corporations have been investigated and analysed. These solutions are oriented on mainly large and medium-sized businesses and are quite expensive. Companies which can afford those solutions have many complex processes to maintain; therefore, by buying accounting solutions they can gain greater control of their finances, automate routine tasks and as a result make better decisions over time. Many experts and developers have been working on the flagships’ solutions for years, so their analysis might help to discover essential capabilities and give an idea of vital features which can be inherited to help individuals with tighter budgets.

Solutions of four corporations have been investigated: SAP, Oracle, Microsoft and Sage. The products of the first two companies (SAP, Oracle) are mainly intended for large and medium-sized enterprises, whereas the last two company (Microsoft, Sage) focuses on small and medium-sized. These companies do not provide free trial versions to individuals, so the analysis was done only by examining the information from their websites and official handbooks.

### 1.1.1 SAP accountancy solutions

SAP SE (Systems Application & Products in Data Processing) is a multinational software corporation that makes enterprise software to manage business operations and customer relations. (1) SAP website is very good structured and contains videos that describe the main features and benefits of their solutions. Some videos show parts of the workflow of mobile and/or desktop applications; this gives quite accurate representation of the products.

SAP Business Network Group includes Concur Technologies (2) which offer three interesting products:

1. Concur Expense;
2. Concur Travel;
3. Concur Invoice.

All these solutions integrate with the world most popular finance systems. Moreover, SAP provides a combination of their two products: Concur Travel & Expense.

Concur Expense presents automated, fast and accurate expense management. (3) It captures data from credit card transactions, e-receipts and receipt photos, automatically categorises and maps expenses and enforces policy compliance. Concur expense helps to get accurate and on-time expense reports which can be easily reviewed and approved by managers.

Concur Travel focuses on corporate travels and helps to simplify their bookings and control company costs. (4) The system provides searching and booking of flight, hotel, rental car and rail using multiple global distribution systems. Employees can attach receipt photos of their trip expenses; additionally, trip expenses are automatically imported from e-receipts and credit cards charges. These data collection methods help to create practically complete report which employees just need to submit after they return home.

The last product, Concur Invoice, aims to automate the invoice processing which helps company to reduce costs and achieve larger control over accounts payable. (5) The solution manages requests, requires validating of purchase requests before spending and matches invoices to received goods and services, saving companies time and money. Intelligence monitoring tools give organization complete visibility of employees’ expenditures and help better forecast cash, time payments and identify additional vendor savings and discounts.

### 1.1.2 Oracle accountancy solutions

Oracle Corporation is an American multinational computer technology corporation which specialises primarily in developing and marketing database software, cloud engineering systems and enterprise software products. (6) Oracle website has a complex structure which leads to difficulties in finding necessary solutions. It does not provide videos of specific products demonstration; only abstract description of benefits and main features is available. Consequently, it was hard to form concrete opinion about Oracle products and imagine how they should work. If the company is interested in purchasing Oracle products, they should contact Oracle and request a demo version.

One of the major Oracle application is Oracle E-Business Suite which contains applications that enable organisations to make better decisions, reduce costs and increase performance. The Oracle E-Business Suite provides a set of financial applications: Asset Lifecycle Management; Cash & Treasury Management; Credit-To-Cash; Financial Control & Reporting; Lease and Finance Management; Procure-To-Pay; Travel & Expense Management. (7)

Each of Oracle Financial application is devoted to some specific area of interest in business world. For instance, Asset Lifecycle Management (8) helps companies to manage different types of their assets and check their states during their lifecycle to intelligently manage budget; Cash & Treasury Management (9) supports core treasury operations and offers functionality to manage liquidity, interest rate and foreign exchange risk. Only Financial Control & Reporting and Travel & Expense Management are closely related to the area of the project interest, so will be described in more detail.

Financial Control & Reporting is a fully automated solution which offers range of possibilities from creating and managing transaction to consolidating and reporting results. (10) Built-in best practices facilitate robust planning and budgeting, help to save time and money and deliver more reliable, accurate data.

Travel & Expense Management solution aims to automate and streamline travel spend management. (11) It offers credit card import functionality, robust workflow approval which help to reduce the amount of time and effort require to submit, approve, process and pay expense reports. Moreover, the system promises user-friendly interface with flexible user preferences which should increase employee productivity and satisfaction.

### 1.1.3 Microsoft accountancy solutions

Microsoft has a line of enterprise resource planning (ERP) and customer relationship management (CRM) software applications which is called Microsoft Dynamics. (12) There are six separate products that compose the Microsoft Dynamics family:

1. Microsoft Dynamics AX;
2. Microsoft Dynamics GP;
3. Microsoft Dynamics NAV;
4. Microsoft Dynamics SL;
5. Microsoft Dynamics CRM;
6. Microsoft Dynamics RMS.

Microsoft has a quite difficult structure of the website, it was problematic to find and understand the differences between various products. There are no videos or screenshots of applications; however, each product is followed by detailed handbook with extended description of capabilities and available packages and guides how to buy them.

The interest of the research is only ERP solutions which are the first four products. Microsoft Dynamics AX is best suited for large enterprises whereas GP, NAV, and SL are intended for small and medium; and so are described in detail below.

Microsoft Dynamics GP (13) introduces management tools for various business parts, such as financial management and accounting, stock management and operations, sales and service, human resources and payroll. Dynamics GP product is accompanied by lightweight mobile applications to provide access to some specific tasks from everywhere. Financial management capabilities include such useful features as tracking multiple budgets in multiple currencies, keeping and audit trail of who modifies a budget and when, quick transferring information between customer and vendor. Moreover, task and reminders capability helps to proactively inform users when specific incidents occur, such as customers going over their credit limit or payables due.

Microsoft Dynamics NAV (14) is designed for growing businesses and contains more capabilities than Dynamics GP. It is a multi-language multi-currency business management solution which can adapt for specific business needs of a company when they change. It has the same basic financial capabilities as Dynamics GP such as working with budgets, logging of users’ modifications to the data and powerful reporting tools. Additionally, it provides prediction of a company’s liquidity, bank account management and international currency support in full compliance with euro legislation.

Microsoft Dynamics SL (15) solution is more focused on business management (project planning, material management, people management). It also supports financial management of the business, but the major part of the system is not devoted to accounting.

### 1.1.4 Sage accountancy solutions

Sage is a British multinational enterprise software company which began as a startup in 1981. (16) They help businesses of small and medium sizes to manage accounting and finances, payments, people and payroll. The Sage website is a bit intricate, it contains a few videos with products’ descriptions, but they are too abstract and do not give a clear picture of the software. Nevertheless, the website contains plain description of products and in addition Google Play and App Store have screenshots of mobile applications to obtain the better impression about the granted solutions. Only products related to accounting and finances have been investigated.

Sage Business Cloud Accounting (17) works with invoices, records transactions, displays performance statistics and syncs information across all devices using cloud storage. Invoices for done work can be created in the system and sent to customers. It is possible to track and manage overdue payments, send quotes for review and agreements by a customer. Accounting solution also allows to enter money in and money out transactions, as well as attach photos of receipts; a category, description and reference can be applied to transactions. The great focus of Sage product is on the facilitation of managing transaction and invoices which helps companies to run their business more professionally.

Sage Business Cloud Financials focused on growing and medium-sized businesses (18) and allows more additional and professional possibilities for accounting than Sage Business Accounting. Since the developing software is not going to focus on invoice managing it was decided not to analyse Sage Financials in details.

Sage solutions are not very expensive, it is just £10 for simple account each month, but the solution is not oriented on finance management of ordinary individuals.

Summary of main capabilities of flagships’ solutions is presented in Table 1.1.

|  |  |
| --- | --- |
| Company | Main capabilities |
| SAP | Capturing data from credit card transactions, e-receipts, receipt photos; automatic categorisation; report generation; invoice processing; invoices matching to received goods; monitor tools. |
| Oracle | Transaction creation and management; reporting tools; credit card import functionality; flexible user settings. |
| Microsoft | Budget tracking; multiple currencies support; audit of modifications; reminders capabilities; reporting tools. |
| Sage | Tracking and managing of overdue payments; entering money in and money out transactions; receipt photos attachment; transaction and invoice management. |

Table 1.1 Main capabilities of flagships’ accountancy solutions

## 1.2 Overspending issues and microloans system in the UK

As any business, ordinal people also should maintain their budgets and ensure that their outcomes do not exceed incomes. Moreover, it is very important to keep enough money in a savings account to be prepared for unforeseen circumstances.

Some events in life require to spend more money as people usually do, such as holidays. According to debt advice charity National Debtline, millions of Britons (16% which equivalent to around 7.9 million people) start 2018 with huge overspending after Christmas season; this is 11% more than the last year. (19) The poll has shown that more than half of the participants had not started to save money for Christmas before December arrived. (19) In a separate report, National Debtline published that 37% of Britons bought Christmas presents on credit, whereas the last year the number was 33%. (19)

Another research by the Money Advice Service revealed that two thirds (60%) of UK adults exceed the planned budget during their holidays, with the average £220 overspending. (20) One of the most common reason for unexpected costs is unrealistic budget; for instance, more expensive food and drinks (32%) and more expensive activities (22%) than expected. (20) Also, many people do not consider pre-holiday purchases when planning for a holiday. The research demonstrated that 68% of respondents save enough money for their holiday while 24% of UK adults have been on a holiday which they could not afford which caused the average debt of £421. (20)

These reports highlight the growing problem of reliance by consumers on credit. According to research from Compare the Market, the personal debt level is now the highest in recent history, with the average person owing over £8,000. (19) Credit unions across the country promote thrift and encourage their members to save money regularly to help them overcome financial difficulties. (21)

Several reasons influence such high levels of personal debts in the UK, including the more positive attitude towards borrowing in the country. (22) This more relaxed attitude is impacted by media and advertising. According to research (22), there are “more loans advertisements than savings advertisements” in the UK newspapers. Moreover, mostly positive emotional appeals are used in these advertisements which focuses only on the positive credit aspect and leads to “normalising debt”. (22)

The form of short-term credit known as a payday loan increased significantly in the UK from 2006-2012, resulting to media and public anxiety regarding their exceptionally high cost. (23) The original goal of payday loan was to lend a small amount to someone before their payday. When person receives salary, the loan must be repaid. Technological developments have simplified the access to payday loans (23); modern online platforms allow to get the credit the same day the application being accepted. Some people also enjoy the anonymity of the online process. (23)

Due to the availability of payday loans, people have started to use it for casual spending such as groceries, bills and car costs when they are out of cash. Almost 3 in 5 (59%) stated their payday loan was for something urgent and essential. (24) However, 1 in 4 (24%) of these people reported that they would have managed without the buy if a payday loan had not been accessible. (24)

A payday loan is a short-term and high-cost loan. The debtor provides a small-amount loan for a fee and expects it to be paid in full in a few weeks. Because the loans have such short terms, the interest rate is generally high, with some costing of nearly 400%. (25) If a borrower cannot pay the debt in full at the end of the term, the debtor will charge additional fees.

Payday loans could be a good tool for quick money in the case of emergency, but they might trap consumers in spirals of debt from which it is hard to escape. (21) People who usually borrow money using payday loans have variable wages or insecure work; they are self-employed or lost their job. (23) When the few weeks expire, such categories of consumers can be unable to repay the loan. Instead, they take another loan to reimburse the old one and again pay fees and other administrative costs. If this cycle continues, it can bring individuals to severe financial problems. Therefore, it is always better to prepare for unexpected costs by careful planning of expenses and saving money each month.

## 1.3 Analysis of existing budgeting solutions for individuals

Nowadays, there are many budgeting applications for individuals which are available as desktop, web or mobile applications. Analysis of three popular applications (YNAB, Goodbudget, MoneyLover) has been conducted to get an overview of the offered functionality of competitors on the market.

You need a budget (YNAB) (26) is a paid application which costs £6.99 per month and has 34 days trial period. The mantra of the application is “give every dollar a job” (26), and it supports the concept of having virtual pots of money for different things. Users divide their income into independent pots, such as groceries or rent, and if they overspend in an area they should move money from one to another pot. Additionally, YNAB encourages users to save money for larger expenses. It offers both desktop and mobile interfaces, options to enter expenses manually or sync bank accounts automatically.

The next competitor, Goodbudget (27), offers free and plus versions. Plus version offers more capabilities and costs $6 per month. It also divides users’ cash into pots of money for different things; authors call it “envelope budgeting method”. (27) Expense tracking allows to check envelope and bank balances. The app produces reports to view income and spending which should help users to modify their budgets if necessary.

MoneyLover (29), as previous competitor, offers free and premium versions; premium costs £7.99. MoneyLover offers manual input for each transaction and helps to track income and expenses. Expenses are put into categories which allows to display statistics over time and identify on what users spend their money. The application also show warning when you overspend. Furthermore, MoneyLover offers receipts scanning as many professional accountancy solutions.

Not only budget planning and management causes issues in everyday life of individuals. The second tedious activity is sharing bills, for example between friends or flatmates. One single bill can be easily split within small group; however, increasing number of bills even in small group is likely to become a problem. Some of the bills might be forgotten or misplaced, especially during travelling; as a result, someone may lose part of their money.

An application Splitwise (30) attempts to help people to solve the problem of shared expenses. The application focuses only on sharing bills, it does not provide capabilities for budget planning. Users can add friends, create groups with their friends and add expenses inside the group. Different splitting methods within group members are available. Additionally, the application shows balances between group members and sends reminders to people who owe money.

## 1.4 Identifying the gap

The research has shown that enterprise accountancy solutions offer vast of capabilities and cover plenty of important aspects related to finance management. Budgeting software for individual users also offer that functionality, but in a smaller scale.

During the research it was identified that existing budgeting applications has different approaches to solve the problem of overspending. In general, sending notification is used to inform users when the set limit has been reached; in other words, when overspending has occurred. Some applications (for instance, MoneyLover) just display warning inside the application (without push notification), so user can see it only when he/she manually opens the app. Other applications (YNAB and Goodbudget) use the system of pots and ask users to distribute the money between pots. Later, they request to redistribute the money if some pot reaches its limit.

Taking everything into account, part of applications notifies users only post factum when the budget limit was exceeded and users are in minus, whereas others try to partly solve this problem by pots approach that requires more complex setup and planning from users’ side and additional time and effort to redistribute money. Thereby, effective preventing algorithm to assist users in avoiding overspending forms a gap on the market. Smart notifications, which avert users from going to minus and inform when users’ financial state has been stabilised and returned to safe area, can be the unique feature of the project.

Furthermore, there is no application which combines budget management and bills sharing. As a result, users who use both applications have to duplicate some expenses. Interviews among such users were conducted to explore their experience and discover their thoughts about using both applications simultaneously. Ten users participated in the interview. Collected data is contradictory: 50% of users finds using both applications inconvenient and redundant, especially during their trips; 25% of users reckon that applications contain different functionality and should not be combined; 25% of users admit that combination might be useful but doubt that it is possible to elegantly incorporate all capabilities within one application. Since the majority of users agrees that using both applications is a bit disturbing, an attempt to merge budget management and expense sharing will be made.

# 2. Requirements for Intelligent Expense Manager

The list of Intelligent Expense Manager requirements has been created based on the background research described in the previous chapters. The requirements are divided into four logical groups: basic, budgeting aspect, sharing expenses aspect, statistics displaying. Only the core requirements are given below. The extended list of requirements can be found in Appendix A.

Basic requirements:

1. User should be able to register and sign in the application.
2. All entered user’s data should be synchronised and fully accessible between different devices (mobile, desktop).
3. User should be able to sign out from the application.

Requirements for the budgeting aspect:

1. User should be able to set a budget for a month.
2. User should be able to enter his/her everyday expenses.
3. While entering expense data, user should be able to set the following information: the date of expense, the description of expense, the amount of expense.
4. User should be able to edit and delete expenses.
5. **User should be notified if he/she is getting closer to the limit of the month.**

Requirements for the sharing expenses aspect:

1. User should be able to create a group for sharing expenses. To create a group user should specify the name and add minimum one friend to it.
2. While creating expense, user should be able to specify that it is shared expense, choose the group to which it belongs and select the splitting method within group members.
3. All shared expenses should be synchronised between group members.
4. User should be able to view the list of friends with the debts.

Requirements for statistics displaying:

1. User should be able to view statistics of his/her expenses.
2. User can choose the period for which the statistics should be displayed.

The most of requirements describe essential features, such as editing and deleting expenses; without them the user experience will be very constrained and frustrating. Budgeting requirement related to notifications (number 5, marked as bold) requires some intelligent algorithm to predict the proper time for notification and help user instead of annoying. The algorithm is described in detail in Implementation Chapter.

# 3. Design

## 3.1 Choice of monetisation strategies

Each project should not only help users to solve some problem, but also it should bring revenue to its creators. For choosing the best app monetisation options it is important to consider the purpose of the application and methods which are used by competitors. (31)

An application can be supplied as free or as one-time paid app. In the case of one-time paid app, users pay only once to download the app, and all features and updates should be available for free. (32) Free applications have several different strategies to make money which are described below in detail.

According to Statista, the amount of paid app downloads decreased remarkably in 2016. (33) The paid app description should be convincing enough for users to pay for it without sampling. Moreover, this model requires continuous search for new customers to provide stable revenue stream. As a result, proceeds generated from paid apps are expected to continue to decline in the upcoming years (33) and it is more preferably to focus on free app monetisation strategies.

The main revenue models for free apps are: in-app advertising, freemium, in-app purchases, sponsorship, collecting and selling data.

In-App advertising is very popular and simple approach to monetise an application. The idea is to display advertisements inside the app and get paid from the advertisement network. Advertisements can be displayed in different formats, such as banners, videos, native ads (which are seamlessly integrated into mobile application) or interstitial ads (ads during launch or inside the pop-ups appearing periodically). (31) In-app advertising strategy has multiple payment models: cost per click, cost per view, cost per install. (31) The drawback of this monetisation method is that it works well only with big audiences. Moreover, advertisements annoy users, so it is important to smartly integrate them inside the app. This easy strategy is used by almost all competitors’ apps described in the chapter 1.3 and can be effectively integrated in the projected application; however, it should not be used exclusively.

Another strategy, called Freemium, implies existence of two app versions: free and paid versions. Paid version contains premium features which offer additional useful functionality. The users download and use free version and then can purchase full version if they decide that it is worth paying. It is quite efficient monetisation option; however, it is very important to distribute carefully functions between versions. Free version should provide a proper app experience, but too many available capabilities may prevent users from buying premium version. (33) This is also very proficient strategy which fits well with the project structure; it is widely used by competitors.

Next, In-App purchases is extremely popular strategy which allows users to buy something in the application. Purchases can be consumable (products that can be used by user only once, for instance virtual moneys and extra lives in mobile games), non-consumable (features in the applications that are bought for permanent use, for instance filters in photo editors) and subscriptions. (31) Subscriptions approach is useful for applications, such as video and audio streaming, online newspapers or magazines. A subscription fee allows users to get a full access and unlock all features or content. (33) In-App purchases strategy is more oriented on games and cannot be easily adapted for Intelligent Expense Manager. Subscription option might work, for instance to remove advertisements, but it can be also implemented using freemium strategy which works better with this kind of the application.

Sponsorship is very difficult monetisation approach. The idea is to create a mobile application, gain a lot of users, and then find a sponsor company with similar or the same target audience and adjust the app’s design to match the sponsor’s brand. (31) The revenue can be split with the sponsor or sponsor can pay monthly fee; nonetheless, it is very challenging to find potential sponsor. Possibly, some bank or credit union may be interested to become a sponsor of this application. However, banks in general have already had their own applications, and credit unions only have assets to support themselves and their activities. So, this approach is questionable for the project and should not be selected as the major.

Ethically controversial and not the most widespread way of making money is collecting and selling personal data to third parties. (33) Some free mobile applications collect users’ information, such as their email addresses, personal preferences, social media accounts, and then sell their databases to interested companies. Moreover, it is advantageous to track what users do; behavioral data is very useful from a marketing point of view. (34) This technique might be powerful for the projected application; some banks are likely to purchase the data to provide better credit plans for their clients.

Unfortunately, the new GDPR regulation puts now restrictions on the abovementioned monetisation option. The GDPR is Europe’s new legislation for personal data protection. It is designed to “harmonise” data privacy laws across Europe and provide higher protection and rights to individuals. (35) Now it is clear responsibility for companies to attain the consent of people for processing their personal data. (35) Personal data is any information that relates to an identified individual. (36) Encrypted, pseudonymised or reversible anonymised data remains personal data and falls in the scope of the law. (36)

The GDPR does not mean that collecting and selling personal data is no longer available as monetisation strategy, but now the companies must comply with several rules to be able to use this approach. The data processing can only take place by the user’s consent. It should be explained very transparent and understandable to users that their personal data is collected and how it is used; some specific purposes should be explicit. (37) In practice, it should be done by displaying to user three checkboxes during the registration process which must not be pre-clicked:

1. “I do accept privacy policy.”
2. “I do consent for the use of my data for purposes of the app.”
3. “I do consent to give the data for marketing purposes.”

According to new rules, it is very important that users should agree to give their data for marketing purposes to allow the companies to sell their data to third parties. Although it is permitted to forbid access to the application, if user do not consent to marketing usage of the data.

To summarise, for Intelligent Expense Manager the best ways to gain revenue is the combination of in-app advertising, freemium, and collecting and selling personal data. Sponsorship method can also be considered, but due to its complexity it will be preferable not to rely on it. In the prototype version, the monetisation methods will not be implemented, but it is essential to consider them before the further design stages.

## 3.2 UI design

Donald Norman has introduced the interaction model which states that user faces with two gulfs while using the system: gulf of execution and gulf of evaluation. (38) During the gulf of execution, user tries to understand how to achieve the goal and what actions must be performed for it; during the gulf of evaluation, user perceives the feedback from the system and decides whether the intended goal was achieved or not. Norman claims that a good design should allow users to bridge these two gulfs easily and proposes principles which should help in creating successful design: visibility, natural mapping and feedback. (38)

The visibility principle states that in a well-designed system all important functions and controls should be visible, so the user can find them just by looking at the system. According to natural mapping, the designer should organise the interface by using physical analogies and cultural standards. For example, red colour symbolises danger/error state in most cultures, whereas green means safe/correct state. The last feedback principle implies that users should receive immediate and informative feedback after their actions. Appropriate feedback allows users to evaluate their actions and amend them if necessary to achieve the goal.

A medium-fidelity design has been created based on Norman’s design principles and is shown in Figure 3.1. The most important and frequent functions according to specification include abilities to: view expenses, view shared expenses and debts, add new expenses and view expense statistics. It was decided to create a tab menu via which user can access those most important functions. For adding new expense, round plus button is located at the bottom of All expenses and Shared expenses pages; this button should be displayed above all content and be visible all the time (when user scrolls expenses, the plus button should be visible). The plus button signifies creating something new and should be intuitively understandable by users. Moreover, such approach is used in many modern applications and users should be already familiar with this design decision. Less frequent functions, such as profile settings, signing out, will be available via three dots menu in the top-left corner of the application.

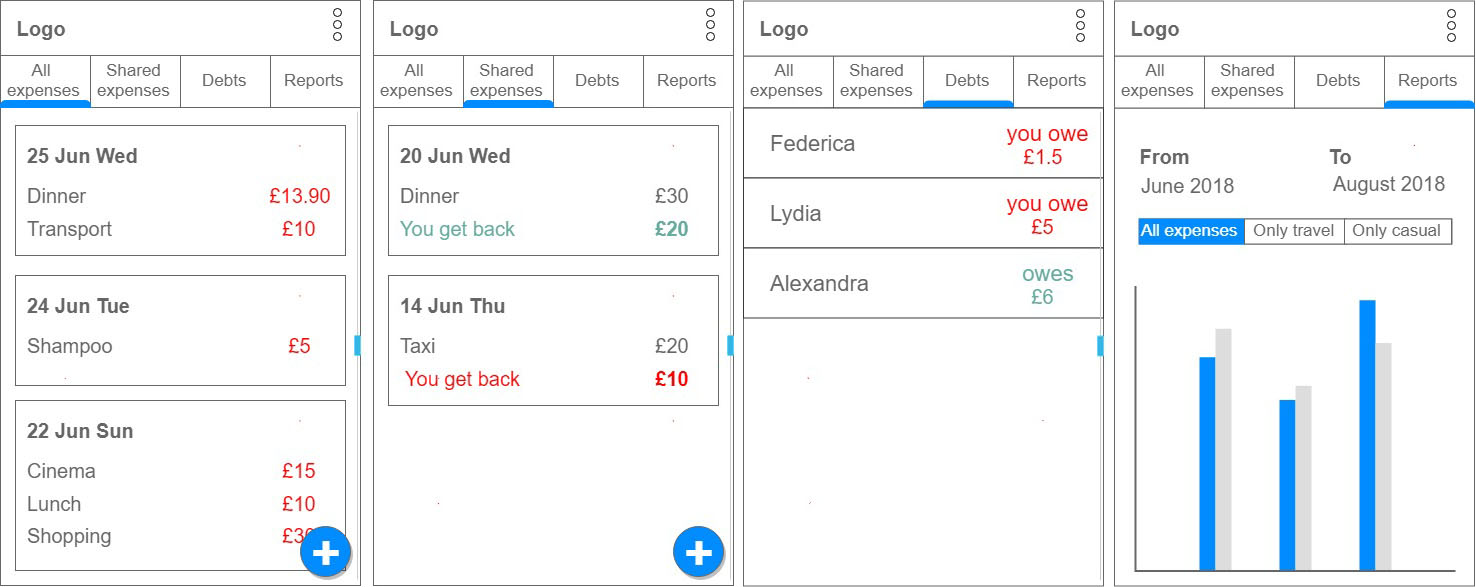


Figure 3.1. Low-fidelity design of Intelligent Expense Manager

Expenses are organized in chronological order as a vertically scrollable list; this corresponds to natural mapping principle. Additionally, colours are used to highlight spent money (red colour) and money which people should return to you (green colour). The overlay popup will be displayed to notify users when they are going closer to the set limit per month.

## 3.3 Architecture design

One of the most common architecture pattern is a client-server model which has many variations. The client-server architecture focuses on the partitioning processing between two or more machines. (39) Client machines request and receive services from a centralised server machine. (40) Ideally, clients are not aware of the specific hardware and software of the server machine and access it via standardised transparent interface. (40) Figure 3.2 shows a three-tier variation of client-server architecture.



Figure 3.2 A three-tier architecture (38)

In the three-tier architecture, many clients which can be users personal computers or devices send requests to the application server layer that isolates data processing and maximises object reuse. (39) The server comprises of the application server and the data store which can be located on separate machines. The described architecture fits well the demands of Intelligent Expense Manager. Many users should be able to connect to the server and retrieve the required information from the data store. Generally, the server is situated on more powerful machine (40) which will allow to perform faster complex balance calculations on the application server side. All information is stored in the data store accessed by the common for all clients application server; therefore, the requirement that users should be able to access their data from multiple devices is fulfilled by the selected architecture.

In a traditional client-server approach (Figure 3.3) which was introduced in the beginning of Internet era, each client interaction with webpage makes a new request to the server, and the server always responds with a whole HTML page. (41) This approach requires to redraw the entire webpage at the client side which leads to more bandwidth, delay in receiving result and unhappy user experience. (41)



Figure 3.3 Client-Server Request-Response Cycle (4)

To overcome those drawbacks and to meet new demands of serving the dynamic pages, Single Page Application (SPA) approach has been developed. In SPA (Figure 3.3) the client makes initial request and all required resources, such as CSS, images and scripts, are loaded at one time from the server. (41) Now when the client has an anitial page version and does any interaction with a page, the server responds with a JSON result only required by performed user’s action (41); the server do not generate and respond with the entire HTML page again. (41) At client-side only the particular part of the page should be refreshed after getting server response. SPA helps to reduce the time of every subsequent request because only necessary data is loading, increase the speed of page update and provide very pleasant user experience. (41)

SPA approach will be used at client-side of Intellgent Expense Manager because of its beneficial effect on user experience. Additionally, the standard server JSON response containing requested data allows to use the same server API by website and mobile applications.

## 3.4 Database design

To provide the necessary functionality the database of Intelligent Expense Manager should store the number of entities, such as users, users’ settings, expenses, friends, groups, common expenses between friends.

The User entity should contain user’s name, email and password. The user should be able to add friends with which he/she is going to share expenses. One user can have many friends and can be a friend of many other users; therefore, it is many-to-many relationship. Each user should also be able to enter settings for his profile (for example, how much money they want to spend during the month). The user can have only one set of settings, but it is decided to store them in the separate table UserSettings to provide more understandable structure (one-to-one relationship).

According to the specification, the user should be able to add his/her own expenses, as well as expenses which are shared with other users. To maintain these features, it was decided to create two separate entities: Expense and SharedExpense. Expense entity contains the essential information entered by user (description, date, amount), and if it is marked by user as shared then it has SharedExpenseId which is a foreign key to connect it with SharedExpense table. Moreover, Expense table should have an additional foreign key UserId which relates it with User table (one-to-many relationship).

Shared expenses can be created only inside the group; this is done to support splitting between group members. Group has a name and members. Each group can have many users and each user can be part of many groups; it is many-to-many relationship.

The shared expense belongs to the specific group (one-to-many relationship) and additionally contains a PaidBy foreign key to the User table. SharedExpense also includes amount, date and split type. The amount fields in SharedExpense and Expense tables contain different values. In Expense table it is the spending of the particular user, whereas in SharedExpense table amount holds the overall spending which is split between group members. The date field by default is the same in both tables, but this duplication allows the user to change the date only among his/her expenses.

It was decided to store the debts between the users in the database instead of calculating them each time when they are requested. Each shared expense is split between many users; it is many-to-many relationship. Therefore, additional table Debtor is needed, which stores the amount owed by user, UserId foreign key and SharedExpenseId foreign key.

The described relational design of the database is displayed in Figure 3.4. All many-to-many relationships are substituted by two one-to-many relationships on the design.



Figure 3.4 Normalised database design for Intelligent Expense Manager

The abovementioned database is designed in accordance with the first three normal forms. The database normalization helps to reduce data redundancy and improve data integrity. It provides many benefits; however, is has a significant downside related to system performance. (42) Complex relationships may lead to inefficient data retrievals when there are few updates and many join operations. (42) The concept of denormalization may increase the performance and provide more intuitive data structure but deteriorate data integrity. (42) Denormalisation should be adopted carefully based on the application purpose and ways the data will be used. (42)

For Intelligent Expense Manager it is possible to simplify the database structure by collapsing the number of logical objects in the current design. The Group and Debtor entities will be relatively static; as a result, it might be beneficial to remove GroupUser and Debtor tables by adding Members and Debtors array fields to Group and SharedExpense table respectively. The same approach can be applied for Friend table. Friends array field can be added to the User table; each item of Friends array should contain user id and name. Moreover, this approach allows more flexibility; it offers users to change names of their friends which will be displayed privately for them. Figure 3.5 presents the amended denormalised design.

The UserSettings table can be also collapsed and be stored as the embedded document inside the User table. However, these settings will be mostly used on the back-end side and only when the balance check will be requested; therefore, storing them separately will increase the bandwidth and allows easier accessibility for updates.



Figure 3.5 Denormalised database design for Intelligent Expense Manager

# 4. Implementation

## 4.1 Server-side implementation

ASP .NET Core has been selected for server part implementation. ASP.NET Core is an improved redesign of ASP.NET 4.x. (43) It is a cross-platform, high-performance and open-source framework (43) which allows to build modern Internet applications.

A web API has been created, and Figure 4.1 presents the basic design of the application.

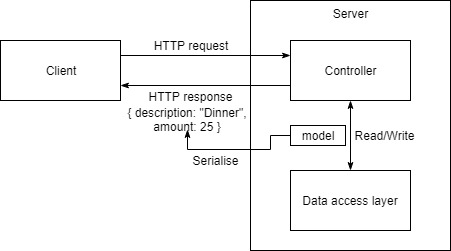


Figure 4.1 Design of server-side part of Intelligent Expense Manager

A client can be a mobile app or browser that consumes the web API. It sends HTTP request to the server and waits for HTTP response. A controller object handles HTTP requests from the client and generates HTTP responses. An application contains several controllers: AuthController, BalancesController, ExpensesController, SharedExpensesController, UserSettingsControole; each controller provides access to functionality related to certain entities. All controllers do not contain complex business logic; they accept HTTP requests, call the necessary services from Data access layer and pack the retrieved results to HTTP response.

A model is an object that represents the data in the application. Not all model objects exactly represent the entities from the database; some of them are modified or created specially to provide clients with more convenient data representation. Models are serialised to JSON format and passed to the client in the message body of HTTP responses.

Data access layer contains services and repositories which access and process the entities contained in the database. All repositories contain CRUD operations (create, read, update, delete); some other repositories additionally contain some specific operations (for example, UserRepository contains method GetByEmail). Services implement the business logic required by application specification. For example, BalanceService calculates users’ balances and evaluates the financial state of users. By using results of BalanceService, the client decides whether to show warning notification to user or not.

## 4.2 Client-side implementation

Angular is a framework for creating client applications in HTML and TypeScript. (44) Angular application has many components that are associated with an HTML template. (44) Component and template together define a view. Before a view is displayed, templates are modified according to components program logic and data. (44) For special functionality which is not directly associated with views, components use services. Services are injected into components as dependencies which makes the code modular and reusable. (44) Figure 4.2 shows how the basic Angular blocks are connected.

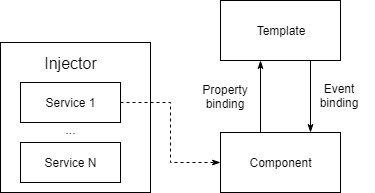


Figure 4.2 Client-side design of Intelligent Expense Manager

The client-side application is divided into two parts: secure-app and unsecure-app. Unsecure-app contains components, models and services which are used before user signs in the application. Secure-app describes the main functionality and contains its own models, services and feature views. Logic and data for specific app functionality is held by feature view. Feature views include: expenses, shared-expenses, new-expense, profile-settings, debts, reports. A view hierarchy is displayed in Figure 4.3. Secure-app services send requests to the web API to retrieve or change the data from the server.

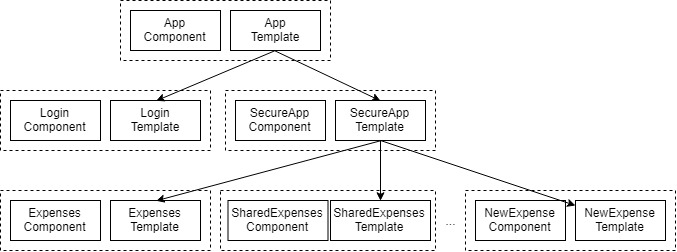


Figure 4.3 View hierarchy of Intelligent Expense Manager

In addition, the client-side project has common-services folder where services useful for both application parts are stored. It is HelpersService which contains method to access localStorage for caching and AuthService which allows users to sign in and sign out.

## 4.3 Implementation of smart notification algorithm

Display of notifications to help users to avoid overspending (requirement 5 in the budgeting section from chapter 2) should be discussed in detail as it demands some intelligent way of making a decision.

The goal of Intelligent Expense Manager is at least to prevent users from overspending, but ideally it should also help them to save more money. By entering or collecting user’s everyday transactions, the application accumulates important information about user’s expenditures which can be used to predict the future expenses and decide whether the user is going to meet his/her budget in this month or not. If the user is going to overspend, Intelligent Expense Manager should send a warning notification to the user.

The main concern is to determine the proper time when this warning notification should be send. Two important factors should be taken into account:

1. A notification should be helpful for the user; it should not be send too late or too early. Too late notification would not assist users to achieve the main goal of sticking to the budget; whereas too early might be annoying and fruitless.
2. Notifications should not disturb users too much; if user intentionally ignores that he/she is going further the limit, the application should stop be very intrusive with its notifications and inform user about critical balance in a more delicate form.

The easiest solution is to introduce a threshold and send the notification when the user reaches it. For example, the budget for a month is £600, for the threshold value we use 85% from the budget which equals £510. When the sum of user’s expenditures goes beyond the threshold value of £510, the manager should send the notification with the following information: “You are getting closer to your limit! Reduce your expenses if you do not want to overspend this month.” This algorithm is very straightforward, but not very powerful. The problem is that the user could reach the limit during the first few days of the month; in this case, the received notification would be belated, the user would not be able to adjust his/her expenditures to survive till the end of the month without going to minus.

The abovementioned solution can be improved in order to solve the problem with delayed warning, but it requires more complex configuration from the user’s side. At the beginning, the system does not have any data about previous user’s expenses; as a result, it is impossible to predict user’s spending and produce smart decision about useful notification time. Therefore, the system should ask user not only about his/her limit for this month, but also about his/her minimum casual spending. Casual spending can be separated for weekday, Saturday and Sunday.

Based on the data entered by user, the system will be able to calculate the minimum amount of money user requires to survive till the end of the month. Additional maximum limit parameter provided by user in combination with estimated minimum allows the system to calculate so-called “safety pillow” – the amount of money which user can spend without the risk to overspend during the month. If user spends more than some fixed percentage from this “safety pillow”, the system should notify him/her about the potential danger of going to minus. This percentage can be defined inside the system or it can be defined by user during configuration. A flow chart in Figure 4.4 demonstrates discussed algorithm.



Figure 4.4 Flow-chart of smart notification algorithm

The flow-chart in Figure 4.4 contains several external procedures, the most important CalculateNewBalance takes user settings and time period and calculates the balance and safety pillow which are used in taking the final decision. The implementation of CalculateNewBalance is shown in the flow-chart in Figure 4.5.



Figure 4.5 Flow-chart of procedure CalculateNewBalance

The system should monitor user’s behavior. If user proceeds to spend money unconsciously, the system does not send additional notifications, but it should unobtrusively notify the user about dangerous state, for instance by red interface of the application.

If the user cuts his/her costs, at some point the balance will stabilise and the “safety pillow” might increase back. When it reaches defined percentage value, the system should send new notification to user and return interface to green colours. The notification text can be: “Congratulations! You have controlled your expenses carefully and now you are going to meet the set budget by the end of the month”.

An example describes the algorithm more transparently. Suppose, there is a user Alice and she sets the maximum limit of £700 and the following budgets: weekday - £10, Saturday - £30, Sunday - £20. Assume the month have 28 days (4 weeks) to simplify calculations in the example. The system calculates the minimum which Alice requires for decent living till the end of the month based on her entered budgets: minimum limit = 5 \* £10 + 4 \* £30 + 4 \* £20 = £400. Based on the Alice’s maximum limit: safety pillow = £700 - £400 = £300. The default parameter 20% in the system defines that Alice can spend safely up to 80% of her “safety pillow” (£300 \* 0.8 = £240).

1. On the first day (Monday) Alice bought very expensive dress and spent £250. It is weekday, so Alice can spend £10 and the remaining £240 (£250 - £10 = £240) are deducted from her spare money. The “safety pillow” becomes £300 - £240 = £60. The hazardous limit for Alice is 20% which is £300 \* 0.2 = £60. Therefore, Alice has reached it and the system sends her the warning notification and changes its interface to red alert state.
2. The next day Alice again spent more than expected - £60. Her “safety pillow” becomes £0, but the system does not send additional warning.
3. Then, Alice starts to worry about her budget and for the following 10 days she lives very thriftily and spends only £50 (8 weekdays, one Saturday, one Sunday). Normally she should have spent 8 \* £10 + £30 + £20 = £130; as a result, she saved £80. This saved money goes now to top up her “safety pillow”; it becomes £80 which is more than 20% (£60) which means that Alice managed to stabilise her financial state. The system sends congratulation notification and switches the interface back to green.

The algorithm can be improved in the future after collecting a significant amount of user’s data by automatic recommendation of percentage parameter. In the current implementation it is set by the user. If the system has user’s expenses for the several previous months, it can calculate the average percentage and recommend user to use it.

## 4.4 Final application screenshots

The most important screens of the developed prototype are shown in Figure 4.6. They display the following pages: All expenses, Shared expenses, Create new expense, Profile settings. Additionally, warning and congratulation popups are attached among screenshots. On the screenshot with warning popup, the red application header can be seen that is used for all pages when the balance state becomes critical.

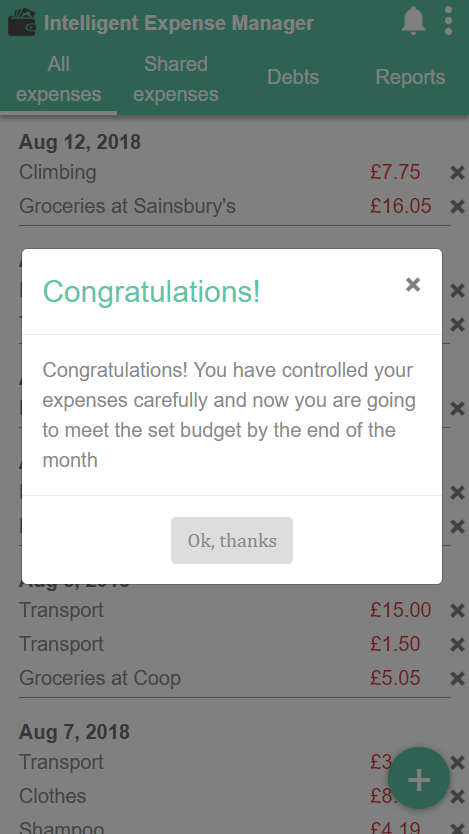
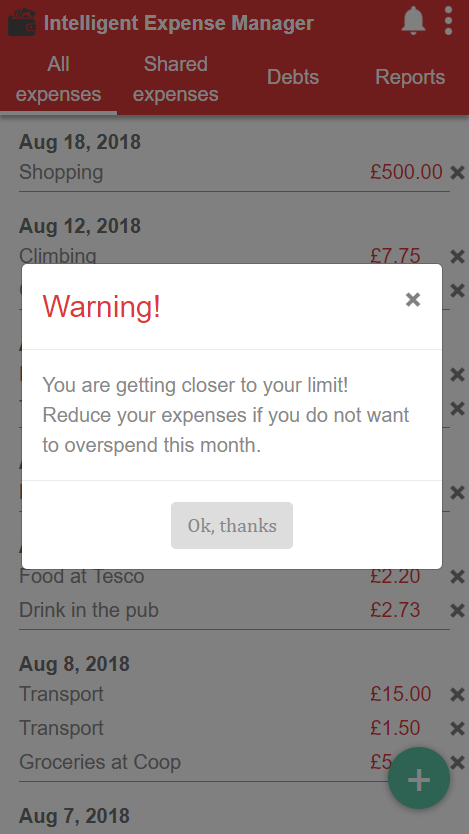
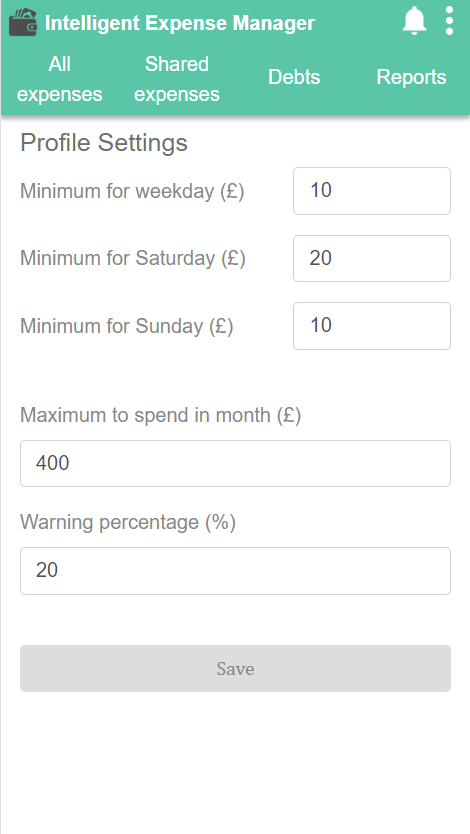
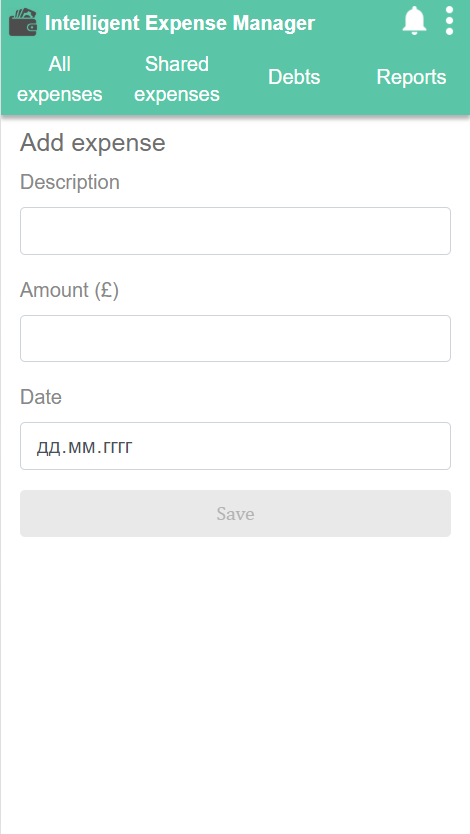
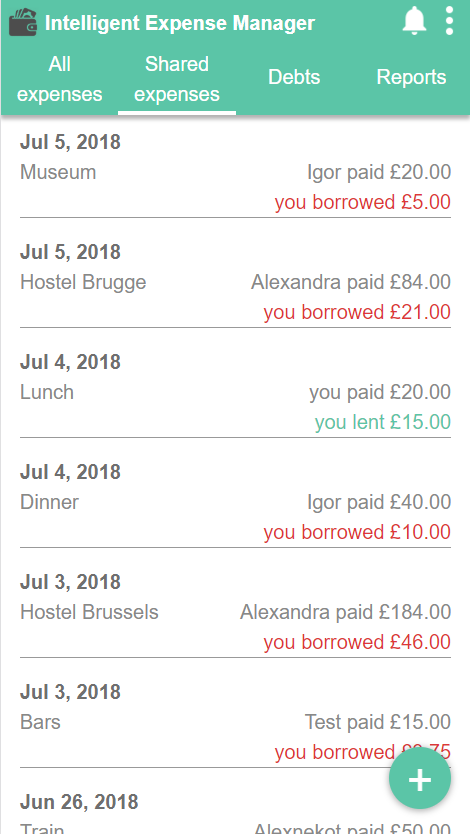
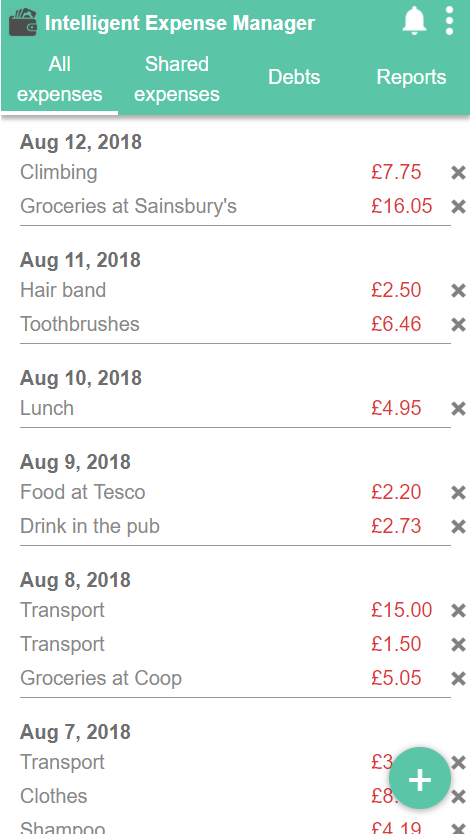


Figure 4.6 Final application screenshots

# 5. Testing and evaluation

To test and evaluate the project two approaches have been used:

1. Testing of correctness: all functions work according to specification;
2. Testing of helpfulness and user experience: the project achieves its goal to help people avoid overspending and save money without being intrusive and annoying.

## 5.1 Testing of correctness

For testing of correctness, a list of test cases was created based on the specification. Then, test cases were executed, and the actual results were compared to the expected results. Only functions implemented in the prototype were tested. All test cases were passed successfully. Only test cases related to the smart notification functionality are described below. The entire list of run test cases can be found in Appendix B.

To test the smart notification algorithm, it is required to know the user settings (minimum weekday, Saturday and Sunday spending, maximum to spend in a month, percentage for warning) and to calculate the safety pillow for the month using Formula 1. Test cases should be executed with initial empty list of expenses in the current month. Formula 2 calculates the critical amount of expense which should cause warning message and is used in many test cases. Formula 3 calculates the saved money till the current date with assumption that no expenses were done during the month till the current day. Table 5.1 contains the list and statuses of executed test cases for notification algorithm; all test cases were passed successfully.

SafetyPillow = MaximumToSpend – (WeekdaysNumber \* MinWeekday + (1)  
SaturdaysNumber \* MinSaturday + SundaysNumber \* MinSunday)

CriticalAmount = MinDaySpending + (SafetyPillow - SafetyPillow \* Percentage / 100) (2)

SavedMoney = PassedWeekdaysNumber \* MinWeekday + PassedSaturdaysNumber \* (3) MinSaturday + PassedSundayNumber \* MinSunday

|  |  |  |  |
| --- | --- | --- | --- |
| Test Num | Procedure | Expected Result | Result: Pass/Fail |
| 1 | Add expenses for the whole week, the sum of each day expenses should be equal to minimum day spending. | No popup notification. Interface is green. | Pass |
| 2 | Calculate the amounts using Formula 2 and Formula 3. Add two expenses in the same day which equals to the sum of calculated amounts. | After the first expense, no popup notification, interface is green. After the second expense, warning popup notification appears, interface becomes red. | Pass |
| 3 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Add another expense in the same day with any amount. | 1. Warning popup notification appears, interface becomes red.  2. No popup, interface remains red. | Pass |
| 4 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Delete the created expense. | 1. Warning popup notification appears, interface becomes red.  2. Congratulation popup notification appears, interface becomes green. | Pass |
| 5 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Add expense in the next day with the amount less than normal day spending. | 1. Warning popup notification appears, and interface becomes red.  2. Congratulation popup notification appears, interface becomes green. | Pass |
| 6 | Use only weekdays or set equal minimum day spending for all days in user settings.  1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Add expense in the next day with the amount equals two normal day spending.  3. Add three expenses in three different consecutive days with the amount half less than normal day spending. | 1. Warning popup notification appears, and interface becomes red.  2. No popup, interface remains red.  3. After two expenses, no popup, interface remains red. After the third expense, congratulation popup notification appears, interface becomes green. | Pass |
| 7 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Close the application.  3. Open the application again. | 1. Warning popup notification appears, and interface becomes red.  2. Application is closed.  3. No popup notification appears. Interface is red. | Pass |

Table 5.1 Test cases for notification algorithm

## 5.2 Testing of helpfulness and user experience

To evaluate the project objectives, three real users have been asked to use the developed prototype for one week, and then the interviews were held to ask them about their experience and impressions. Users were asked if they exceeded their week limit, if the app helped them to prevent this, if they were able to spend less than expected. All three users are familiar with software development and used budgeting applications before. The limitations of prototype version were explained to the users and specific guides how to use the prototype were given. The collected results and their analysis are provided below.

**User 1 settings**: minimum day spending = £15, minimum Saturday spending = £20, minimum Sunday spending = £10, maximum to spend during the week = £140, percentage for warning = 20%.

User 1 did not receive any notifications during the week and always saw green interface. User 1 admitted that the week was very quiet, and he/she did not spend a lot of money. The analysis of collected data confirmed that the user did not exceed the set maximum limit. The initial safety pillow for User 1 was £35, but after the week it became £48.97; hence, the user was even able to save some money. The current prototype does not contain any notifications to tell the user about his/her achievements in saving money. This can be considered as a further improvement.

**User 2 settings**: minimum day spending = £10, minimum Saturday spending = £20, minimum Sunday spending = £10, maximum to spend during the week = £110, percentage for warning = 20%.

User’s 2 experience was different from User 1. User 2 mentioned that he/she do not usually spend a lot of money and always try to live frugally. He/she has set desired limits in settings and believes that this reflects his/her expenditures in general. However, the week appeared to be tough, since User 2 had to pay for the phone, top up an Oyster card and additionally he/she had a small trip and party at the end of the week. In the middle of the week, User 2 received a warning, so the next day he/she did not spend any money on eating out and was able to stabilise the budget the next day (congratulations notification was received). But the remaining week expenses were unavoidable; therefore, after the second warning notification, User 2 continued to spend the money despite the red interface. User 2 noticed that it is likely that without the first warning notification, he/she would have spent even more money.

The analysis of User 2 data showed that £117.99 was spent instead of planned £110 maximum. The initial safety pillow was £30 and at the end of the week it became -£7.99. The algorithm worked correctly, and notifications received by User 2 were according to the specification. Unfortunately, User 2 was not able to stabilise the balance after the second warning due to mandatory expenses. However, one week is too short time period to make a conclusion about efficiency of algorithm. Longer tests (1-2 months) should be conducted to see how User 2 manage his/her budget. If the situation remains the same, then it is likely that User 2 underestimate his/her spending and the program can be improved to recommend better settings to user based on the history data.

**User 3 settings**: minimum day spending = £10, minimum Saturday spending = £10, minimum Sunday spending = £10, maximum to spend during the week = £86.5, percentage for warning = 20%.

User 3 said that he/she always has a problem with the budget and constantly overspend. Moreover, User 3 complained that the application showed the warning notification three times a week and it was a bit annoying. He/she said that the first warning was received on Monday, but then it disappeared on Tuesday. On Wednesday User 3 had to buy some medicine and again received the warning which disappeared the next day. After receiving the second congratulations notification, User 3 continued to spend money as usual (no special expensive purchases) and again received warning notification (the third during the week). The red interface remained with User 3 untill the end of the week, in spite of all attempts to spend less.

The analysis has shown that User’s 3 calculated minimum balance (based on three set minimums) was very close to the maximum to spend value. This caused a small safety pillow and therefore the user received notifications often. The initial safety pillow was £16.5 and became -£6.5 at the end. During the week User 3 spent £93 instead of planned £86.5. The overall expenditure is not very high and again longer tests should provide better results. However, since the user was always very close to the edge of safety pillow, the notifications became intrusive. This behavior can be improved by taking into account the time (do not send congratulation notification in the morning when new expenses are likely to appear soon) or by setting another condition for showing congratulations notification. Currently it is displayed when the safety pillow became more then 20% of its initial value, even if it is greater by £1. The threshold value which is added to 20% of safety pillow might enhance the algorithm, so the users are notified only when they reach more stable state.

# 6. Conclusions

TBD

# 7. Further work

Talk about additional features which can be found in appendix specification.

Also talk about improvements of notification sending algorithm. How it can be possibly improved by adding item-based recommendation algorithm to suggest users how they can reduce expenses.

Talk about design improvements, better adaptation for mobile platforms.

Тесты проводились всего неделю – непоказательно. Надо бы сделать более хороший прототип и проводить тест 2-3 месяца, чтобы получить более реалистичные результаты.

References

Finalise reference style (Vancouver is used).

1. <https://www.sap.com/corporate/en/company/history.html>

2. <https://www.sap.com/uk/products/financial-management/travel-expense-management.html>

3. <https://www.concur.com/en-us/expense-management>

4. <https://www.concur.com/en-us/travel-booking>

5. <https://www.concur.com/en-us/invoice-management>

6. <https://www.oracle.com/corporate/index.html>

7. <https://www.oracle.com/applications/ebusiness/products.html>

8. <http://www.oracle.com/us/products/applications/ebusiness/financials/053265.html>

9. <http://www.oracle.com/us/products/applications/ebusiness/financials/053268.html>

10. <http://www.oracle.com/us/products/applications/ebusiness/financials/053318.html>

11. <http://www.oracle.com/us/products/applications/ebusiness/financials/053350.html>

12. <https://www.softwareadvice.com/erp/microsoft-dynamics-brand/>

13. <https://dynamics.microsoft.com/en-gb/gp-overview/>

14. <https://dynamics.microsoft.com/en-gb/nav-overview/>

15. <https://dynamics.microsoft.com/en-gb/sl-overview/>

16. <https://www.sage.com/company/about-sage>

17. <https://www.sage.com/en-gb/sage-business-cloud/accounting/>

18. <https://www.sage.com/en-gb/sage-business-cloud/financials/>

19. <https://www.independent.co.uk/news/business/news/christmas-spending-debts-credit-card-payments-loan-british-people-britons-household-incomes-a8138251.html>

20. <https://www.moneyadviceservice.org.uk/en/corporate/press-release-average-uk-adult-overspends-on-holiday-by-220>

21. Lee B, Brierley J. UK government policy, credit unions, and payday loans. International Journal of Public Administration. 2017 Mar 21;40(4):348-60.

22. Mogaji E, Czarnecka B, Danbury AH, Farquhar JD. How we are tempted into debt: emotional appeals in loan advertising in UK newspapers.

23. Rowlingson K, Appleyard L, Gardner J. Payday lending in the UK: the regul (aris) ation of a necessary evil?. Journal of social policy. 2016 Jul;45(3):527-43.

24. <https://www.finder.com/uk/payday-loan-statistics>

25. <https://www.credit.com/loans/loan-articles/the-truth-about-payday-loans/>

26. <https://www.youneedabudget.com/method/>

27. <https://goodbudget.com/>

28. <https://www.yolt.com/>

29. <https://moneylover.me/>

30. <https://www.splitwise.com/>

31. <https://rubygarage.org/blog/how-do-free-apps-make-money>

32. <https://www.entrepreneur.com/article/228089>

33. <https://thinkmobiles.com/blog/how-do-free-apps-make-money/>

34. <https://buildfire.com/ways-monetize-mobile-app/>

35. <https://www.wired.co.uk/article/what-is-gdpr-uk-eu-legislation-compliance-summary-fines-2018>

36. <https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en>

37. Voigt P, Von dem Bussche A. The EU General Data Protection Regulation (GDPR). Springer; 2017.

38. Norman Design of everyday things

39. "Distributed Application Architecture" (PDF). Sun Microsystem. Archived from the original (PDF) on 6 April 2011. Retrieved 2009-06-16. <https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf>

40. <https://www.britannica.com/technology/client-server-architecture>

41. Jadhav MA, Sawant BR, Deshmukh A. Single page application using angularjs. International Journal of Computer Science and Information Technologies. 2015;6(3):2876-9.

42. Shin SK, Sanders GL. Denormalization strategies for data retrieval from data warehouses. Decision Support Systems. 2006 Oct 1;42(1):267-82.

43. <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1>

44. <https://angular.io/guide/architecture>

# Appendix A. List of requirements for Intelligent Expense Manager

Basic requirements:

1. User should be able to register and sign in the application.
2. User should be able to sign in via social networks Facebook).
3. All entered user’s data should be synchronised and fully accessible between different devices (mobile, desktop).
4. User should be able to sign out from the application.

Requirements for the budgeting aspect:

1. User should be able to set a budget for a month.
2. User should be able to enter his/her everyday expenses.
3. While entering expense data, user should be able to set the following information: the date of expense, the category of expense, the amount of expense, whether the expense was done during travelling (by default it is false).
4. User should be able to connect his/her bank account, and the card transactions should be recorded automatically.
5. The app should parse the information from credit card transactions and assign all required fields of expense automatically.
6. User should be able to edit and delete expenses.
7. **User should be notified if he/she is getting closer to the limit of the month.**
8. User should be able to select important dates in the calendar and the app will send a reminder to start saving money for this event in advance. The default reminder will send notification one month before, but it should be customisable by user.

Requirements for the sharing expenses aspect:

1. User should be able to create or add a friend.
2. User should be able to create a group for sharing expenses. To create a group user should specify the name and add minimum one friend to it.
3. User should be able to change the group name.
4. While creating expense, user should be able to specify that it is shared expense, choose the group to which it belongs and select the splitting method within group members.
5. User should be able to modify any added expense and set/change/delete for it splitting options.
6. The app should support the following splitting rules:
   1. The expense can be paid by one or several group members.
   2. The amount of expense can be split not equally and not between all group members.
7. All shared expenses should be synchronised between group members.
8. User should be able to view balances between group members.
9. User should be able to view the list of friends with the debts.
10. User should be able to add more friends to a group. Previous shared expenses in the group do not affect new added members.
11. User should be able to delete a friend from a group only if this friend does not have any debts within the group.

Requirements for displaying statistics:

1. User should be able to view statistics of his/her expenses.
2. User can choose the period for which the statistics should be displayed.
3. User should be able to switch the statistics display between:
   1. All expenses.
   2. Only travelling expenses.
   3. Only everyday expenses without travelling expenses).

# Appendix B. List of performed test cases to check correctness of Intelligent Expense Manager

|  |  |  |  |
| --- | --- | --- | --- |
| Test Num | Procedure | Expected Result | Result: Pass/Fail |
| 1 | Enter correct credentials on log in page. Press login button. | User is logged in. All expenses page is opened. | Pass |
| 2 | Enter incorrect email, but correct password. Press login button. | User is not logged in. | Pass |
| 3 | Enter correct email, but incorrect password. Press login button. | User is not logged in. | Pass |
| 4 | Log in the application. Then, log out from the application. | User is logged out. Login page appears. | Pass |
| 5 | Open All expenses page. | All user expenses from database are displayed and sorted by descending. | Pass |
| 6 | Open Shared expenses page. | All user only shared expenses from database are displayed and sorted by descending. For borrowed money the text is red; for lent money the text is green. | Pass |
| 7 | 1. On All expenses page press plus button.  2. Entered the valid data for expense and press save button. | 1. Page for creating an expense is opened.  2. All expenses page is opened. New expense appears in the list. | Pass |
| 8 | 1. Delete an expense from All expenses page by pressing cross button.  2. Refresh the page. | 1. Expense disappears from the list.  2. Deleted expense is still not in the list of all expenses. | Pass |
| 9 | 1. Go to Profile Setting page.  2. Compare the displayed user settings with the settings stored in the database. | 1. Profile page is opened. All inputs contain some data.  2. The data in the inputs is equal to the respective data in the database. | Pass |
| 10 | 1. Go to Profile Settings page.  2. Change the data in the inputs and press save button.  3. Refresh the page. | 1. Profile page is opened. All inputs contain some data.  2. The inputs contain new data.  3. The inputs contain new data which was entered and saved. | Pass |
| 11 | Add expenses for the whole week, the sum of each day expenses should be equal to minimum day spending. | No popup notification. Interface is green. | Pass |
| 12 | Calculate the amounts using Formula 2 and Formula 3. Add two expenses in the same day which equals to the sum of calculated amounts. | After the first expense, no popup notification, interface is green. After the second expense, warning popup notification appears, interface becomes red. | Pass |
| 13 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Add another expense in the same day with any amount. | 1. Warning popup notification appears, interface becomes red.  2. No popup, interface remains red. | Pass |
| 14 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Delete the created expense. | 1. Warning popup notification appears, interface becomes red.  2. Congratulation popup notification appears, interface becomes green. | Pass |
| 15 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Add expense in the next day with the amount less than normal day spending. | 1. Warning popup notification appears, and interface becomes red.  2. Congratulation popup notification appears, interface becomes green. | Pass |
| 16 | Use only weekdays or set equal minimum day spending for all days in user settings.  1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Add expense in the next day with the amount equals two normal day spending.  3. Add three expenses in three different consecutive days with the amount half less than normal day spending. | 1. Warning popup notification appears, and interface becomes red.  2. No popup, interface remains red.  3. After two expenses, no popup, interface remains red. After the third expense, congratulation popup notification appears, interface becomes green. | Pass |
| 17 | 1. Add expense with the amount equals the sum of Formulas 2 and 3.  2. Close the application.  3. Open the application again. | 1. Warning popup notification appears, and interface becomes red.  2. Application is closed.  3. No popup notification appears. Interface is red. | Pass |